Springer Handbooks of Computational Statistics

1

James E. Gentle
Wolfgang Karl Härdle
Yuichi Mori *Editors*

Handbook of Computational Statistics

Concepts and Methods

Second Edition

Springer

Springer Handbooks of Computational Statistics

Series Editors

James E. Gentle
Wolfgang K. Härdle
Yuichi Mori

For further volumes:
http://www.springer.com/series/7286

James E. Gentle • Wolfgang Karl Härdle
Yuichi Mori
Editors

Handbook of Computational Statistics

Concepts and Methods

Second revised and updated Edition

Springer

Editors
James E. Gentle
George Mason University
Dept. Computational and Data Sciences
Fairfax, VA
USA

Prof. Yuichi Mori
Okayama University
Dept. Socio-Information
Okayama
Japan

Wolfgang Karl Härdle
Humboldt-Universität zu Berlin
L.v.Bortkiewicz Chair of Statistics
C.A.S.E. Centre f. Appl. Stat. and Econ.
School of Business and Economics
Berlin
Germany

ISBN 978-3-642-21550-6 e-ISBN 978-3-642-21551-3
DOI 10.1007/978-3-642-21551-3
Springer Heidelberg Dordrecht London New York

Library of Congress Control Number: 2012938637

Contents

Contributors

Luc Bauwens Université Catholique de Louvain, Louvain-la-Neuve, Belgium, luc.bauwens@uclouvain.be

Claus Boyens 1&1 Internet AG, Karlsruhe, Germany, claus.boyens@web.de

Simon A. Broda Department of Quantitative Economics, University of Amsterdam, Amsterdam, The Netherlands, s.a.broda@uva.nl

Peter Bühlmann ETH Zürich, Seminar für Statistik, Zürich, Switzerland, buhlmann@stat.math.ethz.ch

Siddhartha Chib Olin Business School, Washington University in St. Louis, St. Louis, MO, USA, chib@wustl.edu

Pavel Čížek Department of Econometrics & Operations Research, Tilburg University, Tilburg, The Netherlands, P.Cizek@uvt.nl

Lenka Čížková Department of Econometrics & Operations Research, Tilburg University, Tilburg, The Netherlands, lenka@lenka-photography.eu

Laurie Davies Universität Duisburg-Essen, Essen, Germany, laurie.davies@uni-due.de

William F. Eddy Department of Statistics, Carnegie Mellon University, Pittsburgh, PA, USA, bill@stat.cmu.edu

Ursula Gather Technische Universität Dortmund, Dortmund, Germany, gather@statistik.tu-dortmund.de

James E. Gentle Department of Computational and Data Sciences, George Mason University, Fairfax, VA, USA, jgentle@gmu.edu

Oliver Günther Universität Potsdam, Präsidialamt Potsdam, Germany, guenther@wiwi.hu-berlin.de

Wolfgang Karl Härdle Humboldt-Universität zu Berlin, L.v.Bortkiewicz Chair of Statistics, C.A.S.E. – Centre for Applied Statistics and Economics, School of Business and Economics, Berlin, Germany, haerdle@wiwi.hu-berlin.de

Joel L. Horowitz Department of Economics, Northwestern University, Evanston, IL, USA, joel-horowitz@northwestern.edu

Alfred Inselberg School of Mathematical Sciences, Tel Aviv University, Tel Aviv, Israel, aiisreal@post.tau.ac.il

Toshinari Kamakura Chuo University, Tokyo, Japan, kamakura@indsys.chuo-u.ac.jp

Jack P.C. Kleijnen Department of Information Management/Center for Economic Research (CentER), Tilburg University, Tilburg, The Netherlands, Kleijnen@UvT.NL

Sigbert Klinke Ladislaus von Bortkiewicz Chair of Statistics, C.A.S.E. – Center for Applied Statistics and Economics, Humboldt- Universität zu Berlin, Berlin, Germany, sigbert@wiwi.hu-berlin.de

Thriyambakam Krishnan Mu-Sigma Business Solutions Pvt. Ltd, Kalyani Platina, K.R. Puram Hobli, Bangalore, India, krishnant001@gmail.com

Pierre L'Ecuyer Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, Montréal (Québec), Canada, lecuyer@iro.umontreal.ca

Pavel Laskov University of Tübingen, Tübingen, Germany, laskov@first.fhg.de

Hans-J. Lenz Institut für Statistik und Ökonometrie, Freie Universität Berlin, Berlin, Germany, Hans-J.Lenz@fu-berlin.de

Catherine Loader Department of Statistics, Case Western Reserve University, Cleveland, OH, USA, catherine@case.edu

Enno Mammen University of Mannheim, Mannheim, Germany, emammen@rumms.uni-mannheim.de

David J. Marchette Naval Surface Warfare Center, Dahlgren, VA, USA, david.marchette@navy.mil

Geoffrey. J. McLachlan Department of Mathematics, University of Queensland, Brisbane, QLD, Australia, g.mclachlan@uq.edu.au

Rebecca L. McNamee Department of Statistics, Carnegie Mellon University, Pittsburgh, PA, USA

Sebastian Mika idalab GmbH, Berlin, Germany, mika@idalab.de; mika@first.fhg.de

Adam Misiorek Santander Consumer Bank S.A., Wrocław, Poland, adam.misiorek@santanderconsumer.pl

Masahiro Mizuta Information Initiative Center, Hokkaido University, Sapporo, Japan, mizuta@iic.hokudai.ac.jp

John F Monahan Department of Statistics, North Carolina State University, Raleigh, NC, USA, monahan@ncsu.edu

Yuichi Mori Department of Socio-information, Okayama University, Okayama, Japan, mori@soci.ous.ac.jp

Klaus-Robert Müller Berlin Institute of Technology, Berlin, Germany, klaus-robert.mueller@tu-berlin.de

Marlene Müller Beuth University of Applied Sciences, Berlin, Germany, marlene.mueller@beuth-hochschule.de

Junji Nakano Department of Data Science, The Institute of Statistical Mathematics, Tachikawa, Tokyo, Japan, nakanoj@ism.ac.jp

Swagata Nandi Indian Statistical Institute, Delhi Centre, New Delhi, India, nandi@isid.ac.in

Shu Kay Ng School of Medicine, Griffith University, Meadowbrook, QLD, Australia, s.ng@griffith.edu.au

Marc S. Paolella Swiss Banking Institute, University of Zurich, Zurich, Switzerland
and
Swiss Finance Institute, Zurich, Switzerland, paolella@isb.uzh.ch

Konrad Rieck Berlin Institute of Technology, Berlin, Germany, konrad.rieck@tu-berlin.de

Christian P. Robert Université Paris-Dauphine, CEREMADE, and CREST-INSEE, Paris, France, Christian.Robert@ceremade.dauphine.fr

Jeroen V.K. Rombouts HEC Montréal, CIRANO, CIRPEE, CORE, Montreal, Canada, jeroen.rombouts@hec.ca

Christin Schäfer Fraunhofer Institute FIRST, Berlin, Germany, christin@first.fhg.de

David W. Scott Department of Statistics, Rice University, Houston, TX, USA, scottdw@rice.edu

Sören Sonnenburg Berlin Institute of Technology, Berlin, Germany

James C. Spall The Johns Hopkins University, Applied Physics Laboratory, Laurel, MD, USA, James.Spall@jhuapl.edu

Masashi Sugiyama Tokyo Institute of Technology, Meguro-ku, Tokyo, Japan, sugi@cs.titech.ac.jp

Jürgen Symanzik Department of Mathematics and Statistics, Utah State University, Logan, UT, USA, juergen.symanzik@usu.edu

David Tax Delft University of Technology, Delft, The Netherlands, ldavidt@first.fhg.de

Iosif I. Vaisman Department of Bioinformatics and Computational Biology, George Mason University, Fairfax, VA, USA, ivaisman@gmu.edu

Brani Vidakovic The Wallace H. Coulter Department of Biomedical Engineering, Georgia Institute of Technology, Atlanta, GA, USA, brani@bme.gatech.edu

Miroslav Virius Czech Technical University in Prague, Faculty of Nuclear Sciences and Physical Engineering, Prague, Czech Republic, miroslav.virius@fjfi.cvut.cz

Yuedong Wang Department of Statistics and Applied Probability, University of California, Santa Barbara, CA, USA, yuedong@pstat.ucsb.edu

Rafał Weron Institute of Organization and Management, Wrocław University of Technology, Wrocław, Poland, rafal.weron@gmail.com

Adalbert Wilhelm Commerzbank Chair of Information Management, School of Humanities and Social Sciences, Jacobs University Bremen gGmbH, Bremen, Germany, a.wilhelm@jacobs-university.de

Leland Wilkinson SYSTAT Software Inc., Chicago, IL, USA, Leland.Wilkinson@systat.com

Heping Zhang Yale University School of Medicine, New Haven, CT, USA
and
Sun Yat-Sen University, Guangzhou, China, heping.zhang@yale.edu

Part I
Computational Statistics

Chapter 1
How Computational Statistics Became the Backbone of Modern Data Science

James E. Gentle, Wolfgang Karl Härdle, and Yuichi Mori

This first chapter serves as an introduction and overview for a collection of articles surveying the current state of the science of computational statistics. Earlier versions of most of these articles appeared in the first edition of *Handbook of Computational Statistics: Concepts and Methods*, published in 2004.

There have been advances in all of the areas of computational statistics, so we feel that it is time to revise and update this *Handbook*. This introduction is a revision of the introductory chapter of the first edition.

1.1 Computational Statistics and Data Analysis

To do data analysis is to do computing. Statisticians have always been heavy users of whatever computing facilities are available to them. As the computing facilities have become more powerful over the years, those facilities have obviously decreased the amount of effort the statistician must expend to do routine analyses. As the computing facilities have become more powerful, an opposite result has occurred, however; the computational aspect of the statistician's work has increased. This is because of paradigm shifts in statistical analysis that are enabled by the computer.

J.E. Gentle (✉)
Department of Computational and Data Sciences, George Mason University, Fairfax, VA, USA
e-mail: jgentle@gmu.edu

W.K. Härdle
Humboldt-Universität zu Berlin, L.v.Bortkiewicz Chair of Statistics, C.A.S.E. - Centre for Applied Statistics and Economics, School of Business and Economics, Berlin, Germany
e-mail: haerdle@wiwi.hu-berlin.de

Y. Mori
Department of Socio-information, Okayama University, Okayama, Japan
e-mail: mori@soci.ous.ac.jp

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_1,

Statistical analysis involves use of observational data together with domain knowledge to develop a model to study and understand a data-generating process. The data analysis is used to refine the model or possibly to select a different model, to determine appropriate values for terms in the model, and to use the model to make inferences concerning the process. This has been the paradigm followed by statisticians for centuries. The advances in statistical theory over the past two centuries have not changed the paradigm, but they have improved the specific methods. The advances in computational power have enabled newer and more complicated statistical methods. Not only has the exponentially-increasing computational power allowed use of more detailed and better models, however, it has shifted the paradigm slightly. Many alternative views of the data can be examined. Many different models can be explored. Massive amounts of simulated data can be used to study the model/data possibilities.

When exact models are mathematically intractable, approximate methods, which are often based on asymptotics, or methods based on estimated quantities must be employed. Advances in computational power and developments in theory have made ***computational inference*** a viable and useful alternative to the standard methods of asymptotic inference in traditional statistics. Computational inference is based on simulation of statistical models.

The ability to perform large numbers of computations almost instantaneously and to display graphical representations of results immediately has opened many new possibilities for statistical analysis. The hardware and software to perform these operations are readily available and are accessible to statisticians with no special expertise in computer science. This has resulted in a two-way feedback between statistical theory and statistical computing. The advances in statistical computing suggest new methods and development of supporting theory; conversely, the advances in theory and methods necessitate new computational methods.

Computing facilitates the development of statistical theory in two ways. One way is the use of symbolic computational packages to help in mathematical derivations (particularly in reducing the occurrences of errors in going from one line to the next!). The other way is in the quick exploration of promising (or unpromising!) methods by simulations. In a more formal sense also, simulations allow evaluation and comparison of statistical methods under various alternatives. This is a widely-used research method. For example, 66 out of 79 articles published in the Theory and Methods section of the *Journal of the American Statistical Association* in 2010 reported on Monte Carlo studies of the performance of statistical methods. (In 2002, this number was 50 out of 61 articles.) A general outline of many research articles in statistics is

1. State the problem and summarize previous work on it.
2. Describe a new approach.
3. Work out some asymptotic properties of the new approach.
4. Conduct a Monte Carlo study showing the new approach in a favorable light.

Much of the effort in mathematical statistics has been directed toward the easy problems of exploration of asymptotic properties. The harder problems for finite

samples require different methods. Carefully conducted and reported Monte Carlo studies often provide more useful information on the relative merits of statistical methods in finite samples from a range of model scenarios.

While to do data analysis is to compute, we do not identify all data analysis, which necessarily uses the computer, as "statistical computing" or as "computational statistics". By these phrases we mean something more than just using a statistical software package to do a standard analysis. We use the term "statistical computing" to refer to the computational methods that enable statistical methods. Statistical computing includes numerical analysis, database methodology, computer graphics, software engineering, and the computer/human interface. We use the term "computational statistics" somewhat more broadly to include not only the methods of statistical computing, but also statistical methods that are computationally intensive. Thus, to some extent, "computational statistics" refers to a large class of modern statistical methods. Computational statistics is grounded in mathematical statistics, statistical computing, and applied statistics. While we distinguish "computational statistics" from "statistical computing", the emergence of the field of computational statistics was coincidental with that of statistical computing, and would not have been possible without the developments in statistical computing.

One of the most significant results of the developments in statistical computing during the past few decades has been the statistical software package. There are several of these, but a relatively small number that are in widespread use. While referees and editors of scholarly journals determine what statistical theory and methods are *published*, the developers of the major statistical software packages determine what statistical methods are *used*. Computer programs have become necessary for statistical analysis. The specific methods of a statistical analysis are often determined by the available software. This, of course, is not a desirable situation, but, ideally, the two-way feedback between statistical theory and statistical computing dimishes the effect over time.

The importance of computing in statistics is also indicated by the fact that there are at least ten major journals with titles that contain some variants of both "computing" and "statistics". The journals in the mainstream of statistics without "computing" in their titles also have a large proportion of articles in the fields of statistical computing and computational statistics. This is because, to a large extent, recent developments in statistics and in the computational sciences have gone hand in hand. There are also two well-known learned societies with a primary focus in statistical computing: the **International Association for Statistical Computing** (IASC), which is an affiliated society of the International Statistical Institute (ISI), and the **Statistical Computing Section** of the American Statistical Association (ASA). There are also a number of other associations focused on statistical computing and computational statistics, such as the Statistical Computing Section of the Royal Statistical Society (RSS), and the **Japanese Society of Computational Statistics** (JSCS).

Developments in computing and the changing role of computations in statistical work have had significant effects on the curricula of statistical education programs

both at the graduate and undergraduate levels. Training in statistical computing is a major component in some academic programs in statistics (see Gentle 2004; Lange 2004; Monahan 2004; Nolan and Temple Lang 2010). In all academic programs, some amount of computing instruction is necessary if the student is expected to work as a statistician. The extent and the manner of integration of computing into an academic statistics program, of course, change with the developments in computing hardware and software and advances in computational statistics.

We mentioned above the two-way feedback between statistical theory and statistical computing. There is also an important two-way feedback between applications and statistical computing, just as there has always been between applications and any aspect of statistics. Although data scientists seek commonalities among methods of data analysis, different areas of application often bring slightly different problems for the data analyst to address. In recent years, an area called "data mining" or "knowledge mining" has received much attention. The techniques used in data mining are generally the methods of exploratory data analysis, of clustering, and of statistical learning, applied to very large and, perhaps, diverse datasets. Scientists and corporate managers alike have adopted data mining as a central aspect of their work. Specific areas of application also present interesting problems to the computational statistician. Financial applications, particularly risk management and derivative pricing, have fostered advances in computational statistics. Biological applications, such as bioinformatics, microarray analysis, and computational biology, are fostering increasing levels of interaction with computational statistics.

The hallmarks of computational statistics are the use of more complicated models, larger datasets with both more observations and more variables, unstructured and heterogeneous datasets, heavy use of visualization, and often extensive simulations.

1.2 The Emergence of a Field of Computational Statistics

Statistical computing is truly a multidisciplinary field and the diverse problems have created a yeasty atmosphere for research and development. This has been the case from the beginning. The roles of statistical laboratories and the applications that drove early developments in statistical computing are surveyed by Grier (1999). As digital computers began to be used, the field of statistical computing came to embrace not only numerical methods but also a variety of topics from computer science.

The development of the field of statistical computing was quite fragmented, with advances coming from many directions – some by persons with direct interest and expertise in computations, and others by persons whose research interests were in the applications, but who needed to solve a computational problem. Through the 1950s the major facts relevant to statistical computing were scattered through a variety of journal articles and technical reports. Many results were incorporated into computer programs by their authors and never appeared in the open literature. Some

persons who contributed to the development of the field of statistical computing were not aware of the work that was beginning to put numerical analysis on a sound footing. This hampered advances in the field.

1.2.1 Early Developments in Statistical Computing

An early book that assembled much of the extant information on digital computations in the important area of linear computations was by Dwyer (1951). In the same year, Von Neumann's NBS publication (Von Neumann 1951) described techniques of random number generation and applications in Monte Carlo. At the time of these publications, however, access to digital computers was not widespread. Dwyer (1951) was also influential in regression computations performed on calculators. Some techniques, such as use of "machine formulas", persisted into the age of digital computers.

Developments in statistical computing intensified in the 1960s, as access to digital computers became more widespread. Grier (1991) describes some of the effects on statistical practice by the introduction of digital computers, and how statistical applications motivated software developments. The problems of rounding errors in digital computations were discussed very carefully in a pioneering book by Wilkinson (1963). A number of books on numerical analysis using digital computers were beginning to appear. The techniques of random number generation and Monte Carlo were described by Hammersley and Handscomb (1964). In 1967 the first book specifically on statistical computing appeared, Hemmerle (1967).

1.2.2 Early Conferences and Formation of Learned Societies

The 1960s also saw the beginnings of conferences on statistical computing and sections on statistical computing within the major statistical societies. The Royal Statistical Society sponsored a conference on statistical computing in December 1966. The papers from this conference were later published in the RSS's *Applied Statistics* journal. The conference led directly to the formation of a Working Party on Statistical Computing within the Royal Statistical Society. The first Symposium on the **Interface of Computer Science and Statistics** was held February 1, 1967. This conference has continued as an annual event with only a few exceptions since that time (see Billard and Gentle 1993; Goodman 1993; Wegman 1993). The attendance at the Interface Symposia initially grew rapidly year by year and peaked at over 600 in 1979. In recent years the attendance has been slightly under 300. The proceedings of the Symposium on the Interface have been an important repository of developments in statistical computing. In April, 1969, an important conference on statistical computing was held at the University of Wisconsin. The papers presented at that conference were published in a book edited by Milton and Nelder (1969),

which helped to make statisticians aware of the useful developments in computing and of their relevance to the work of applied statisticians.

In the 1970s two more important societies devoted to statistical computing were formed. The **Statistical Computing Section** of the ASA was formed in 1971 (see Chambers and Ryan 1990). The Statistical Computing Section organizes sessions at the annual meetings of the ASA, and publishes proceedings of those sessions. The **International Association for Statistical Computing** (IASC) was founded in 1977 as a Section of ISI. In the meantime, the first of the biennial **COMPSTAT Conferences** on computational statistics was held in Vienna in 1974. Much later, regional sections of the IASC were formed, one in Europe and one in Asia. The European Regional Section of the IASC is now responsible for the organization of the COMPSTAT conferences.

Also, beginning in the late 1960s and early 1970s, most major academic programs in statistics offered one or more courses in statistical computing. More importantly, perhaps, instruction in computational techniques has permeated many of the standard courses in applied statistics.

As mentioned above, there are several journals whose titles include some variants of both "computing" and "statistics". The first of these, the *Journal of Statistical Computation and Simulation*, was begun in 1972. There are dozens of journals in numerical analysis and in areas such as "computational physics", "computational biology", and so on, that publish articles relevant to the fields of statistical computing and computational statistics.

By 1980 the field of statistical computing, or computational statistics, was well-established as a distinct scientific subdiscipline. Since then, there have been regular conferences in the field, there are scholarly societies devoted to the area, there are several technical journals in the field, and courses in the field are regularly offered in universities.

1.2.3 The PC

The 1980s was a period of great change in statistical computing. The personal computer brought computing capabilities to almost everyone. With the PC came a change not only in the number of participants in statistical computing, but, equally important, completely different attitudes toward computing emerged. Formerly, to do computing required an account on a mainframe computer. It required laboriously entering arcane computer commands onto punched cards, taking these cards to a card reader, and waiting several minutes or perhaps a few hours for some output – which, quite often, was only a page stating that there was an error somewhere in the program. With a personal computer for the exclusive use of the statistician, there was no incremental costs for running programs. The interaction was personal, and generally much faster than with a mainframe. The software for PCs was friendlier and easier to use. As might be expected with many non-experts writing software, however, the general quality of software probably went down.

The democratization of computing resulted in rapid growth in the field, and rapid growth in software for statistical computing. It also contributed to the changing paradigm of the data sciences.

1.2.4 The Cross Currents of Computational Statistics

Computational statistics of course is more closely related to statistics than to any other discipline, and computationally-intensive methods are becoming more commonly used in various areas of application of statistics. Developments in other areas, such as computer science and numerical analysis, are also often directly relevant to computational statistics, and the research worker in this field must scan a wide range of literature.

Numerical methods are often developed in an ad hoc way, and may be reported in the literature of any of a variety of disciplines. Other developments important for statistical computing may also be reported in a wide range of journals that statisticians are unlikely to read. Keeping abreast of relevant developments in statistical computing is difficult not only because of the diversity of the literature, but also because of the interrelationships between statistical computing and computer hardware and software.

An example of an area in computational statistics in which significant developments are often made by researchers in other fields is Monte Carlo simulation. This technique is widely used in all areas of science, and researchers in various areas often contribute to the development of the science and art of Monte Carlo simulation. Almost any of the methods of Monte Carlo, including random number generation, are important in computational statistics.

1.2.5 Reproducible Research

Reproducibility in the sense of replication within experimental error has always been a touchstone of science. In recent years, however, the term "**reproducible research**" (RR), or sometimes "reproducible analysis", has taken on a stronger meaning. The standards for RR include provision of computer codes (preferably in source) and/or data that would allow the reader to replicate the reported results (see Baggerly and Berry 2011).

Many journals enforce these requirements, or at least facilitate the provisions. The *Journal of American Statistical Association*, for example, encourages authors to provide code and/or data, as well as other supporting material. This additional material is linked with an electronic version of the article at the journal's web site.

Many articles in computational statistics are written in LaTeX and the computations are done in R. The R code, together with any input data, allows the reader to perform the same computations for simulations and analyses that yielded the results

reported in the accompanying text. The Sweave package facilitates the incorporation of code with text in the same file (see Leisch 2002). Instructions for obtaining Sweave as well as the current user manual can be obtained at http://www.statistik.lmu.de/~leisch/Sweave/Sweave-manual.pdf

1.2.6 Literature

Some of the major periodicals in statistical computing and computational statistics are listed below. Some of these journals and proceedings are refereed rather rigorously, some refereed less so, and some are not refereed. Although most of these serials are published in hardcopy form, most are also available electronically.

- *ACM Transactions on Mathematical Software*, published quarterly by the ACM (Association for Computing Machinery), includes algorithms in Fortran and C. Most of the algorithms are available through `netlib`. The ACM collection of algorithms is sometimes called *CALGO*.
  www.acm.org/toms/
- *ACM Transactions on Modeling and Computer Simulation*, published quarterly by the ACM.
  www.acm.org/tomacs/
- *Applied Statistics*, published quarterly by the Royal Statistical Society. (Until 1998, it included algorithms in Fortran. Some of these algorithms, with corrections, were collected by Griffiths and Hill, 1985. Most of the algorithms are available through `statlib` at Carnegie Mellon University.)
  www.rss.org.uk/publications/
- *Communications in Statistics – Simulation and Computation*, published quarterly by Marcel Dekker. (Until 1996, it included algorithms in Fortran. Until 1982, this journal was designated as *Series B*.)
  www.dekker.com/servlet/product/productid/SAC/
- *Computational Statistics* published quarterly by Physica-Verlag (formerly called *Computational Statistics Quarterly*).
  comst.wiwi.hu-berlin.de/
- *Computational Statistics. Proceedings of the* xx^{th} *Symposium on Computational Statistics* (COMPSTAT), published biennially by Physica-Verlag/Springer.
- *Computational Statistics & Data Analysis*, published by Elsevier Science. There are twelve issues per year. (This is also the official journal of the International Association for Statistical Computing and as such incorporates the *Statistical Software Newsletter*.)
  www.cbs.nl/isi/csda.htm
- *Computing Science and Statistics*. This is an annual publication containing papers presented at the Interface Symposium. Until 1992, these proceedings were named *Computer Science and Statistics: Proceedings of the* xx^{th} *Symposium on the Interface*. (The 24th symposium was held in 1992.) In 1997, Volume 29 was

published in two issues: Number 1, which contains the papers of the regular Interface Symposium; and Number 2, which contains papers from another conference. The two numbers are not sequentially paginated. Since 1999, the proceedings have been published only in CD-ROM form, by the Interface Foundation of North America.
www.galaxy.gmu.edu/stats/IFNA.html

- *Journal of Computational and Graphical Statistics*, published quarterly as a joint publication of ASA, the Institute of Mathematical Statistics, and the Interface Foundation of North America.
  www.amstat.org/publications/jcgs/
- *Journal of the Japanese Society of Computational Statistics*, published once a year by JSCS.
  www.jscs.or.jp/oubun/indexE.html
- *Journal of Statistical Computation and Simulation*, published in twelve issues per year by Taylor & Francis.
  www.tandf.co.uk/journals/titles/00949655.asp
- *Journal of Statistical Software*, a free on-line journal that publishes articles, book reviews, code snippets, and software reviews.
  www.jstatsoft.org/
- *Proceedings of the Statistical Computing Section*, published annually by ASA.
  www.amstat.org/publications/
- *SIAM Journal on Scientific Computing*, published bimonthly by SIAM. This journal was formerly *SIAM Journal on Scientific and Statistical Computing*.
  www.siam.org/journals/sisc/sisc.htm
- *Statistical Computing & Graphics Newsletter*, published quarterly by the Statistical Computing and the Statistical Graphics Sections of ASA.
  www.statcomputing.org/
- *Statistics and Computing*, published quarterly by Chapman & Hall.

In addition to literature and learned societies in the traditional forms, an important source of communication and a repository of information are computer databases and forums. In some cases, the databases duplicate what is available in some other form, but often the material and the communications facilities provided by the computer are not available elsewhere.

1.3 This Handbook

The purpose of this handbook is the same as that of the first edition of *Concepts and Fundamentals*. It is to provide a survey of the basic concepts of computational statistics. A glance at the table of contents reveals a wide range of articles written by experts in various subfields of computational statistics. The articles are generally expository, taking the reader from the basic concepts to the current research trends.

The emphasis throughout, however, is on the concepts and fundamentals. Most chapters have been revised to provide up-to-date references to the relevant literature.

We have retained the organization of the main body in three parts. Part II on "statistical computing" addresses the computational methodology; Part III on "statistical methodology" covers techniques of applied statistics that are computer-intensive, or otherwise that make use of the computer as a tool of discovery, rather than as just a large and fast calculator; and, finally, Part IV describes a number of application areas in which computational statistics plays a major role.

1.3.1 Summary and Overview; Part II: Statistical Computing

Statistical computing is in the interface of numerical analysis, computer science, and statistics. This interface includes computer arithmetic, algorithms, database methodology, languages and other aspects of the user interface, and computer graphics.

For statistical numerical analysis, it is important to understand how the computer does arithmetic, and more importantly what the implications are for statistical (or other) computations. In addition to understanding of the underlying arithmetic operations, the basic principles of numerical algorithms, such as divide and conquer, must be in the working knowledge of persons writing numerical software for statistical applications. Although many statisticians do not need to know the details, it is important that all statisticians understand the implications of computations within a system of numbers and operators that is not the same system that we are accustomed to in mathematics. Anyone developing computer algorithms, no matter how trivial the algorithm may appear, must understand the details of the computer system of numbers and operators.

One of the important uses of computers in statistics, and one that is central to computational statistics, is the simulation of random processes. This is a theme of several chapters of this handbook, but in Part II, the basic numerical methods relevant to simulation are discussed. These include the basics of random number generation, including assessing the quality of random number generators, and simulation of random samples from various distributions, as well as the class of methods called Markov chain Monte Carlo. Statistical methods using simulated samples are discussed further in Part III.

Some chapters of Part II address specific numerical methods, such as methods for linear algebraic computations, for optimization, and for transforms. Separate chapters in Part II discuss two specific areas of optimization, the EM algorithm and its variations, and stochastic optimization. Another chapter describes transforms, such as the well-known Fourier and wavelet transforms, that effectively restructure a problem by changing the domain are important statistical functionals.

Other chapters of Part II focus on efficient usage of computing resources. Specific topics include parallel computing, database management methodology, issues relating to the user interface, and even paradigms, such as an object orientation, for software development.

Statistical graphics, especially interactive and dynamic graphics, play an increasingly prominent role in data analysis. Two chapters of Part II are devoted to this important area.

1.3.2 Summary and Overview; Part III: Statistical Methodology

Part III covers several aspects of computational statistics. In this part the emphasis is on the statistical methodology that is enabled by computing. Computers are useful in all aspects of statistical data analysis, of course, but in Part III, and generally in computational statistics, we focus on statistical methods that are computationally intensive. Although a theoretical justification of these methods often depends on asymptotic theory, in particular, on the asymptotics of the empirical cumulative distribution function, asymptotic inference is generally replaced by **computational inference**.

The first few chapters of this part deal directly with techniques of computational inference; that is, the use of cross validation, resampling, and simulation of data-generating processes to make decisions and to assign a level of confidence to the decisions. Selection of a model implies consideration of more than one model. As we suggested above, this is one of the hallmarks of computational statistics: looking at data through a variety of models. Cross validation and its generalizations and resampling are important techniques for addressing the problems. Resampling methods also have much wider applicability in statistics, from estimating variances and setting confidence regions to larger problems in statistical data analysis. Computational inference depends on simulation of data-generating processes. Any such simulation is an *experiment*, and in Part III, principles for design and analysis of experiments using computer models are discussed.

Estimation of a multivariate probability density function is also addressed in Part III. This area is fundamental in statistics, and it utilizes several of the standard techniques of computational statistics, such as cross validation and visualization methods.

The next few chapters of Part III address important issues for discovery and analysis of relationships among variables. One class of models are asymmetric, that is, models for the effects of a given set of variables (“independent variables") on another variable or set of variables. Smoothing methods for these models, which include use of kernels, splines, and orthogonal series, are generally nonparametric or semiparametric. Two important types of parametric asymmetric models discussed in Part III are generalized linear models and nonlinear regression models. In any models that explore the relationships among variables, it is often desirable to reduce the effective dimensionality of a problem. All of these chapters on using models of variate relationships to analyze data emphasize the computational aspects.

One area in which computational inference has come to play a major role is in Bayesian analysis. Computational methods have enabled a Bayesian approach in

practical applications, because no longer is this approach limited to simple problems or conjugate priors.

Survival analysis, with applications in both medicine and product reliability, has become more important in recent years. Computational methods for analyzing models used in survival analysis are discussed in Part III.

The final chapters of Part III address an exciting area of computational statistics. The general area may be called "data mining", although this term has a rather anachronistic flavor because of the hype of the mid-1990s. Other terms such as "knowledge mining" or "knowledge discovery in databases" ("KDD") are also used. To emphasize the roots in artificial intelligence, which is a somewhat discredited area, the term "computational intelligence" is also used. This is an area in which machine learning from computer science and statistical learning have merged.

1.3.3 Summary and Overview; Part IV: Statistical Methodology

Many areas of applications can only be addressed effectively using computationally-intensive statistical methods. This is often because the input datasets are so large, but it may also be because the problem requires consideration of a large number of possible alternatives. In Part IV, there are separate chapters on some areas of applications of computational statistics. One area is finance and economics, in which heavy-tailed distributions or models with nonconstant variance are important.

Human biology has become one of the most important areas of application, and many computationally-intensive statistical methods have been developed, refined, and brought to bear on problems in this area. Two important question involve the geometrical structure of protein molecules and the functions of the various areas in the brain. While much is known about the order of the components of the molecules, the three-dimensional structure for most important protein molecules is not known, and the tools for discovery of this structure need extensive development. Understanding the functions of different areas in the brain will allow more effective treatment of diseased or injured areas and the resumption of more normal activities by patients with neurological disorders.

Another important area of application of computational statistics is computer network intrusion detection. Because of the importance of computer networks around the world, and because of their vulnerability to unauthorized or malicious intrusion, detection has become one of the most important – and interesting – areas for data mining.

The articles in this handbook cover the important subareas of computational statistics and give some flavor of the wide range of applications. While the articles emphasize the basic concepts and fundamentals of computational statistics, they provide the reader with tools and suggestions for current research topics. The reader may turn to a specific chapter for background reading and references on a

particular topic of interest, but we also suggest that the reader browse and ultimately peruse articles on unfamiliar topics. Many surprising and interesting tidbits will be discovered!

1.3.4 Other Handbooks in Computational Statistics

This handbook on concepts and fundamentals sets the stage for future handbooks that go more deeply into the various subfields of computational statistics. These handbooks will each be organized around either a specific class of theory and methods, or else around a specific area of application.

The development of the field of computational statistics has been rather fragmented. We hope that the articles in this handbook series can provide a more unified framework for the field.

In the years since the publication of the first volume in the series of Handbooks in Computational Statistics, which covered general concepts and methods, three other volumes have appeared. These are on somewhat more narrow topics within the field of computational statistics: data visualization, partial least squares, and computational finance.

References

Baggerly, K.A., Berry, D.A.: Reproducible research, *AmStatNews*, January, http://magazine.amstat.org/blog/2011/01/01/scipolicyjan11/. (2011)
Billard, L., Gentle, J.E.: The middle years of the interface. Comput. Science Stat. **25**, 19–26 (1993)
Chambers, J.M., Ryan, B.F.: The ASA statistical computing section. Am. Stat. **44**(2), 87–89 (1990)
Dwyer, P.S.: Linear Computations. Wiley, New York (1951)
Gentle, J.E.: Courses in statistical computing and computational statistics. Am. Stat. **58**, 2–5 (2004)
Goodman, A.: Interface insights: From birth into the next century. Comput. Science Stat. **25**, 14–18 (1993)
Grier, D.A.: Statistics and the introduction of digital computers. Chance **4**(3), 30–36 (1991)
Grier, D.A.: Statistical laboratories and the origins of statistical computing. Chance **4**(2), 14–20 (1999)
Hammersley, J.M., Handscomb, D.C.: Monte Carlo Methods, Methuen & Company, London (1964)
Hemmerle, W.J.: Statistical Computations on a Digital Computer. Blaisdell, Waltham, Massachusetts (1967)
Lange, K.: Computational Statistics and Optimization Theory at UCLA. Am. Stat. 58:9–11 (2004)
Leisch, F.: Sweave: Dynamic Generation of Statistical Reports Using Literate Data Analysis. In Compstat 2002 – Proceedings in Computational Statistics, edited by W. Härdle and B. Rönz, 575–580 (2002)
Milton, R., Nelder, J. (ed.): Statistical Computation, Academic Press, New York (1969)
Monahan, J.: Teaching statistical computing at NC state. Am. Stat. **58**, 6–8 (2004)
Nolan, D., Temple Lang, D.: Computing in the Statistics Curriculum. Am. Stat. **64**, 97–107 (2010)

Von Neumann, J.: Various Techniques Used in Connection with Random Digits, National Bureau of Standards Symposium, NBS Applied Mathematics Series 12, National Bureau of Standards (now National Institute of Standards and Technology), Washington, DC (1951)
Wegman, E.J.: History of the Interface since 1987: The corporate era. Comput. Science Stat. **25**, 27–32 (1993)
Wilkinson, J.H.: Rounding Errors in Algebraic Processes, Prentice-Hall, Englewood Cliffs, NJ (1963)

Part II
Statistical Computing

Chapter 2
Basic Computational Algorithms

John F. Monahan

2.1 Computer Arithmetic

Numbers are the lifeblood of statistics, and computational statistics relies heavily on how numbers are represented and manipulated on a computer. Computer hardware and statistical software handle numbers well, and the methodology of computer arithmetic is rarely a concern. However, whenever we push hardware and software to their limits with difficult problems, we can see signs of the mechanics of floating point arithmetic around the frayed edges. To work on difficult problems with confidence and explore the frontiers of statistical methods and software, we need to have a sound understanding of the foundations of computer arithmetic. We need to know how arithmetic works and why things are designed the way they are.

As scientific computation began to rely heavily on computers, a monumental decision was made during the 1960s to change from base ten arithmetic to base two. Humans had been doing base ten arithmetic for only a few hundred years, during which time great advances were possible in science in a short period of time. Consequently, the resistance to this change was strong and understandable. The motivation behind the change to base two arithmetic is merely that it is so very easy to do addition (and subtraction) and multiplication in base two arithmetic. The steps are easy enough that a machine can be designed – wire a board of relays – or design a silicon chip – to do base two arithmetic. Base ten arithmetic is comparatively quite difficult, as its recent mathematical creation would suggest. However two big problems arise in changing from base ten to base two: (1) we need to constantly convert numbers written in base ten by humans to base two number system and then back again to base ten for humans to read the results, and (2) we need to understand the limits of arithmetic in a different number system.

J.F. Monahan (✉)
Department of Statistics, North Carolina State University, Raleigh, NC, USA
e-mail: monahan@ncsu.edu

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_2,

2.1.1 Integer Arithmetic

Computers use two basic ways of writing numbers: fixed point (for integers) and floating point (for real numbers). Numbers are written on a computer following base two positional notation. The *positional number system* is a convention for expressing a number as a list of integers (digits), representing a number x in base B by a list of digits $a_m, a_{m-1}, \ldots, a_1, a_0$ whose mathematical meaning is

$$x = a_{m-1}B^{m-1} + \ldots + a_2B^2 + a_1B + a_0 \tag{2.1}$$

where the digits a_j are integers in $\{0, \ldots, B-1\}$. We are accustomed to what is known in the West as the Arabic numbers, $0, 1, 2, \ldots, 9$ representing those digits for writing for humans to read. For base two arithmetic, only two digits are needed $\{0, 1\}$. For base sixteen, although often viewed as just a collection of four binary digits ($1\,byte = 4\,bits$), the Arabic numbers are augmented with letters, as $\{0, 1, 2, \ldots, 9, a, b, c, d, e, f\}$, so that $f_{sixteen} = 15_{ten}$.

The system based on (2.1), known as *fixed point arithmetic*, is useful for writing integers. The choice of $m = 32$ dominates current computer hardware, although smaller ($m = 16$) choices are available via software and larger ($m = 48$) hardware had been common in high performance computing. Recent advances in computer architecture may soon lead to the standard changing to $m = 64$. While the writing of a number in base two requires only the listing of its binary digits, a convention is necessary for expression of negative numbers. The survivor of many years of intellectual competition is the *two's complement* convention. Here the first (leftmost) bit is reserved for the sign, using the convention that 0 means positive and 1 means negative. Negative numbers are written by complementing each bit (replace 1 with 0, 0 with 1) and adding one to the result. For $m = 16$ (easier to display), this means that 22_{ten} and its negative are written as

$$(0\ 001\ 0110) = 22_{\text{ten}}$$

and

$$(1\ 110\ 1010) = -22_{\text{ten}}\ .$$

Following the two's complement convention with m bits, the smallest (negative) number that can be written is -2^{m-1} and the largest positive number is $2^{m-1} - 1$; zero has a unique representation of $(0\ 000\ \cdots\ 0000)$. Basic arithmetic (addition and multiplication) using two's complement is easy to code, essentially taking the form of mod 2^{m-1} arithmetic, with special tools for overflow and sign changes. See, for example, Knuth (1997) for history and details, as well as algorithms for base conversions.

The great advantage of fixed point (integer) arithmetic is that it is so very fast. For many applications, integer arithmetic suffices, and most nonscientific computer software only uses fixed point arithmetic. Its second advantage is that it does not

suffer from the rounding error inherent in its competitor, floating point arithmetic, whose discussion follows.

2.1.2 Floating Point Arithmetic

To handle a larger subset of the real numbers, the positional notation system includes an exponent to express the location of the radix point (generalization of the decimal point), so that the usual format is a triple (sign, exponent, fraction) to represent a number as

$$x = (-1)^{\text{sign}} B^{\text{exponent}} \left(a_1 B^{-1} + a_2 B^{-2} + \ldots + a_d B^{-d}\right) , \tag{2.2}$$

where the fraction is expressed by its list of base B digits $0.a_1 a_2 a_3 \ldots a_d$. To preserve as much information as possible with the limited d digits to represent the fraction, *normalization* is usually enforced, that is, the leading/most significant digit a_1 is nonzero – except for the special case $x = 0$. The mathematical curiosity of an infinite series expansion of a number has no place here where only d digits are available. Moreover, a critical issue is what to do when only d digits are available. *Rounding* to the nearest number is preferred to the alternative *chopping*; in the case of representing $\pi = 3.14159265\ldots$ to $d = 5$ decimal ($B =$ ten) digits leads to the more accurate $(+, +1, 0.31416)$ in the case of rounding, rather than $(+, +1, 0.31415)$ for the chopping alternative. Notice that normalization and the use of this positional notation reflects a goal of preserving relative accuracy, or reducing the relative error in the approximation. The expression of a real number x in floating point arithmetic can be expressed mathematically in terms of a function $fl : R \to \mathcal{F}$ where $\mathcal{F}$ is the set of numbers that can be represented using this notation, the set of floating point numbers. The relative accuracy of this rounding operation can be expressed as

$$fl(x) = (1 + u)x , \tag{2.3}$$

where $|u| \leq U$ where U is known as the *machine unit*. Seen in terms of the relative error of $fl(x)$ in approximating x, the expression above can be rewritten as

$$|x - fl(x)|/|x| \leq U \quad \text{for} \quad x \neq 0 .$$

For base B arithmetic with d digits and chopping, $U = B^{1-d}$; rounding reduces U by a factor of 2.

An important conceptual leap is the understanding that most numbers are represented only approximately in floating point arithmetic. This extends beyond the usual irrational numbers such as π or e that cannot be represented with a finite number of digits. A novice user may enter a familiar assignment such as $x = 8.6$ and, observing that the computer prints out 8.6000004, may consider this an error.

When the "8.6" was entered, the computer had to first parse the text "8.6" and recognize the decimal point and arabic numbers as a representation, for humans, of a real number with the value $8 + 6 \times 10^{-1}$. The second step is to convert this real number to a base two floating point number – approximating this base ten number with the closest base two number – this is the function $fl(\cdot)$. Just as $1/3$ produces the repeating decimal $0.33333\ldots$ in base 10, the number 8.6 produces a repeating binary representation $1000.100110011\ldots_{two}$, and is chopped or rounded to the nearest floating point number $fl(8.6)$. Later, in printing this same number out, a second conversion produces the closest base 10 number to $fl(8.6)$ with few digits; in this case 8.6000004, not an error at all. Common practice is to employ numbers that are integers divided by powers of two, since they are exactly represented. For example, distributing 1,024 equally spaced points makes more sense than the usual 1,000, since $j/1024$ can be exactly represented for any integer j.

A breakthrough in hardware for scientific computing came with the adoption and implementation of the IEEE 754 binary floating point arithmetic standard, which has standards for two levels of precision, *single precision* and *double precision* (IEEE 1985). The single precision standard uses 32 bits to represent a number: a single bit for the sign, 8 bits for the exponent and 23 bits for the fraction. The double precision standard requires 64 bits, using 3 more bits for the exponent and adds 29 to the fraction for a total of 52. Since the leading digit of a normalized number is nonzero, in base two the leading digit must be one. As a result, the floating point form (2.2) above takes a slightly modified form:

$$x = (-1)^{\text{sign}} B^{\text{exponent}-\text{excess}} \left(1 + a_1 B^{-1} + a_2 B^{-2} + \ldots + a_d B^{-d}\right) \tag{2.4}$$

as the fraction is expressed by its list of binary digits $1.a_1a_2a_3\ldots a_d$. As a result, while only 23 bits are stored, it works as if one more bit were stored. The exponent using 8 bits can range from 0 to 255; however, using an excess of 127, the range of the difference $(exponent - excess)$ goes from -126 to 127. The finite number of bits available for storing numbers means that the set of floating point numbers $\mathcal{F}$ is a finite, discrete set. Although well-ordered, it does have a largest number, smallest number, and smallest positive number. As a result, this IEEE Standard expresses positive numbers from approximately 1.4×10^{-45} to 3.4×10^{38} with a machine unit $U = 2^{-24} \approx 10^{-7}$ using only 31 bits. The remaining 32nd bit is reserved for the sign. Double precision expands the range to roughly $10^{\pm 300}$ with $U = 2^{-53} \approx 10^{-16}$, so the number of accurate digits is more than doubled.

The two extreme values of the exponent are employed for special features. At the high end, the case exponent $= 255$ signals two infinities ($\pm\infty$) with the largest possible fraction. These values arise as the result of an *overflow* operation. The most common causes are adding or multiplying two very large numbers, or from a function call that produces a result that is larger than any floating point number. For example, the value of $\exp(x)$ is larger than any finite number in $\mathcal{F}$ for $x > 88.73$ in single precision. Before adoption of the standard, $\exp(89.9)$ would cause the program to cease operation due to this "exception". Including $\pm\infty$ as members of $\mathcal{F}$ permits the computations to continue, since a sensible result is now available.

As a result, further computations involving the value $\pm\infty$ can proceed naturally, such as $1/\infty = 0$. Again using the exponent $= 255$, but with any other fraction represents *not-a-number*, usually written as "NaN", and used to express the result of *invalid operations*, such as $0/0$, $\infty - \infty$, $0 \times \infty$, and square roots of negative numbers. For statistical purposes, another important use of NaN is to designate missing values in data. The use of infinities and NaN permit continued execution in the case of anomalous arithmetic operations, instead of causing computation to cease when such anomalies occur. The other extreme exponent $= 0$ signals a denormalized number with the net exponent of -126 and an unnormalized fraction, with the representation following (2.2), rather than the usual (2.4) with the unstated and unstored 1. The *denormalized* numbers further expand the available numbers in $\mathcal{F}$, and permit a *soft underflow*. Underflow, in contrast to overflow, arises when the result of an arithmetic operation is smaller in magnitude than the smallest representable positive number, usually caused by multiplying two small numbers together. These denormalized numbers begin approximately 10^{-38} near the reciprocal of the largest positive number. The denormalized numbers provide even smaller numbers, down to 10^{-45}. Below that, the next number in $\mathcal{F}$ is the floating point zero: the smallest exponent and zero fraction – all bits zero.

Most statistical software employs only double precision arithmetic, and some users become familiar with apparent aberrant behavior such as a sum of residuals of 10^{-16} instead of zero. While many areas of science function quite well using single precision, some problems, especially nonlinear optimization, nevertheless require double precision. The use of single precision requires a sound understand of rounding error. However, the same rounding effects remain in double precision, but because their effects are so often obscured from view, double precision may promote a naive view that computers are perfectly accurate.

The machine unit expresses a relative accuracy in storing a real number as a floating point number. Another similar quantity, the *machine epsilon*, denoted by ϵ_m, is defined as the smallest positive number that, when added to one, gives a result that is different from one. Mathematically, this can be written as

$$fl(1 + x) = 1 \text{ for } 0 < x < \epsilon_m \, . \tag{2.5}$$

Due to the limited precision in floating point arithmetic, adding a number that is much smaller in magnitude than the machine epsilon will not change the result. For example, in single precision, the closest floating point number to $1 + 2^{-26}$ is 1. Typically, both the machine unit and machine epsilon are nearly the same size, and these terms are often used interchangeably without grave consequences.

2.1.3 Cancellation

Often one of the more surprising aspects of floating point arithmetic is that some of the more familiar laws of algebra are occasionally violated: in particular, the

associative and distributive laws. While most occurrences are just disconcerting to those unfamiliar to computer arithmetic, one serious concern is cancellation. For a simple example, consider the case of base ten arithmetic with $d = 6$ digits, and take $x = 123.456$ and $y = 123.332$, and note that both x and y may have been rounded, perhaps x was 123.456478 or 123.456000 or 123.455998. Now x would be stored as $(+, 3, 0.123456)$ and y would be written as $(+, 3, 0.123332)$, and when these two numbers are subtracted, we have the unnormalized difference $(+, 3, 0.000124)$. Normalization would lead to $(+, 0, .124???)$ where merely "?" represents that some digits need to take their place. The simplistic option is to put zeros, but 0.124478 is just as good an estimate of the true difference between x and y as 0.124000, or 0.123998, for that matter. The problem with cancellation is that the relative accuracy that floating point arithmetic aims to protect has been corrupted by the loss of the leading significant digits. Instead of a small error in the sixth digit, we now have that error in the third digit; the relative error has effectively been magnified by a factor of 1,000 due to the cancellation of the first 3 digits.

The best way to deal with the potential problem caused by catastrophic cancellation is to avoid them. In many cases, the cancellation may be avoided by reworking the computations analytically to handle the cancellation:

$$1 - (1 - 2t)^{-1} = \frac{1 - 2t - 1}{1 - 2t} = \frac{-2t}{1 - 2t} \; .$$

In this case, there is significant cancellation when t is small, and catastrophic cancellation whenever t drops below the machine epsilon. Using six digit decimal arithmetic to illustrate, at $t = 0.001$, the left hand expression, $1 - (1 - 2t)^{-1}$, gives

$$1.00000 - 1.00200 = 0.200000 \times 10^{-2}$$

while the right hand expression, $-2t/(1 - 2t)$, gives

$$0.200401 \times 10^{-2} \; ,$$

the correct (rounded) result. The relative error in using the left hand expression is an unacceptable 0.002. At $t = 10^{-7}$, the left hand expression leads to a complete cancellation yielding zero and a relative error of one. Just a little algebra here avoids the most of the effect of cancellation. When the expressions involve functions, cases where cancellation occurs can often be handled by approximations. In the case of $1 - \mathrm{e}^{-t}$, serious cancellation will occur whenever t is very small. The cancellation can be avoided for this case by using a power series expansion:

$$1 - \mathrm{e}^{-t} = 1 - \left(1 - t + t^2/2 - \ldots\right) \approx t - t^2/2 = t\,(1 - t/2) \; .$$

When $t = 0.0001$, the expression $1 - \mathrm{e}^{-t}$ leads to the steps

$$1.00000 - 0.999900 = 0.100000 \times 10^{-4} \; ,$$

while the approximation gives

$$(0.0001)(0.999950) = 0.999950 \times 10^{-4}$$

which properly approximates the result to six decimal digits. At $t = 10^{-5}$ and 10^{-6}, similar results occur, with complete cancellation at 10^{-7}. Often the approximation will be accurate just when cancellation must be avoided.

One application where rounding error must be understood and cancellation cannot be avoided is numerical differentiation, where calls to a function are used to approximate a derivative from a first difference:

$$f'(x) \approx [\, f(x+h) - f(x)] \,/\, h \;. \tag{2.6}$$

Mathematically, the accuracy of this approximation is improved by taking h very small; following a quadratic Taylor's approximation, we can estimate the error as

$$[\, f(x+h) - f(x)] \,/\, h \approx f'(x) + \frac{1}{2} h f''(x) \;.$$

However, when the function calls $f(x)$ and $f(x+h)$ are available only to limited precision – a relative error of ϵ_m, taking h smaller leads to more cancellation. The cancellation appears as a random rounding error in the numerator of (2.6) which becomes magnified by dividing by a small h. Taking h larger incurs more bias from the approximation; taking h smaller incurs larger variance from the rounding error. Prudence dictates balancing bias and variance. Dennis and Schnabel (1983) recommend using $h \approx \epsilon_m^{1/2}$ for first differences, but see also Bodily (2002).

The second approach for avoiding the effects of cancellation is to develop different methods. A common cancellation problem in statistics arises from using the formula

$$\sum_{i=1}^{n} y_i^2 - n\overline{y}^2 \tag{2.7}$$

for computing the sum of squares around the mean. Cancellation can be avoided by following the more familiar two-pass method

$$\sum_{i=1}^{n} (y_i - \overline{y})^2 \tag{2.8}$$

but this algorithm requires all of the observations to be stored and repeated updates are difficult. A simple adjustment to avoid cancellation, requiring only a single pass and little storage, uses the first observation to center:

$$\sum_{i=1}^{n} (y_i - \overline{y})^2 = \sum_{i=1}^{n} (y_i - y_1)^2 - n(y_1 - \overline{y})^2 \;. \tag{2.9}$$

An orthogonalization method from regression using Givens rotations (see Chan et al. 1983) can do even better to find $s_n = \sum_{i=1}^{n}(y_i - \bar{y})^2$:

$$t_i = t_{i-1} + y_i \tag{2.10}$$

$$s_i = s_{i-1} + (iy_i - t_i)^2/(i(i-1)) \; . \tag{2.11}$$

To illustrate the effect of cancellation, take the simple problem of $n = 5$ observations, $y_i = 4{,}152 + i$ so that $y_1 = 4{,}153$ through $y_5 = 4{,}157$. Again using six decimal digits, the computations of the sum and mean encounter no problems, and we easily get $\overline{y} = 4{,}155$ or 0.415500×10^4, and $\sum y_i = 20{,}775$ or 0.207750×10^5. However, each square loses some precision in rounding:

$$\begin{aligned}
y_1 &= 4{,}153 \; , \quad y_1^2 = 4{,}153^2 = 17{,}247{,}409 \quad \text{rounded to} \quad 0.172474 \times 10^8 \\
y_2 &= 4{,}154 \; , \quad y_2^2 = 4{,}154^2 = 17{,}255{,}716 \quad \text{rounded to} \quad 0.172557 \times 10^8 \\
y_3 &= 4{,}155 \; , \quad y_3^2 = 4{,}155^2 = 17{,}264{,}025 \quad \text{rounded to} \quad 0.172640 \times 10^8 \\
y_4 &= 4{,}156 \; , \quad y_4^2 = 4{,}156^2 = 17{,}272{,}336 \quad \text{rounded to} \quad 0.172723 \times 10^8 \\
y_5 &= 4{,}157 \; , \quad y_5^2 = 4{,}157^2 = 17{,}280{,}649 \quad \text{rounded to} \quad 0.172806 \times 10^8 \; .
\end{aligned}$$

Summing the squares encounters no further rounding on its way to 0.863200×10^8, and we compute the corrected sum of squares as

$$0.863200 \times 10^8 - (0.207750 \times 10^5) \times 4{,}155$$

$$0.863200 \times 10^8 - 0.863201 \times 10^8 = -100 \; .$$

The other three algorithms, following (2.8), (2.9), (2.10), and (2.11), each give the perfect result of 10 in this case.

Admittedly, while this example is contrived to show an absurd result, a negative sum of squares, the equally absurd value of zero is hardly unusual. Similar computations – differences of sum of squares – are routine, especially in regression and in the computation of eigenvalues and eigenvectors. In regression, the orthogonalization method (2.10) and (2.11) is more commonly seen in its general form. In all these cases, simply centering can improve the computational difficulty and reduce the effect of limited precision arithmetic.

2.1.4 Accumulated Roundoff Error

Another problem with floating point arithmetic is the sheer accumulation of rounding error. While many applications run well in spite of a large number of calculations, some approaches encounter surprising problems. An enlightening example is just to add up many ones: $1 + 1 + 1 + \ldots$. Astonishingly, this infinite

series appears to converge – the partial sums stop increasing as soon as the ratio of the new number to be added, in this case, one, to the current sum (n) drops below the machine epsilon. Following (2.5), we have $fl(n+1) = fl(n)$, from which we find

$$1/n \approx \epsilon_m \quad \text{or} \quad n \approx 1/\epsilon_m \, .$$

So you will find the infinite series of ones converging to $1/\epsilon_m$. Moving to double precision arithmetic pushes this limit of accuracy sufficiently far to avoid most problems – but it does not eliminate them. A good mnemonic for assessing the effect of accumulated rounding error is that doing m additions amplifies the rounding error by a factor of m. For single precision, adding 1,000 numbers would look like a relative error of 10^{-4} which is often unacceptable, while moving to double precision would lead to an error of 10^{-13}. Avoidance strategies, such as adding smallest to largest and nested partial sums, are discussed in detail in Monahan, (2001, Chap. 2).

2.1.5 Interval Arithmetic

One of the more interesting methods for dealing with the inaccuracies of floating point arithmetic is interval arithmetic. The key is that a computer can only do arithmetic operations: addition, subtraction, multiplication, and division. The novel idea, though, is that instead of storing the number x, its lower and upper bounds $(\underline{x}, \overline{x})$ are stored, designating an interval for x. Bounds for each of these arithmetic operations can be then established as functions of the input. For addition, the relationship can be written as:

$$\underline{x} + \underline{y} < x + y < \overline{x} + \overline{y} \, .$$

Similar bounds for the other three operations can be established. The propagation of rounding error through each step is then captured by successive upper and lower bounds on intermediate quantities. This is especially effective in probability calculations using series or continued fraction expansions. The final result is an interval that we can confidently claim contains the desired calculation. The hope is always that interval is small. Software for performing interval arithmetic has been implemented in a practical fashion by modifying a Fortran compiler. See, for example, Hayes (2003) for an introductory survey, and Kearfott and Kreinovich (1996) for articles on applications.

2.2 Algorithms

An algorithm is a list of directed actions to accomplish a designated task. Cooking recipes are the best examples of algorithms in everyday life. The level of a cookbook reflects the skills of the cook: a gourmet cookbook may include the instruction

"saute the onion until transparent" while a beginner's cookbook would describe how to choose and slice the onion, what kind of pan, the level of heat, etc. Since computers are inanimate objects incapable of thought, instructions for a computer algorithm must go much, much further to be completely clear and unambiguous, and include all details.

Most cooking recipes would be called *single pass* algorithms, since they are a list of commands to be completed in consecutive order. Repeating the execution of the same tasks, as in baking batches of cookies, would be described in algorithmic terms as *looping*. Looping is the most common feature in mathematical algorithms, where a specific task, or similar tasks, are to be repeated many times. The computation of an inner product is commonly implemented using a loop:

$$\boldsymbol{a}^\top \boldsymbol{b} = a_1 b_1 + a_2 b_2 + \ldots + a_n b_n \ ,$$

implemented as

```
s = 0
do i = 1 to n
    s = s + a_i × b_i
end do
```

where the range of the loop includes the single statement with a multiplication and addition. In an *iterative* algorithm, the number of times the loop is be repeated is not known in advance, but determined by some monitoring mechanism. For mathematical algorithms, the focus is most often monitoring convergence of a sequence or series. Care must be taken in implementing iterative algorithms to insure that, at some point, the loop will be terminated, otherwise an improperly coded procedure may proceed indefinitely in an *infinite loop*. Surprises occur when the convergence of an algorithm can be proven analytically, but, because of the discrete nature of floating point arithmetic, the procedure implementing that algorithm may not converge. For example, in a square-root problem to be examined further momentarily, we cannot find $x \in \mathcal{F}$ so that $x \times x$ is exactly equal to 2. The square of one number may be just below two, and the square of the next largest number in $\mathcal{F}$ may be larger than 2. When monitoring convergence, common practice is to convert any test for equality of two floating point numbers or expressions to tests of closeness:

$$\text{if } (\text{abs}(x^* x - 2) < \text{ eps}) \text{ then exit.} \tag{2.12}$$

Most mathematical algorithms have more sophisticated features. Some algorithms are *recursive*, employing relationships such as the gamma function: $\Gamma(x+1) = x\Gamma(x)$ so that new values can be computed using previous values. Powerful recursive algorithms, such as the Fast Fourier Transform (FFT) and sorting algorithms, follow a *divide-and-conquer* paradigm: to solve a big problem, break it into little problems and use the solutions to the little problems to solve the big problem. In the case of sorting, the algorithm may look something like:

```
algorithm sort(list)
break list into two pieces: first and second
sort (first)
sort (second)
put sorted lists first and second together to form
 one sorted list
end algorithm sort
```

Implemented recursively, a big problem is quickly broken into tiny pieces and the key to the performance of divide-and-conquer algorithms is in combining the solutions to lots of little problems to address the big problem. In cases where these solutions can be easily combined, these recursive algorithms can achieve remarkable breakthroughs in performance. In the case of sorting, the standard algorithm, known as bubblesort, takes $O(n^2)$ work to sort a problem of size n – if the size of the problem is doubled, the work goes up by factor of 4. The Discrete Fourier Transform, when written as the multiplication of an $n \times n$ matrix and a vector, involves n^2 multiplications and additions. In both cases, the problem is broken into two subproblems, and the mathematics of divide and conquer follows a simple recursive relationship, that the time/work $T(n)$ to solve a problem of size n is the twice the time/work to solve two subproblem with half the size, plus the time/work $C(n)$, to put the solutions together:

$$T(n) = 2T\,(n/2) + C(n)\,. \tag{2.13}$$

In both sorting and the Discrete Fourier Transform, $C(n) \approx cn + d$, which leads to $T(n) = cn\log(n) + O(n)$. A function growing at the rate $O(n\log n)$ grows so much slower than $O(n^2)$, that the moniker "Fast" in Fast Fourier Transform is well deserved. While some computer languages preclude the use of recursion, recursive algorithms can often be implemented without explicit recursion through clever programming.

The performance of an algorithm may be measured in many ways, depending on the characteristics of the problems the it may be intended to solve. The sample variance problem above provides an example. The simple algorithm using (2.7) requires minimal storage and computation, but may lose accuracy when the variance is much smaller than the mean: the common test problem for exhibiting catastrophic cancellation employs $y_i = 2^{12} + i$ for single precision. The two-pass method (2.8) requires all of the observations to be stored, but provides the most accuracy and least computation. Centering using the first observation (2.9) is nearly as fast, requires no extra storage, and its accuracy only suffers when the first observation is unlike the others. The last method, arising from the use of Givens transformations (2.10) and (2.11), also requires no extra storage, gives sound accuracy, but requires more computation. As commonly seen in the marketplace of ideas, the inferior methods have not survived, and the remaining competitors all have tradeoffs with speed, storage, and numerical stability.

2.2.1 Iterative Algorithms

The most common difficult numerical problems in statistics involve optimization, or root-finding: maximum likelihood, nonlinear least squares, M-estimation, solving the likelihood equations or generalized estimating equations. And the algorithms for solving these problems are typically iterative algorithms, using the results from the current step to direct the next step.

To illustrate, consider the problem of computing the square root of a real number y. Following from the previous discussion of floating point arithmetic, we can restrict y to the interval $(1, 2)$. One approach is to view the problem as a root-finding problem, that is, we seek x such that $f(x) = x^2 - y = 0$. The bisection algorithm is a simple, stable method for finding a root. In this case, we may start with an interval known to contain the root, say (x_1, x_2), with $x_1 = 1$ and $x_2 = 2$. Then bisection tries $x_3 = 1.5$, the midpoint of the current interval. If $f(x_3) < 0$, then $x_3 < \sqrt{y} < x_2$, and the root is known to belong in the new interval (x_3, x_2). The algorithm continues by testing the midpoint of the current interval, and eliminating half of the interval. The rate of convergence of this algorithm is *linear*, since the interval of uncertainty, in this case, is cut by a constant $(1/2)$ with each step. For other algorithms, we may measure the rate at which the distance from the root decreases. Adapting Newton's method to this root-finding problem yields Heron's iteration

$$x_{n+1} = \frac{1}{2}(x_n + y/x_n) \; .$$

Denoting the solution as $x^* = \sqrt{y}$, the error at step n can be defined as $\epsilon_n = x_n - x^*$, leading to the relationship

$$\epsilon_{n+1} = \frac{1}{2}\frac{\epsilon_n^2}{x_n} \; . \tag{2.14}$$

This relationship of the errors is usually called *quadratic convergence*, since the new error is proportional to the square of the error at the previous step. The relative error $\delta_n = (x_n - x^*)/x^*$ follows a similar relationship,

$$\delta_n = \frac{1}{2}\delta_n^2/(1 + \delta_n) \; . \tag{2.15}$$

Here, the number of accurate digits is doubled with each iteration. For the secant algorithm, analysis of the error often leads to a relationship similar to (2.14), but $|\epsilon_{n+1}| \approx C|\epsilon_n|^p$, with $1 < p < 2$, achieving a rate of convergence known as *superlinear*. For some well-defined problems, as the square root problem above, the number of iterations needed to reduce the error or relative error below some criterion can be determined in advance.

While we can stop this algorithm when $f(x_n) = 0$, as discussed previously, there may not be any floating point number that will give a zero to the function, hence the stopping rule (2.12). Often in root-finding problems, we stop when $|f(x_n)|$ is small

enough. In some problems, the appropriate "small enough" quantity to ensure the desired accuracy may depend on parameters of the problem, as in this case, the value of y. As a result, termination criterion for the algorithm is changed to: stop when the relative change in x is small

$$|x_{n+1} - x_n|/|x_n| < \delta \ .$$

While this condition may cause premature stopping in rare cases, it will prevent infinite looping in other cases. Many optimization algorithms permit the iteration to be terminated using any combination – and "small enough" is within the user's control. Nevertheless, unless the user learns a lot about the nature of the problem at hand, an unrealistic demand for accuracy can lead to unachievable termination criteria, and an endless search.

As discussed previously, rounding error with floating point computation affects the level of accuracy that is possible with iterative algorithms for root-finding. In general, the relative error in the root is at the same relative level as the computation of the function. While optimization problems have many of the same characteristics as root-finding problems, the effect of computational error is a bit more substantial: k digits of accuracy in the function to be optimization can produce but $k/2$ digits in the root/solution.

2.2.2 *Iterative Algorithms for Optimization and Nonlinear Equations*

In the multidimensional case, the common problems are solving a system of nonlinear equations or optimizing a function of several variables. The most common tools for these problems are Newton's method or secant-like variations. Given the appropriate regularity conditions, again we can achieve quadratic convergence with Newton's method, and superlinear convergence with secant-like variations. In the case of optimization, we seek to minimize $f(x)$, and Newton's method is based on minimizing the quadratic approximation:

$$f(\boldsymbol{x}) \approx f(\boldsymbol{x}_0) + (\boldsymbol{x} - \boldsymbol{x}_0)^\top \nabla f(\boldsymbol{x}_0) + (\boldsymbol{x} - \boldsymbol{x}_0)^\top \nabla^2 f(\boldsymbol{x}_0)(\boldsymbol{x} - \boldsymbol{x}_0) \ .$$

This leads to the iteration step

$$\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} - \left[\nabla^2 f\left(\boldsymbol{x}^{(n)}\right)\right]^{-1} \nabla f\left(\boldsymbol{x}^{(n)}\right) \ .$$

In the case of solving a system of nonlinear equations, $\boldsymbol{g}(\boldsymbol{x}) = \boldsymbol{0}$, Newton's method arises from solving the affine (linear) approximation

$$\boldsymbol{g}(\boldsymbol{x}) \approx \boldsymbol{g}\left(\boldsymbol{x}_0\right) + \boldsymbol{J}_{\boldsymbol{g}}\left(\boldsymbol{x}_0\right)\left(\boldsymbol{x} - \boldsymbol{x}_0\right) \ ,$$

leading to a similar iteration step

$$\boldsymbol{x}^{(n+1)} = \boldsymbol{x}^{(n)} - [\boldsymbol{J}_{\boldsymbol{g}}(\boldsymbol{x}^{(n)})]^{-1}\boldsymbol{g}(\boldsymbol{x}^{(n)}) \ .$$

In both cases, under suitable smoothness conditions, the Newton iteration will achieve quadratic convergence – using norms to measure the error at each step:

$$\left\|\boldsymbol{x}^{(n+1)} - \boldsymbol{x}^{*}\right\| \approx C \left\|\boldsymbol{x}^{(n)} - \boldsymbol{x}^{*}\right\|^{2} \ .$$

For both problems, Newton's method requires the computation of lots of derivatives, either the gradient $\nabla f(\boldsymbol{x}_0)$ and Hessian $\nabla^2 f(\boldsymbol{x}_0)$, or the Jacobian matrix $\boldsymbol{J}_{\boldsymbol{g}}(\boldsymbol{x}^{(n)})$. In the univariate root-finding problem, the secant method arises by approximating the derivative with the first difference using the previous evaluation of the function. Secant analogues can be constructed for both the optimization and nonlinear equations problems, with similar reduction in the convergence rate: from quadratic to superlinear.

In both problems, the scaling of the parameters is quite important, as measuring the error with the Euclidean norm presupposes that errors in each component are equally weighted. Most software for optimization includes a parameter vector for suitably scaling the parameters, so that one larger parameter does not dominate the convergence decision. In solving nonlinear equations, the condition of the problem is given by

$$\left\|\boldsymbol{J}_{\boldsymbol{g}}\left(\boldsymbol{x}^{(n)}\right)\right\| \left\|\left[\boldsymbol{J}_{\boldsymbol{g}}\left(\boldsymbol{x}^{(n)}\right)\right]^{-1}\right\|$$

(as in solving linear equations) and the problem of scaling involves the components of $\boldsymbol{g}(\boldsymbol{x})$. In many statistical problems, such as robust regression, the normal parameter scaling issues arise with the covariates and their coefficients. However, one component of $\boldsymbol{g}(\boldsymbol{x})$, associated with the error scale parameter may be orders of magnitude larger or smaller than the other equations. As with parameter scaling, this is often best done by the user and is not easily overcome automatically.

With the optimization problem, there is a natural scaling with $\nabla f(\boldsymbol{x}_0)$ in contrast with the Jacobian matrix. Here, the eigenvectors of the Hessian matrix $\nabla^2 f(\boldsymbol{x}_0)$ dictate the condition of the problem; see, for example, Gill et al. (1981) and Dennis and Schnabel (1983). Again, parameter scaling remains one of the most important tools.

References

Bodily, C.H.: Numerical Differentiation Using Statistical Design. Ph.D. Thesis, NC State University (2002)

Chan, T.F., Golub, G.H., LeVeque, R.J.: Algorithms for computing the sample variance. Am. Stat. **37**, 242–247 (1983)

Dennis, J.E. Jr., Schnabel, R.B.: Numerical Methods for Unconstrained Optimization. Prentice-Hall, Englewood Cliffs, NJ (1983)

Gill, P.E., Murray, W., Wright, M.H.: Practical Optimisation. Academic Press, London (1981)
Goldberg, D. (1991) *What Every Computer Scientist Should Know About Floating-Point Arithmetic*, ACM Computing Surveys 23(1):5–48.
Hayes, B.: A lucid interval. Am. Sci. **91**, 484–488 (2003)
Institute of Electrical and Electronics Engineers: A Proposed IEEE-CS Standard for Binary Floating Point Arithmetic. Standard, 754–1985, IEEE, New York (1985)
Kearfott, R.B., Kreinovich, V. (ed.): Applications of Interval Computations. Boston, Kluwer (1996)
Knuth, D.E.: The Art of Computer Programming, Seminumerical Algorithms, (3rd edn.), Vol. 2, Addison-Wesley, Reading MA (1997)
Monahan, J.F.: Numerical Methods of Statistics. Cambridge University Press, Cambridge (2001)
Overton, M.L.: Numerical Computing with IEEE Floating Point Arithmetic. Philadelphia, SIAM (2001)

Chapter 3
Random Number Generation

Pierre L'Ecuyer

3.1 Introduction

The fields of probability and statistics are built over the abstract concepts of probability space and random variable. This has given rise to elegant and powerful mathematical theory, but exact implementation of these concepts on conventional computers seems impossible. In practice, random variables and other random objects are *simulated* by *deterministic algorithms*. The purpose of these algorithms is to produce sequences of numbers or objects whose behavior is very hard to distinguish from that of their "truly random" counterparts, at least for the application of interest. Key requirements may differ depending on the context. For Monte Carlo methods, the main goal is to reproduce the statistical properties on which these methods are based, so that the Monte Carlo estimators behave as expected, whereas for gambling machines and cryptology, observing the sequence of output values for some time should provide no practical advantage for predicting the forthcoming numbers better than by just guessing at random.

In computational statistics, random variate generation is usually made in two steps: (1) generating imitations of independent and identically distributed (i.i.d.) random variables having the uniform distribution over the interval $(0, 1)$ and (2) applying transformations to these i.i.d. $U(0, 1)$ random variates to generate (or imitate) random variates and random vectors from arbitrary distributions. These two steps are essentially independent and the world's best experts on them are two different groups of scientists, with little overlap. The expression *(pseudo)random number generator* (RNG) usually refers to an algorithm used for step (1).

P. L'Ecuyer (✉)
Département d'Informatique et de Recherche Opérationnelle,
Université de Montréal,
Montréal (Québec), Canada
e-mail: lecuyer@iro.umontreal.ca
http://www.iro.umontreal.ca/~lecuyer

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_3,

In principle, the simplest way of generating a random variate X with distribution function F from a $U(0,1)$ random variate U is to apply the inverse of F to U:

$$X = F^{-1}(U) \stackrel{\text{def}}{=} \min\{x \mid F(x) \geq U\}. \tag{3.1}$$

This is the *inversion* method. It is easily seen that X has the desired distribution: $P[X \leq x] = P[F^{-1}(U) \leq x] = P[U \leq F(x)] = F(x)$. Other methods are sometimes preferable when F^{-1} is too difficult or expensive to compute, as will be seen later.

The remainder of this chapter is organized as follows. In the next section, we give a definition and the main requirements of a uniform RNG. Generators based on linear recurrences modulo a large integer m, their lattice structure and quality criteria, and their implementation, are covered in Sect. 3.3. In Sect. 3.4, we have a similar discussion for RNGs based on linear recurrences modulo 2. Nonlinear RNGs are briefly presented in Sect. 3.5. In Sect. 3.6, we discuss empirical statistical testing of RNGs and give some examples. Section 3.7 contains a few pointers to recommended RNGs and software. In Sect. 3.8, we cover non-uniform random variate generators. We first discuss inversion and its implementation in various settings. We then explain the rejection, ratio-of-uniform, composition and convolution methods, provide pointers to other methods that apply in special cases, and discuss multivariate distributions.

Important basic references that we recommend are Knuth (1998), L'Ecuyer (1994, 1998), Niederreiter (1992), and Tezuka (1995) for uniform RNGs, and Devroye (1986, 2006), Gentle (2003), and Hörmann et al. (2004) for non-uniform RNGs.

3.2 Uniform Random Number Generators

3.2.1 Physical Devices

Random numbers can be generated via physical mechanisms such as the timing between successive events in atomic decay, thermal noise in semiconductors, photon counting and photon trajectory detectors, and the like. A key issue when constructing a RNG based on a physical device is that a "random" or "chaotic" output does not suffice; the numbers produced must be, at least to a good approximation, realizations of *independent and uniformly distributed* random variables. If the device generates a stream of bits, which is typical, then each bit should be 0 or 1 with equal probability, and be independent of all the other bits. In general, this cannot be proved, so one must rely on the results of empirical statistical testing to get convinced that the output values have the desired statistical behavior, at least approximately. Not all these devices are reliable, but some are and, as far as we know, they pass all statistical tests that can be run in reasonable time.

For computational statistics, physical devices have several disadvantages compared to a good algorithmic RNG that stands in a few lines of code. For example, (a) they are much more cumbersome to install and run; (b) they are more costly; (c) they are slower; (d) they cannot reproduce exactly the same sequence twice. Item (d) is important in several contexts, including program verification and debugging as well as comparison of similar systems by simulation with common random numbers to reduce the variance (Bratley et al. 1987; Fishman 1996; Law and Kelton 2000). Nevertheless, these physical RNGs can be useful for selecting the seed of an algorithmic RNG, more particularly for applications in cryptology and for gaming machines, where frequent reseeding of the RNG with an external source of entropy (or randomness) is important. A good algorithmic RNG whose seed is selected at random can be viewed as an extensor of randomness, stretching a short random seed into a long sequence of *pseudorandom* numbers.

3.2.2 Generators Based on a Deterministic Recurrence

RNGs used for simulation and other statistical applications are almost always based on deterministic algorithms that fit the following framework, taken from L'Ecuyer (1994): a RNG is a structure $(\mathcal{S}, \mu, f, \mathcal{U}, g)$ where $\mathcal{S}$ is a finite set of *states* (the *state space*), μ is a probability distribution on $\mathcal{S}$ used to select the *initial state* (or *seed*) s_0, $f : \mathcal{S} \to \mathcal{S}$ is the *transition function*, $\mathcal{U}$ is the *output space*, and $g : \mathcal{S} \to \mathcal{U}$ is the *output function*. Usually, $\mathcal{U} = (0, 1)$, and we shall assume henceforth that this is the case. The state of the RNG evolves according to the recurrence $s_i = f(s_{i-1})$, for $i \geq 1$, and the *output* at step i is $u_i = g(s_i) \in \mathcal{U}$. The output values $u_0, u_1, u_2, \ldots$ are the so-called *random numbers* produced by the RNG.

Because $\mathcal{S}$ is finite, there must be some finite $l \geq 0$ and $j > 0$ such that $s_{l+j} = s_l$. Then, for all $i \geq l$, one has $s_{i+j} = s_i$ and $u_{i+j} = u_i$, because both f and g are deterministic. That is, the state and output sequences are eventually periodic. The smallest positive j for which this happens is called the *period* of the RNG, and is denoted by ρ. When $l = 0$, the sequence is said to be *purely periodic*. Obviously, $\rho \leq |\mathcal{S}|$, the cardinality of $\mathcal{S}$. If the state has a k-bit representation on the computer, then $\rho \leq 2^k$. Good RNGs are designed so that their period ρ is not far from that upper bound. In general, the value of ρ may depend on the seed s_0, but good RNGs are normally designed so that the period is the same for all admissible seeds.

In practical implementations, it is important that the output be *strictly* between 0 and 1, because $F^{-1}(U)$ is often infinite when U is 0 or 1. All good implementations take care of that. However, for the mathematical analysis of RNGs, we often assume that the output space is $[0, 1)$ (i.e., 0 is admissible), because this simplifies the analysis considerably without making much difference in the mathematical structure of the generator.

3.2.3 *Quality Criteria*

What important quality criteria should we consider when designing RNGs? An extremely *long period* is obviously essential, to make sure that no wrap-around over the cycle can occur in practice. The length of the period must be guaranteed by a mathematical proof. The RNG must also be *efficient* (run fast and use only a small amount of memory), *repeatable* (able to reproduce exactly the same sequence as many times as we want), and *portable* (work the same way in different software/hardware environments). The availability of efficient *jump-ahead* methods that can quickly compute $s_{i+\nu}$ given s_i, for any large ν and any i, is also very useful, because it permits one to partition the RNG sequence into long disjoint *streams* and *substreams* of random numbers, to create an arbitrary number of *virtual generators* from a single RNG (Law and Kelton 2000; L'Ecuyer 2008; L'Ecuyer et al. 2002). These virtual generators can be used on parallel processors or to support different sources of randomness in a large simulation model, for example.

To show that these properties are *not sufficient*, consider a RNG with state space $\mathcal{S} = \{0, \ldots, 2^{1000} - 1\}$, transition function $s_{i+1} = f(s_i) = (s_i + 1) \bmod 2^{1000}$, and $u_i = g(s_i) = s_i / 2^{1000}$. This RNG has period 2^{1000} and enjoys all the nice properties described in the preceding paragraph, but it is far from imitating "randomness."

A sequence of real-valued random variables $u_0, u_1, u_2, \ldots$ are i.i.d. $U(0, 1)$ if and only if for all integers $i \geq 0$ and $t > 0$, the vector $\mathbf{u}_{i,t} = (u_i, \ldots, u_{i+t-1})$ is uniformly distributed over the t-dimensional unit hypercube $(0, 1)^t$. Of course, this cannot hold for algorithmic RNGs because any vector of t successive values produced by the generator must belong to the *finite set*

$$\Psi_t = \{(u_0, \ldots, u_{t-1}) : s_0 \in \mathcal{S}\},$$

which is the set of all vectors of t successive output values, from all possible initial states. Here we interpret Ψ_t as a *multiset*, which means that the vectors are counted as many times as they appear, and the cardinality of Ψ_t is exactly equal to that of $\mathcal{S}$.

Suppose we select the seed s_0 at random, uniformly over $\mathcal{S}$. This can be approximated by using some physical device, for example. Then, the vector $\mathbf{u}_{0,t}$ has the uniform distribution over the finite set Ψ_t. And if the sequence is purely periodic for all s_0, $\mathbf{u}_{i,t} = (u_i, \ldots, u_{i+t-1})$ is also uniformly distributed over Ψ_t for all $i \geq 0$. Since the goal is to approximate the uniform distribution over $(0, 1)^t$, it immediately becomes apparent that Ψ_t should be evenly spread over this unit hypercube. In other words, Ψ_t *approximates* $(0, 1)^t$ as the *sample space* from which the vectors of successive output values are drawn randomly, so it must be a good approximation of $(0, 1)^t$ in some sense. The design of good-quality RNGs must therefore involve practical ways of measuring the uniformity of the corresponding sets Ψ_t even when they have huge cardinalities. In fact, a large state space $\mathcal{S}$ is necessary to obtain a long period, but an even more important reason for having a huge number of states is to make sure that Ψ_t can be large enough to provide a good uniform coverage of the unit hypercube, at least for moderate values of t.

More generally, we may also want to measure the uniformity of sets of the form

$$\Psi_I = \{(u_{i_1}, \ldots, u_{i_t}) \mid s_0 \in \mathcal{S}\},$$

where $I = \{i_1, \cdots, i_t\}$ is a fixed set of non-negative integers such that $0 \le i_1 < \cdots < i_t$. As a special case, we recover $\Psi_t = \Psi_I$ when $I = \{0, \ldots, t-1\}$. Of course, there are so many such sets Ψ_I that we cannot examine the uniformity over all of them, but we can do it over a selected class $\mathcal{J}$ of such sets deemed more important.

The uniformity of a set Ψ_I is typically assessed by measuring the *discrepancy* between the empirical distribution of its points and the uniform distribution over $(0,1)^t$ (L'Ecuyer 2009; L'Ecuyer and Lemieux 2002; Niederreiter 1992). Discrepancy measures are equivalent to goodness-of-fit test statistics for the multivariate uniform distribution. They can be defined in many different ways. The choice of a specific definition typically depends on the mathematical structure of the RNG to be studied and the reason for this is very pragmatic: we must be able to compute these measures quickly even when $\mathcal{S}$ has very large cardinality, for instance 2^{200} or more. This obviously excludes any method that requires explicit generation of the sequence over its entire period. The selected discrepancy measure is usually computed for each set I in a predefined class $\mathcal{J}$, these values are weighted or normalized by factors that depend on I, and the worst-case (or average) over $\mathcal{J}$ is adopted as a *figure of merit* used to rank RNGs. The choice of $\mathcal{J}$ and of the weights are arbitrary. Typically, $\mathcal{J}$ would contain sets I such that t and $i_t - i_1$ are rather small. Examples of such figures of merit will be given when we discuss specific classes of RNGs.

3.2.4 Statistical Testing

Good RNGs are designed based on mathematical analysis of their properties, then implemented and submitted to batteries of *empirical statistical tests*. These tests try to detect empirical evidence against the null hypothesis $\mathcal{H}_0$: "the u_i are realizations of i.i.d. $U(0,1)$ random variables." A test can be defined by any function T that maps a sequence $u_0, u_1, \ldots$ in $(0,1)$ to a real number X, and for which a good approximation is available for the distribution of the random variable X under $\mathcal{H}_0$. For the test to be implementable, X must depend on only a finite (but perhaps random) number of u_i's. Passing many tests may improve one's confidence in the RNG, but never guarantees that the RNG is foolproof for all kinds of simulations.

Building a RNG that *passes all statistical tests* is an impossible dream. Consider, for example, the class of all tests that examine the first (most significant) b bits of n successive output values, $u_0, \ldots, u_{n-1}$, and return a binary value $X \in \{0,1\}$. Select $\alpha \in (0,1)$ so that $\alpha 2^{nb}$ is an integer and let $\mathcal{T}_{n,b,\alpha}$ be the set of tests in this class that return $X = 1$ for *exactly* $\alpha 2^{nb}$ of the 2^{nb} possible output sequences. We say that the sequence *fails the test* when $X = 1$. This $\mathcal{T}_{n,b,\alpha}$ is the set of all statistical tests of (exact) level α. Its cardinality is equal to the number of ways of choosing $\alpha 2^{nb}$

distinct objects among 2^{nb}. The chosen objects are the sequences that fail the test. For any given output sequence, the number of tests in $\mathcal{T}_{n,b,\alpha}$ that return 1 for this sequence is equal to the number of ways of choosing the other $\alpha 2^{nb} - 1$ sequences that also fail the test. This is the number of ways of choosing $\alpha 2^{nb} - 1$ distinct objects among $2^{nb} - 1$. In other words, as pointed out by Leeb (1995), every output sequence fails exactly the same number of tests! Viewed from a different angle, it is a restatement of the well-known fact that under $\mathcal{H}_0$, each of the 2^{nb} possible sequences has the same probability of occurring, so one may argue that none should be considered more random than any other (Knuth 1998).

This viewpoint seems to lead into a dead end. For statistical testing to be meaningful, all tests should not be considered on equal footing. So which ones are more important? Any answer is tainted with arbitrariness. However, for large values of n, the number of tests is huge and all but a tiny fraction are too complicated even to be implemented. So we may say that *bad* RNGs are those that fail simple tests, whereas *good* RNGs fail only complicated tests that are hard to find and run. This common-sense compromise has been generally adopted in one way or another.

Experience shows that RNGs with very long periods, good structure of their set Ψ_t, and based on recurrences that are not too simplistic, pass most reasonable tests, whereas RNGs with short periods or bad structures are usually easy to crack by standard statistical tests. For sensitive applications, it is a good idea, when this is possible, to apply additional statistical tests designed in close relation with the random variable of interest (e.g., based on a *simplification* of the stochastic model being simulated, and for which the theoretical distribution can be computed).

Our discussion of statistical tests continues in Sect. 3.6. A key reference is L'Ecuyer and Simard (2007).

3.2.5 Cryptographically Strong Generators

One way of defining an ideal RNG would be that no statistical test can distinguish its output sequence from an i.i.d. $U(0, 1)$ sequence. If an unlimited computing time is available, no finite-state RNG can satisfy this requirement, because by running it long enough one can eventually figure out its periodicity. But what if we impose a limit on the computing time? This can be analyzed formally in the framework of asymptotic *computational complexity* theory, under the familiar "rough-cut" assumption that polynomial-time algorithms are practical and others are not.

Consider a family of RNGs $\{\mathcal{G}_k = (\mathcal{S}_k, \mu_k, f_k, \mathcal{U}_k, g_k),\ k = 1, 2, \dots\}$ where $\mathcal{S}_k$ of cardinality 2^k (i.e., $\mathcal{G}_k$ has a k-bit state). Suppose that the transition and output functions f and g can be computed in time bounded by a polynomial in k. Let $\mathcal{T}$ be the class of statistical tests that run in time bounded by a polynomial in k and try to differentiate between the output sequence of the RNG and an i.i.d. $U(0, 1)$ sequence. The RNG family is called *polynomial-time perfect* if there is a constant $\epsilon > 0$ such that for all k, no test in $\mathcal{T}$ can differentiate correctly with probability

larger than $1/2 + e^{-k\epsilon}$. This is equivalent to asking that no polynomial-time algorithm can predict any given bit of u_i with probability of success larger than $1/2 + e^{-k\epsilon}$, after observing $u_0, \ldots, u_{i-1}$. This links unpredictability with statistical uniformity and independence. For the proofs and additional details, see, e.g. Blum et al. (1986), L'Ecuyer and Proulx (1989), Lagarias (1993), and Luby (1996). This theoretical framework has been used to define a notion of reliable RNG in the context of cryptography. But the guarantee is only asymptotic; it does not necessarily tell what value of k is large enough for the RNG to be secure in practice. Moreover, specific RNG families have been proved to be polynomial-time perfect only under yet unproven conjectures. So far, no one has been able to prove even their existence. Most RNGs discussed in the remainder of this chapter are known *not* to be polynomial-time perfect. However, they are fast, convenient, and have good enough statistical properties when their parameters are chosen carefully.

3.3 Linear Recurrences Modulo *m*

3.3.1 The Multiple Recursive Generator

The most widely used RNGs are based on the linear recurrence

$$x_i = (a_1 x_{i-1} + \cdots + a_k x_{i-k}) \bmod m, \tag{3.2}$$

where m and k are positive integers called the *modulus* and the *order*, and the *coefficients* $a_1, \ldots, a_k$ are in $\mathbb{Z}_m$, interpreted as the set $\{0, \ldots, m-1\}$ on which all operations are performed with reduction modulo m. The *state* at step i is $s_i = \mathbf{x}_i = (x_{i-k+1}, \ldots, x_i)^\mathsf{T}$. When m is a prime number, the finite ring $\mathbb{Z}_m$ is a finite field and it is possible to choose the coefficients a_j so that the period reaches $\rho = m^k - 1$ (the largest possible value) (Knuth 1998). This maximal period is achieved if and only if the characteristic polynomial of the recurrence, $P(z) = z^k - a_1 z^{k-1} - \cdots - a_k$, is a primitive polynomial over $\mathbb{Z}_m$, i.e., if and only if the smallest positive integer ν such that $(z^\nu \bmod P(z)) \bmod m = 1$ is $\nu = m^k - 1$. Knuth (1998) explains how to verify this for a given $P(z)$. For $k > 1$, for $P(z)$ to be a primitive polynomial, it is necessary that a_k and at least another coefficient a_j be nonzero. Finding primitive polynomials of this form is generally easy and they yield the simplified recurrence:

$$x_n = (a_r x_{n-r} + a_k x_{n-k}) \bmod m. \tag{3.3}$$

A *multiple recursive generator* (MRG) uses (3.2) with a large value of m and defines the output as $u_i = x_i / m$. For $k = 1$, this is the classical *linear congruential generator* (LCG). In practice, the output function is modified slightly to make sure

that u_i never takes the value 0 or 1 (e.g., one may define $u_i = (x_i + 1)/(m + 1)$, or $u_i = x_i/(m + 1)$ if $x_i > 0$ and $u_i = m/(m + 1)$ otherwise) but to simplify the theoretical analysis, we will follow the common convention of assuming that $u_i = x_i/m$ (in which case u_i *does* take the value 0 occasionally).

3.3.2 The Lattice Structure

Let $\mathbf{e}_i$ denote the ith unit vector in k dimensions, with a 1 in position i and 0's elsewhere. Denote by $x_{i,0}, x_{i,1}, x_{i,2}, \ldots$ the values of $x_0, x_1, x_2, \ldots$ produced by the recurrence (3.2) when the initial state $\mathbf{x}_0$ is $\mathbf{e}_i$. An arbitrary initial state $\mathbf{x}_0 = (z_1, \ldots, z_k)^\mathsf{T}$ can be written as the linear combination $z_1\mathbf{e}_1 + \cdots + z_k\mathbf{e}_k$ and the corresponding sequence is a linear combination of the sequences $(x_{i,0}, x_{i,1}, \ldots)$, with reduction of the coordinates modulo m. Conversely, any such linear combination reduced modulo m is a sequence that can be obtained from some initial state $\mathbf{x}_0 \in \mathcal{S} = \mathbb{Z}_m^k$. If we divide everything by m we find that for the MRG, for each $t \geq 1$, $\Psi_t = L_t \cap [0, 1)^t$ where

$$L_t = \left\{ \mathbf{v} = \sum_{i=1}^{t} z_i \mathbf{v}_i \;\mid\; z_i \in \mathbb{Z} \right\},$$

is a t-dimensional *lattice* in $\mathbb{R}^t$, with basis

$$\begin{aligned}
\mathbf{v}_1 &= (1, 0, \ldots, 0, x_{1,k}, \ldots, x_{1,t-1})^\mathsf{T}/m \\
&\vdots \quad \vdots \\
\mathbf{v}_k &= (0, 0, \ldots, 1, x_{k,k}, \ldots, x_{k,t-1})^\mathsf{T}/m \\
\mathbf{v}_{k+1} &= (0, 0, \ldots, 0, 1, \ldots, 0)^\mathsf{T} \\
&\vdots \quad \vdots \\
\mathbf{v}_t &= (0, 0, \ldots, 0, 0, \ldots, 1)^\mathsf{T}.
\end{aligned}$$

For $t \leq k$, L_t contains all vectors whose coordinates are multiples of $1/m$. For $t > k$, it contains a fraction m^{k-t} of those vectors.

This lattice structure implies that the points of Ψ_t are distributed according to a very regular pattern, in equidistant parallel hyperplanes. Graphical illustrations of this, usually for LCGs, can be found in a myriad of papers and books; e.g., Gentle (2003), Law and Kelton (2000), and L'Ecuyer (1998). Define the *dual lattice* to L_t as

$$L_t^* = \{\mathbf{h} \in \mathbb{R}^t : \mathbf{h}^\mathsf{T}\mathbf{v} \in \mathbb{Z} \text{ for all } \mathbf{v} \in L_t\}.$$

Each $\mathbf{h} \in L_t^*$ is a normal vector that defines a family of equidistant parallel hyperplanes, at distance $1/\|\mathbf{h}\|_2$ apart, and these hyperplanes cover all the points of L_t unless $\mathbf{h}$ is an integer multiple of some other vector $\mathbf{h}' \in L_t^*$. Therefore, if ℓ_t is the Euclidean length of a shortest non-zero vector $\mathbf{h}$ in L_t^*, then there is a family of hyperplanes at distance $1/\ell_t$ apart that cover all the points of L_t. A small ℓ_t means there are thick slices of empty space between the hyperplanes and we want to avoid that. A large ℓ_t means a better (more uniform) coverage of the unit hypercube by the point set Ψ_t. Computing the value of $1/\ell_t$ is often called the *spectral test* (Fishman 1996; Knuth 1998).

The lattice property holds as well for the point sets Ψ_I formed by values at arbitrary lags defined by a fixed set of indices $I = \{i_1, \cdots, i_t\}$. One has $\Psi_I = L_I \cap [0,1)^t$ for some lattice L_I, and the largest distance between successive hyperplanes for a family of hyperplanes that cover all the points of L_I is $1/\ell_I$, where ℓ_I is the Euclidean length of a shortest nonzero vector in L_I^*, the dual lattice to L_I.

The lattice L_I and its dual can be constructed as explained in Couture and L'Ecuyer (1996) and L'Ecuyer and Couture (1997). Finding the shortest nonzero vector in a lattice with basis $\mathbf{v}_1, \ldots, \mathbf{v}_t$ can be formulated as an integer programming problem with a quadratic objective function:

$$\text{Minimize } \|\mathbf{v}\|^2 = \sum_{i=1}^{t} \sum_{j=1}^{t} z_i \mathbf{v}_i^\mathsf{T} \mathbf{v}_j z_j$$

subject to $z_1, \ldots, z_t$ integers and not all zero. This problem can be solved by a branch-and-bound algorithm (Fincke and Pohst 1985; L'Ecuyer and Couture 1997; Tezuka 1995).

For any given dimension t and m^k points per unit of volume, there is an absolute upper bound on the best possible value of ℓ_I (Conway and Sloane 1999; Knuth 1998; L'Ecuyer 1999b). Let $\ell_t^*(m^k)$ denote such an upper bound. To define a figure of merit that takes into account several sets I, in different numbers of dimensions, it is common practice to divide ℓ_I by an upper bound, to obtain a standardized value between 0 and 1, and then take the worst case over a given class $\mathcal{J}$ of sets I. This gives a figure of merit of the form

$$M_{\mathcal{J}} = \min_{I \in \mathcal{J}} \ell_I / \ell_{|I|}^*(m^k).$$

A value of $M_{\mathcal{J}}$ too close to zero means that L_I has a bad lattice structure for at least one of the selected sets I. We want a value as close to 1 as possible. Computer searches for good MRGs with respect to this criterion have been reported by L'Ecuyer et al. (1993), L'Ecuyer and Andres (1997), L'Ecuyer (1999a), for example. In most cases, $\mathcal{J}$ was simply the sets of the form $I = \{1, \ldots, t\}$ for $t \le t_1$, where t_1 was an arbitrary integer ranging from 8 to 45. L'Ecuyer and Lemieux (2000) also consider the small dimensional sets I with indices not too

far apart. They suggest taking $\mathcal{J} = \{\{0, 1, \ldots, i\} : i < t_1\} \cup \{\{i_1, i_2\} : 0 = i_1 < i_2 < t_2\} \cup \cdots \cup \{\{i_1, \ldots, i_d\} : 0 = i_1 < \ldots < i_d < t_d\}$ for some positive integers $d, t_1, \ldots, t_d$. We could also take a weighted average instead of the minimum in the definition of $M_{\mathcal{J}}$.

An important observation is that for $t > k$, the t-dimensional vector $\mathbf{h} = (-1, a_1, \ldots, a_k, 0, \ldots, 0)^{\mathsf{T}}$ always belong to L_t^*, because for any vector $\mathbf{v} \in L_t$, the first $k+1$ coordinates of $m\mathbf{v}$ must satisfy the recurrence (3.2), which implies that $(-1, a_1, \ldots, a_k, 0, \ldots, 0)\mathbf{v}$ must be an integer. Therefore, one always has $\ell_t^2 \leq 1 + a_1^2 + \cdots + a_k^2$. Likewise, if I contains 0 and all indices j such that $a_{k-j} \neq 0$, then $\ell_I^2 \leq 1 + a_1^2 + \cdots + a_k^2$ (L'Ecuyer 1997). This means that the sum of squares of the coefficients a_j *must be large* if we want to have any chance that the lattice structure be good.

Constructing MRGs with only two nonzero coefficients and taking these coefficients small has been a very popular idea, because this makes the implementation easier and faster (Deng and Lin 2000; Knuth 1998). However, the MRGs thus obtained have a bad structure. As a worst-case illustration, consider the widely-available additive or subtractive *lagged-Fibonacci* generator, based on the recurrence (3.2) where the two coefficients a_r and a_k are both equal to ± 1. In this case, whenever I contains $\{0, k-r, k\}$, one has $\ell_I^2 \leq 3$, so the distance between the hyperplanes is at least $1/\sqrt{3}$. In particular, for $I = \{0, k-r, k\}$, all the points of Ψ_I (aside from the zero vector) are contained in only two planes! This type of structure can have a dramatic effect on certain simulation problems and is a good reason for staying away from these lagged-Fibonacci generators, regardless of their parameters. They fail several simple empirical statistical tests (L'Ecuyer and Simard 2007).

A similar problem occurs for the "fast MRG" proposed by Deng and Lin (2000), based on the recurrence

$$x_i = (-x_{i-1} + a x_{i-k}) \bmod m = ((m-1)x_{i-1} + a x_{i-k}) \bmod m,$$

with $a^2 < m$. If a is small, the bound $\ell_I^2 \leq 1 + a^2$ implies a bad lattice structure for $I = \{0, k-1, k\}$. A more detailed analysis by L'Ecuyer and Touzin (2004) shows that this type of generator cannot have a good lattice structure even if the condition $a^2 < m$ is removed. Another special case proposed by Deng and Xu (2003) has the form

$$x_i = a(x_{i-j_2} + \cdots + x_{i-j_t}) \bmod m. \tag{3.4}$$

In this case, for $I = \{0, k-j_{t-1}, \ldots, k-j_2, k\}$, the vectors $(1, a, \ldots, a)$ and $(a^*, 1, \ldots, 1)$ both belong to the dual lattice L_I^*, where a^* is the multiplicative inverse of a modulo m. So neither a nor a^* should be small.

To get around this structural problem when I contains certain sets of indices, Lüscher (1994) and Knuth (1998) recommend to skip some of the output values to break up the bad vectors. For the lagged-Fibonacci generator, for example, one can output k successive values produced by the recurrence, then skip the next d values, output the next k, skip the next d, and so on. A large value of d (e.g., $d = 5k$ or more) may get rid of the bad structure, but slows down the generator. See Wegenkittl and Matsumoto (1999) for further discussion.

3.3.3 MRG Implementation Techniques

The modulus m is often taken as a large prime number close to the largest integer directly representable on the computer (e.g., equal or near $2^{31} - 1$ for 32-bit computers). Since each x_{i-j} can be as large as $m - 1$, one must be careful in computing the right side of (3.2) because the product $a_j x_{i-j}$ is typically not representable as an ordinary integer. Various techniques for computing this product modulo m are discussed and compared by Fishman (1996), L'Ecuyer and Tezuka (1991), L'Ecuyer (1999a), and L'Ecuyer and Simard (1999). Note that if $a_j = m - a'_j > 0$, using a_j is equivalent to using the negative coefficient $-a'_j$, which is sometimes more convenient from the implementation viewpoint. In what follows, we assume that a_j can be either positive or negative.

One approach is to perform the arithmetic modulo m in 64-bit (double precision) floating-point arithmetic (L'Ecuyer 1999a). Under this representation, assuming that the usual IEEE floating-point standard is respected, all positive integers up to 2^{53} are represented exactly. Then, if each coefficient a_j is selected to satisfy $|a_j|(m-1) \le 2^{53}$, the product $|a_j| x_{i-j}$ will always be represented exactly and $z_j = |a_j| x_{i-j} \bmod m$ can be computed by the instructions

$$y = |a_j| x_{i-j}; \qquad z_j = y - m\lfloor y/m \rfloor.$$

Similarly, if $(|a_1| + \cdots + |a_k|)(m-1) \le 2^{53}$, $a_1 x_{i-1} + \cdots + a_k x_{i-k}$ will always be represented exactly.

A second technique, called *approximate factoring* (L'Ecuyer and Côté 1991), uses only the integer representation and works under the condition that $|a_j| = i$ or $|a_j| = \lfloor m/i \rfloor$ for some integer $i < \sqrt{m}$. One precomputes $q_j = \lfloor m/|a_j| \rfloor$ and $r_j = m \bmod |a_j|$. Then, $z_j = |a_j| x_{i-j} \bmod m$ can be computed by

$$y = \lfloor x_{i-j}/q_j \rfloor; \qquad z = |a_j|(x_{i-j} - yq_j) - yr_j;$$
$$\text{if } z < 0 \text{ then } z_j = z + m \text{ else } z_j = z.$$

All quantities involved in these computations are integers between $-m$ and m, so no overflow can occur if m can be represented as an ordinary integer (e.g., $m < 2^{31}$ on a 32-bit computer).

The *powers-of-two decomposition* approach selects coefficients a_j that can be written as a sum or difference of a small number of powers of 2 (L'Ecuyer and Simard 1999; L'Ecuyer and Touzin 2000; Wu 1997). For example, one may take $a_j = \pm 2^q \pm 2^r$ and $m = 2^e - h$ for some positive integers q, r, e, and h. To compute $y = 2^q x \bmod m$, decompose $x = z_0 + 2^{e-q} z_1$ (where $z_0 = x \bmod 2^{e-q}$) and observe that

$$y = 2^q(z_0 + 2^{e-q} z_1) \bmod (2^e - h) \;=\; (2^q z_0 + h z_1) \bmod (2^e - h).$$

Suppose now that

$$h < 2^q \quad \text{and} \quad h(2^q - (h+1)2^{-e+q}) < m. \tag{3.5}$$

Then, $2^q z_0 < m$ and $hz_1 < m$, so y can be computed by shifts, masks, additions, subtractions, and a single multiplication by h. Intermediate results never exceed $2m-1$. Things simplify further if $q = 0$ or $q = 1$ or $h = 1$. For $h = 1$, y is obtained simply by swapping the blocks of bits z_0 and z_1 (Wu 1997). L'Ecuyer and Simard (1999) pointed out that LCGs with parameters of the form $m = 2^e - 1$ and $a = \pm 2^q \pm 2^r$ have bad statistical properties because the recurrence does not "mix the bits" well enough. However, good and fast (combined) MRGs can be obtained via the power-of-two decomposition method, as explained in L'Ecuyer and Touzin (2000).

Another idea to improve efficiency is to take all nonzero coefficients a_j equal to the same a, as in (3.4) (Deng and Xu 2003; Marsaglia 1996). Then, computing the right side of (3.2) requires a single multiplication. Deng and Xu (2003) and Deng (2005) provide specific parameter sets and concrete implementations for MRGs of this type, for prime m near 2^{31}, and for k ranging from 102 to 1597.

One may be tempted to take m equal to a power of two, say $m = 2^e$, because then the "$\bmod m$" operation is much easier: it suffices to keep the e least significant bits and mask-out all others. However, taking a power-of-two modulus is not recommended because it has several strong disadvantages in terms of the quality of the RNG (L'Ecuyer 1990, 1998). In particular, the least significant bits have very short periodicity and the period of the recurrence (3.2) cannot exceed $(2^k - 1)2^{e-1}$ if $k > 1$, and 2^{e-2} if $k = 1$ and $e \geq 4$. The maximal period achievable with $k = 7$ and $m = 2^{31}$, for example, is more than 2^{180} times smaller than the maximal period achievable with $k = 7$ and $m = 2^{31} - 1$ (a prime number).

3.3.4 Combined MRGs and LCGs

The conditions that make MRG implementations run faster (e.g., only two nonzero coefficients both close to zero) conflict with those required for having a good lattice structure and statistical robustness. *Combined MRGs* are one solution to this problem. Consider J distinct MRGs evolving in parallel, based on the recurrences

$$x_{j,i} = (a_{j,1}x_{j,i-1} + \cdots + a_{j,k}x_{j,i-k}) \bmod m_j \tag{3.6}$$

where $a_{j,k} \neq 0$, for $j = 1, \ldots, J$. Let $\delta_1, \ldots, \delta_J$ be arbitrary integers,

$$z_i = (\delta_1 x_{1,i} + \cdots + \delta_J x_{J,i}) \bmod m_1, \qquad u_i = z_i/m_1, \tag{3.7}$$

and

$$w_i = (\delta_1 x_{1,i}/m_1 + \cdots + \delta_J x_{J,i}/m_J) \bmod 1. \tag{3.8}$$

This defines two RNGs, with output sequences $\{u_i, i \geq 0\}$ and $\{w_i, i \geq 0\}$.

Suppose that the m_j are pairwise relatively prime, that δ_j and m_j have no common factor for each j, and that each recurrence (3.6) is purely periodic with period ρ_j. Let $m = m_1 \cdots m_J$ and let ρ be the least common multiple of $\rho_1, \ldots, \rho_J$. Under these conditions, L'Ecuyer and Tezuka (1991) and L'Ecuyer (1996a) proved the following: (a) the sequence (3.8) is exactly equivalent to the output sequence of a MRG with (composite) modulus m and coefficients a_j that can be computed explicitly as explained by L'Ecuyer (1996a); (b) the two sequences in (3.7) and (3.8) have period ρ; and (c) if both sequences have the same initial state, then $u_i = w_i + \epsilon_i$ where $\max_{i \geq 0} |\epsilon_i|$ can be bounded explicitly by a constant ϵ which is very small when the m_j are close to each other.

Thus, these combined MRGs can be viewed as practical ways of implementing an MRG with a large m and several large nonzero coefficients. The idea is to cleverly select the components so that: (1) each one is easy to implement efficiently (e.g., has only two small nonzero coefficients) and (2) the MRG that corresponds to the combination has a good lattice structure. If each m_j is prime and if each component j has maximal period $\rho_j = m_j^k - 1$, then each ρ_j is even and ρ cannot exceed $\rho_1 \cdots \rho_J / 2^{J-1}$. Tables of good parameters for combined MRGs of different sizes that reach this upper bound are given in L'Ecuyer (1999a) and L'Ecuyer and Touzin (2000), together with C implementations.

3.3.5 *Jumping Ahead*

The recurrence (3.2) can be written in matrix form as

$$\mathbf{x}_i = \mathbf{A}\mathbf{x}_{i-1} \bmod m = \begin{pmatrix} 0 & 1 & \cdots & 0 \\ \vdots & & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \\ a_k & a_{k-1} & \cdots & a_1 \end{pmatrix} \mathbf{x}_{i-1} \bmod m.$$

To jump ahead directly from $\mathbf{x}_i$ to $\mathbf{x}_{i+\nu}$, for an arbitrary integer ν, it suffices to exploit the relationship

$$\mathbf{x}_{i+\nu} = \mathbf{A}^\nu \mathbf{x}_i \bmod m = (\mathbf{A}^\nu \bmod m)\mathbf{x}_i \bmod m.$$

If this is to be done several times for the same ν, the matrix $\mathbf{A}^\nu \bmod m$ can be precomputed once for all. For a large ν, this can be done in $O(\log_2 \nu)$ matrix multiplications via a standard divide-and-conquer algorithm (Knuth 1998):

$$\mathbf{A}^\nu \bmod m = \begin{cases} (\mathbf{A}^{\nu/2} \bmod m)(\mathbf{A}^{\nu/2} \bmod m) \bmod m & \text{if } \nu \text{ is even;} \\ \mathbf{A}(\mathbf{A}^{\nu-1} \bmod m) \bmod m & \text{if } \nu \text{ is odd.} \end{cases}$$

3.3.6 Linear Recurrences with Carry

These types of recurrences were introduced by Marsaglia and Zaman (1991) to obtain a large period even when m is a power of two (in which case the implementation may be faster). They were studied and generalized by Tezuka et al. (1994), Couture and L'Ecuyer (1994, 1997), and Goresky and Klapper (2003). The basic idea is to add a *carry* to the linear recurrence (3.2). The general form of this RNG, called *multiply-with-carry* (MWC), can be written as

$$x_i = (a_1 x_{i-1} + \cdots + a_k x_{i-k} + c_{i-1}) d \bmod b, \tag{3.9}$$

$$c_i = \lfloor (a_0 x_i + a_1 x_{i-1} + \cdots + a_k x_{i-k} + c_{i-1})/b \rfloor, \tag{3.10}$$

$$u_i = \sum_{\ell=1}^{\infty} x_{i-\ell+1} b^{-\ell}, \tag{3.11}$$

where b is a positive integer (e.g., a power of two), $a_0, \ldots, a_k$ are arbitrary integers such that a_0 is relatively prime to b, and d is the multiplicative inverse of $-a_0$ modulo b. The state at step i is $s_i = (x_{i-k+1}, \ldots, x_i, c_i)^{\mathsf{T}}$. In practice, the sum in (3.11) is truncated to a few terms (it could be a single term if b is large), but the theoretical analysis is much easier for the infinite sum.

Define $m = \sum_{\ell=0}^{k} a_\ell b^\ell$ and let a be the inverse of b in arithmetic modulo m, assuming for now that $m > 0$. A major result proved in Tezuka et al. (1994), Couture and L'Ecuyer (1997), and Goresky and Klapper (2003) is that if the initial states agree, the output sequence $\{u_i, i \geq 0\}$ is exactly the same as that produced by the LCG with modulus m and multiplier a. Therefore, the MWC can be seen as a clever way of implementing a LCG with very large modulus. Couture and L'Ecuyer (1997) have shown that the value of ℓ_t for this LCG satisfies $\ell_t^2 \leq a_0^2 + \cdots + a_k^2$ for $t \geq k$, which means that the lattice structure will be bad unless the sum of squares of coefficients a_j is large.

In the original proposals of Marsaglia and Zaman (1991), called *add-with-carry* and *subtract-with-borrow*, one has $-a_0 = \pm a_r = \pm a_k = 1$ for some $r < k$ and the other coefficients a_j are zero, so $\ell_t^2 \leq 3$ for $t \geq k$ and the generator has essentially the same structural defect as the additive lagged-Fibonacci generator. In the version studied by Couture and L'Ecuyer (1997), it was assumed that $-a_0 = d = 1$. Then, the period cannot exceed $(m-1)/2$ if b is a power of two. A concrete implementation was given in that paper. Goresky and Klapper (2003) pointed out that the maximal period of $\rho = m - 1$ can be achieved by allowing a more general a_0. They provided specific parameters that give the maximal period for b ranging from 2^{21} to 2^{35} and ρ up to approximately 2^{2521}.

3.4 Generators Based on Recurrences Modulo 2

3.4.1 A General Framework

It seems natural to exploit the fact that computers work in binary arithmetic and to design RNGs defined directly in terms of bit strings and sequences. We do this under the following framework, taken from L'Ecuyer and Panneton (2002) and L'Ecuyer and Panneton (2009). Let $\mathbb{F}_2$ denote the finite field with two elements, 0 and 1, in which the operations are equivalent to addition and multiplication modulo 2. Consider the RNG defined by a matrix linear recurrence over $\mathbb{F}_2$, as follows:

$$\mathbf{x}_i = \mathbf{A}\mathbf{x}_{i-1}, \tag{3.12}$$

$$\mathbf{y}_i = \mathbf{B}\mathbf{x}_i, \tag{3.13}$$

$$u_i = \sum_{\ell=1}^{w} y_{i,\ell-1} 2^{-\ell} = .y_{i,0}\ y_{i,1}\ y_{i,2}\ \cdots, \tag{3.14}$$

where $\mathbf{x}_i = (x_{i,0}, \ldots, x_{i,k-1})^\mathsf{T} \in \mathbb{F}_2^k$ is the k-bit *state vector* at step i, $\mathbf{y}_i = (y_{i,0}, \ldots, y_{i,w-1})^\mathsf{T} \in \mathbb{F}_2^w$ is the w-bit *output vector* at step i, k and w are positive integers, $\mathbf{A}$ is a $k \times k$ *transition matrix* with elements in $\mathbb{F}_2$, $\mathbf{B}$ is a $w \times k$ *output transformation matrix* with elements in $\mathbb{F}_2$, and $u_i \in [0, 1)$ is the *output* at step i. All operations in (3.12) and (3.13) are performed in $\mathbb{F}_2$.

It is well-known (L'Ecuyer 1994; Niederreiter 1992) that when the $\mathbf{x}_i$'s obey (3.12), for each j, the sequence $\{x_{i,j},\ i \geq 0\}$ follows the linear recurrence

$$x_{i,j} = (\alpha_1 x_{i-1,j} + \cdots + \alpha_k x_{i-k,j}) \bmod 2, \tag{3.15}$$

whose *characteristic polynomial* $P(z)$ is the characteristic polynomial of $\mathbf{A}$, i.e.,

$$P(z) = \det(\mathbf{A} - z\mathbf{I}) = z^k - \alpha_1 z^{k-1} - \cdots - \alpha_{k-1} z - \alpha_k,$$

where $\mathbf{I}$ is the identity matrix and each α_j is in $\mathbb{F}_2$. The sequences $\{y_{i,j},\ i \geq 0\}$, for $0 \leq j < w$, also obey the same recurrence (although some of them may follow recurrences of shorter order as well, depending on $\mathbf{B}$). We assume that $\alpha_k = 1$, so that the recurrence (3.15) has *order* k and is purely periodic. Its period is $2^k - 1$ (i.e., maximal) if and only if $P(z)$ is a primitive polynomial over $\mathbb{F}_2$ (Knuth 1998; Niederreiter 1992).

To jump ahead directly from $\mathbf{x}_i$ to $\mathbf{x}_{i+\nu}$ with this type of generator, it suffices to precompute the matrix $\mathbf{A}^\nu$ (in $\mathbb{F}_2$) and then multiply $\mathbf{x}_i$ by this matrix.

Several popular classes of RNGs fit this framework as special cases, by appropriate choices of the matrices $\mathbf{A}$ and $\mathbf{B}$. This includes the Tausworthe or LFSR, polynomial LCG, GFSR, twisted GFSR, Mersenne twister, WELL, xorshift, multiple recursive matrix generators, and combinations of these (L'Ecuyer and Panneton

2009; Matsumoto and Nishimura 1998; Niederreiter 1995; Panneton and L'Ecuyer 2005; Panneton et al. 2006; Tezuka 1995). We detail some of them after discussing measures of uniformity.

3.4.2 Measures of Uniformity

The uniformity of point sets Ψ_I produced by RNGs based on linear recurrences over $\mathbb{F}_2$ is usually assessed by measures of equidistribution defined as follows (L'Ecuyer 1996b, 2004; L'Ecuyer and Panneton 2002, 2009; Tezuka 1995). For an arbitrary vector $\mathbf{q} = (q_1, \ldots, q_t)$ of non-negative integers, partition the unit hypercube $[0,1)^t$ into 2^{q_j} intervals of the same length along axis j, for each j. This determines a partition of $[0,1)^t$ into $2^{q_1+\cdots+q_t}$ rectangular boxes of the same size and shape. We call this partition the $\mathbf{q}$-equidissection of the unit hypercube.

For some index set $I = \{i_1, \ldots, i_t\}$, if Ψ_I has 2^k points, we say that Ψ_I is $\mathbf{q}$-*equidistributed in base 2* if there are exactly 2^q points in each box of the $\mathbf{q}$-equidissection, where $k - q = q_1 + \cdots + q_t$. This means that among the 2^k points $(x_{j_1}, \ldots, x_{j_t})$ of Ψ_I, if we consider the first q_1 bits of x_{j_1}, the first q_2 bits of x_{j_2}, ..., and the first q_t bits of x_{j_t}, each of the 2^{k-q} possibilities occurs exactly the same number of times. This is possible only if $q \le k$.

The $\mathbf{q}$-equidistribution of Ψ_I depends only on the first q_j bits of x_{i_j} for $1 \le j \le t$, for the points $(x_{i_1}, \ldots, x_{i_t})$ that belong to Ψ_I. The vector of these $q_1 + \cdots + q_t = k - q$ bits can always be expressed as a linear function of the k bits of the initial state $\mathbf{x}_0$, i.e., as $\mathbf{M_q x}_0$ for some $(k-q) \times k$ binary matrix $\mathbf{M_q}$, and it is easily seen that Ψ_I is $\mathbf{q}$-equidistributed if and only if $\mathbf{M_q}$ has full rank $k - q$. This provides an easy way of checking equidistribution (L'Ecuyer 1996b; L'Ecuyer and Panneton 2009; Tezuka 1995).

If Ψ_I is $(\ell, \ldots, \ell)$-equidistributed for some $\ell \ge 1$, it is called t-*distributed with* ℓ *bits of accuracy*, or (t, ℓ)-*equidistributed* (L'Ecuyer 1996b). The largest value of ℓ for which this holds is called the *resolution* of the set Ψ_I and is denoted by ℓ_I. This value has the upper bound $\ell_t^* = \min(\lfloor k/t \rfloor, w)$. The *resolution gap* of Ψ_I is defined as $\delta_I = \ell_t^* - \ell_I$. In the same vein as for MRGs, a worst-case figure of merit can be defined here by

$$\Delta_{\mathcal{J}} = \max_{I \in \mathcal{J}} \delta_I,$$

where $\mathcal{J}$ is a preselected class of index sets I.

The point set Ψ_I is a (q, k, t)-*net in base 2* (often called a (t, m, s)-net in the context of quasi-Monte Carlo methods, where a different notation is used Niederreiter 1992), if it is $(q_1, \ldots, q_t)$-equidistributed in base 2 for all non-negative integers $q_1, \ldots, q_t$ summing to $k - q$. We call the smallest such q the q-*value* of Ψ_I. The smaller it is, the better. One candidate for a figure of merit could be the q-value of Ψ_t for some large t. Although widely used to construct and evaluate low-discrepancy point sets for quasi-Monte Carlo methods, a major drawback of

this measure is that it is too costly to compute for good long-period generators (for which $k-q$ is large), because there are too many vectors $\mathbf{q}$ for which equidistribution needs to be checked. In practice, one must settle for figures of merit that involve a smaller number of equidissections.

When $\delta_I = 0$ for all sets I of the form $I = \{0, \ldots, t-1\}$, for $1 \le t \le k$, the RNG is said to be *maximally equidistributed* or *asymptotically random* for the word size w (L'Ecuyer 1996b; Tezuka 1995; Tootill et al. 1973). This property ensures perfect equidistribution of all sets Ψ_t, for any partition of the unit hypercube into subcubes of equal sizes, as long as $\ell \le w$ and the number of subcubes does not exceed the number of points in Ψ_t. Large-period maximally equidistributed generators, together with their implementations, can be found in L'Ecuyer (1999c), L'Ecuyer and Panneton (2002), Panneton and L'Ecuyer (2004), and Panneton et al. (2006), for example.

3.4.3 Lattice Structure in Spaces of Polynomials and Formal Series

The RNGs defined via (3.12)–(3.14) do not have a lattice structure in the real space like MRGs, but they do have a lattice structure in a space of formal series, as explained in Couture and L'Ecuyer (2000), L'Ecuyer (2004), L'Ecuyer and Panneton (2009), Lemieux and L'Ecuyer (2003), and Tezuka (1995). The real space $\mathbb{R}$ is replaced by the space $\mathbb{L}_2$ of formal power series with coefficients in $\mathbb{F}_2$, of the form $\sum_{\ell=\omega}^{\infty} x_\ell z^{-\ell}$ for some integer ω. In that setting, the lattices have the form

$$\mathcal{L}_t = \left\{ \mathbf{v}(z) = \sum_{j=1}^{t} h_j(z)\mathbf{v}_j(z) \text{ such that each } h_j(z) \in \mathbb{F}_2[z] \right\},$$

where $\mathbb{F}_2[z]$ is the ring of polynomials with coefficients in $\mathbb{F}_2$, and the basis vectors $\mathbf{v}_j(z)$ are in $\mathbb{L}_2^t$. The elements of the dual lattice $\mathcal{L}_t^*$ are the vectors $\mathbf{h}(z)$ in $\mathbb{L}_2^t$ whose scalar product with any vector of $\mathcal{L}_t$ belongs to $\mathbb{F}_2[z]$. We define the mapping $\varphi : \mathbb{L}_2 \to \mathbb{R}$ by

$$\varphi\left(\sum_{\ell=\omega}^{\infty} x_\ell z^{-\ell}\right) = \sum_{\ell=\omega}^{\infty} x_\ell 2^{-\ell}.$$

Then, it turns out that the point set Ψ_t produced by the generator is equal to $\varphi(\mathcal{L}_t) \cap [0,1)^t$. Moreover, the equidistribution properties examined in Sect. 3.4.2 can be expressed in terms of lengths of shortest vectors in the dual lattice, with appropriate definitions of the length (or norm). Much of the theory and algorithms developed for lattices in the real space can be adapted to these new types of lattices (Couture and L'Ecuyer 2000; L'Ecuyer et al. 2009; Tezuka 1995).

3.4.4 The LFSR Generator

The *Tausworthe* or *linear feedback shift register* (LFSR) generator (L'Ecuyer 1996b; Tausworthe 1965; Tezuka 1995) is a special case of (3.12–3.14) with $\mathbf{A} = \mathbf{A}_0^s$ (in $\mathbb{F}_2$) for some positive integer s, where

$$\mathbf{A}_0 = \begin{pmatrix} & 1 & & \\ & & \ddots & \\ & & & 1 \\ a_k & a_{k-1} & \dots & a_1 \end{pmatrix}, \tag{3.16}$$

$a_1, \dots, a_k$ are in $\mathbb{F}_2$, $a_k = 1$, and all blank entries in the matrix are zeros. If $w \le k$, the matrix $\mathbf{B}$ contains the first w lines of the $k \times k$ identity matrix, otherwise $\mathbf{B}$ is constructed as explained in L'Ecuyer and Panneton (2009). The RNG thus obtained can be defined equivalently by

$$x_i = a_1 x_{i-1} + \cdots + a_k x_{i-k} \bmod 2, \tag{3.17}$$

$$u_i = \sum_{\ell=1}^{w} x_{is+\ell-1} 2^{-\ell}. \tag{3.18}$$

Here, $P(z)$ is the characteristic polynomial of the matrix $\mathbf{A}_0^s$, not the characteristic polynomial of the recurrence (3.17), and the choice of s is important for determining the quality of the generator. A frequently encountered case is when a single a_j is nonzero in addition to a_k; then, $P(z)$ is a trinomial and we have a *trinomial-based* LFSR generator. These generators are known to have important statistical deficiencies (Matsumoto and Kurita 1996; Tezuka 1995) but they can be used a components of combined RNGs (Sect. 3.4.6).

LFSR generators can be expressed as LCGs in a space of polynomials (L'Ecuyer 1994; Tezuka 1995; Tezuka and L'Ecuyer 1991). With this representation, their lattice structure as discussed in Sect. 3.4.3 follows immediately.

3.4.5 The GFSR and Twisted GFSR

Here we take $\mathbf{A}$ as the $pq \times pq$ matrix

$$\mathbf{A} = \begin{pmatrix} & & \mathbf{I}_p & & \mathbf{S} \\ \mathbf{I}_p & & & & \\ & \mathbf{I}_p & & & \\ & & \ddots & & \\ & & & \mathbf{I}_p & \end{pmatrix}$$

for some positive integers p and q, where $\mathbf{I}_p$ is the $p \times p$ identity matrix, $\mathbf{S}$ is a $p \times p$ matrix, and the matrix $\mathbf{I}_p$ on the first line is in columns $(r-1)p+1$ to rp for some positive integer r. Often, $w = p$ and $\mathbf{B}$ contains the first w lines of the $pq \times pq$ identity matrix. If $\mathbf{S}$ is also the identity matrix, the generator thus obtained is the trinomial-based *generalized feedback shift register* (GFSR), for which $\mathbf{x}_i$ is obtained by a bitwise exclusive-or of $\mathbf{x}_{i-r}$ and $\mathbf{x}_{i-q}$. This gives a very fast RNG, but its period cannot exceed $2^q - 1$, because each bit of $\mathbf{x}_i$ follows the same binary recurrence of order $k = q$, with characteristic polynomial $P(z) = z^q - z^{q-r} - 1$. It also fails several simple empirical tests (L'Ecuyer and Simard 2007).

More generally, we can define $\mathbf{x}_i$ as the bitwise exclusive-or of $\mathbf{x}_{i-r_1}, \mathbf{x}_{i-r_2}, \ldots, \mathbf{x}_{i-r_d}$ where $r_d = q$, so that each bit of $\mathbf{x}_i$ follows a recurrence in $\mathbb{F}_2$ whose characteristic polynomial $P(z)$ has $d+1$ nonzero terms. However, the period is still bounded by $2^q - 1$, whereas considering the pq-bit state, we should rather expect a period close to 2^{pq}. This was the main motivation for the *twisted GFSR* (TGFSR) generator. In the original version introduced by Matsumoto and Kurita (1992), $w = p$ and the matrix $\mathbf{S}$ is defined as the transpose of $\mathbf{A}_0$ in (3.16), with k replaced by p. The characteristic polynomial of $\mathbf{A}$ is then $P(z) = P_S(z^q + z^m)$, where $P_S(z) = z^p - a_p z^{p-1} - \cdots - a_1$ is the characteristic polynomial of S, and its degree is $k = pq$. If the parameters are selected so that $P(z)$ is primitive over $\mathbb{F}_2$, then the TGFSR has period $2^k - 1$. Matsumoto and Kurita (1994) pointed out important weaknesses of the original TGFSR and proposed an improved version that uses a well-chosen matrix $\mathbf{B}$ whose lines differ from those of the identity. The operations implemented by this matrix are called *tempering* and their purpose is to improve the uniformity of the points produced by the RNG.

The *Mersenne twister* (Matsumoto and Nishimura 1998; Nishimura 2000) is a variant of the TGFSR where k is slightly less than pq and can be a prime number. A specific instance named MT19937, proposed by Matsumoto and Nishimura (1998), has become quite popular; it runs very fast and has the huge period of $2^{19937} - 1$. However, its state $\mathbf{x}_i$ occupies a large amount of memory (19,937 bits) and changes very slowly as a function of i. Panneton et al. (2006) showed that as a consequence of this slow change, if the generator starts in a state with very few bits equal to 1, then the average output values over the next few thousand steps is likely to be much less than 1/2. In particular, if the initial state has a single bit at 1, say randomly selected, then we need about 3/4 million steps before the average output value gets close to 1/2. Likewise, if two initial states differ by a single bit, it takes the same number of steps before the corresponding outputs differ by about half of their bits. This problem is related to the fact that the characteristic polynomial $P(z)$ has too few nonzero coefficients, namely 135 out of 19,938.

Panneton et al. (2006) went on to develop a class of $\mathbb{F}_2$-linear generators called *well-equidistributed long-period linear* (WELL), which run almost as fast as MT19937, but whose state changes faster and whose polynomial $P(z)$ contains nearly 50% nonzero coefficients. They propose specific instances with periods ranging from $2^{512} - 1$ to $2^{44,497} - 1$, which are all almost (or exactly) maximally equidistributed.

In the *multiple recursive matrix method* of Niederreiter (1995), the first row of $p \times p$ matrices in $\mathbf{A}$ contains arbitrary matrices. However, a fast implementation is possible only when these matrices are sparse and have a special structure.

3.4.6 Combined Linear Generators Over $\mathbb{F}_2$

Many of the best generators based on linear recurrences over $\mathbb{F}_2$ are constructed by combining the outputs of two or more RNGs having a simple structure. The idea is the same as for MRGs: select simple components that can run fast but such that their combination has a more complicated structure and highly-uniform sets Ψ_I for the sets I considered important.

Consider J distinct recurrences of the form (3.12–3.13), where the jth recurrence has parameters $(k, w, \mathbf{A}, \mathbf{B}) = (k_j, w, \mathbf{A}_j, \mathbf{B}_j)$ and state $\mathbf{x}_{j,i}$ at step i, for $j = 1, \ldots, J$. The output of the combined generator at step i is defined by

$$\mathbf{y}_i = \mathbf{B}_1\mathbf{x}_{1,i} \oplus \cdots \oplus \mathbf{B}_J\mathbf{x}_{J,i},$$
$$u_i = \sum_{\ell=1}^{w} y_{i,\ell-1}2^{-\ell},$$

where $\oplus$ denotes the bitwise exclusive-or operation. One can show (Tezuka 1995) that the period ρ of this combined generator is the least common multiple of the periods ρ_j of its components. Moreover, this combined generator is equivalent to the generator (3.12–3.14) with $k = k_1 + \cdots + k_J$, $\mathbf{A} = \text{diag}(\mathbf{A}_1, \ldots, \mathbf{A}_J)$, and $\mathbf{B} = (\mathbf{B}_1, \ldots, \mathbf{B}_J)$.

With this method, by selecting the parameters carefully, the combination of LFSR generators with characteristic polynomials $P_1(z), \ldots, P_J(z)$ gives yet another LFSR with characteristic polynomial $P(z) = P_1(z) \cdots P_J(z)$ and period equal to the product of the periods of the components (L'Ecuyer 1996b; Tezuka 1995; Tezuka and L'Ecuyer 1991; Wang and Compagner 1993). Tables and fast implementations of maximally equidistributed combined LFSR generators are given in L'Ecuyer (1999c).

The TGFSR and Mersenne twister generators cannot be maximally equidistributed. However, concrete examples of maximally equidistributed combined TGFSR generators with periods near 2^{466} and 2^{1250} can be found in L'Ecuyer and Panneton (2002). These generators have the additional property that the resolution gaps δ_I are zero for a class of small sets I with indices not too far apart.

3.5 Nonlinear RNGs

All RNGs discussed so far are based on linear recurrences and their structure may be deemed too regular. For example, we saw earlier that the output binary sequence $\{y_{i,j},\ i \geq 0\}$ of any $\mathbb{F}_2$-linear generator obeys the linear recurrence (3.15). This can

be detected easily by applying statistical tests that measure the linear complexity of this output sequence, or that construct "random" binary matrices from this sequence and compute their ranks (L'Ecuyer and Simard 2007). Because of the linear dependences between the bits, the linear complexity and the matrix ranks will be smaller than what they should be on average. For the great majority of Monte Carlo applications, this linearity is not a problem, because the random numbers are transformed nonlinearly by the simulation algorithm. But for the rare situations where it may matter, we need alternatives.

There are several ways of getting rid of the regular linear structure, including: (1) use a nonlinear transition function f; (2) keep the transition function linear but use a nonlinear output function g; (3) combine two linear RNGs of different types, such as an MRG with an $\mathbb{F}_2$-linear generator; (4) shuffle (randomly permute) the output values using another generator. Several types of genuinely nonlinear RNGs have been proposed over the years; see for example Blum et al. (1986), Eichenauer-Herrmann (1995), Eichenauer-Herrmann et al. (1998), Hellekalek and Wegenkittl (2003), Knuth (1998), L'Ecuyer and Proulx (1989), L'Ecuyer (1994), L'Ecuyer and Simard (2007), Niederreiter and Shparlinski (2002), and Tezuka (1995). Their nonlinear mappings are defined in various ways by multiplicative inversion in a finite field, quadratic and cubic functions in the finite ring of integers modulo m, and other more complicated transformations. Many of them have output sequences that tend to behave much like i.i.d. $U(0, 1)$ sequences even over their entire period length, in contrast with "good" linear RNGs, whose point sets Ψ_t are much more regular than typical random points (Eichenauer-Herrmann et al. 1998; L'Ecuyer and Granger-Piché 2003; L'Ecuyer and Hellekalek 1998; Niederreiter and Shparlinski 2002). On the other hand, their statistical properties have been analyzed only empirically or via asymptotic theoretical results. For specific nonlinear RNGs, the uniformity of the point sets Ψ_t is very difficult to measure theoretically. Moreover, the nonlinear RNGs are generally significantly slower than the linear ones. The RNGs recommended for cryptology are all nonlinear.

An interesting idea for adding nonlinearity without incurring an excessive speed penalty is to combine a small nonlinear generator with a fast long-period linear one (Aiello et al. 1998). L'Ecuyer and Granger-Piché (2003) show how to do this while ensuring theoretically the good uniformity properties of Ψ_t for the combined generator. A fast implementation can be achieved by using precomputed tables for the nonlinear component. Empirical studies suggest that mixed linear-nonlinear combined generators are more robust than the linear ones with respect to statistical tests, because of their less regular structure.

Several authors have proposed various ways of combining RNGs to produce streams of random numbers with less regularity and better "randomness" properties; see, e.g., Collings (1987), Knuth (1998), Gentle (2003), Law and Kelton (2000), L'Ecuyer (1994), Fishman (1996), Marsaglia (1985), and other references given there. This includes *shuffling* the output sequence of one generator using another one (or the same one), alternating between several streams, or just adding them in different ways. Most of these techniques are heuristics. They usually improve the uniformity (empirically), but they can also make it worse. For *random variables*

in the mathematical sense, certain types of combinations (e.g., addition modulo 1) can *provably* improve the uniformity, and some authors have used this fact to argue that combined RNGs are provably better than their components alone (Brown and Solomon 1979; Deng and George 1990; Gentle 2003; Marsaglia 1985), but this argument is faulty because the output sequences of RNGs are deterministic, not sequences of independent random variables. To assess the quality of a combined generator, one must analyze the mathematical structure of the combined generator itself rather than the structure of its components (L'Ecuyer 1996a,b, 1998; L'Ecuyer and Granger-Piché 2003; Tezuka 1995).

3.6 Empirical Statistical Tests

As mentioned earlier, a statistical test for RNGs is defined by a random variable X whose distribution under $\mathcal{H}_0$ can be well approximated. When X takes the value x, we define the right and left *p-values* of the test by

$$p_{\mathrm{R}} = P[X \geq x \mid \mathcal{H}_0] \quad \text{and} \quad p_{\mathrm{L}} = P[X \leq x \mid \mathcal{H}_0].$$

When testing RNGs, there is no need to prespecify the level of the test. If either of the right or left p-value is extremely close to zero, e.g., less than 10^{-15}, then it is clear that $\mathcal{H}_0$ (and the RNG) must be rejected. When a *suspicious* p-value is obtained, e.g., near 10^{-2} or 10^{-3}, one can just repeat this particular test a few more times, perhaps with a larger sample size. Almost always, things will then clarify.

Most tests are defined by partitioning the possible realizations of $(u_0, \ldots, u_{\tau-1})$ into a finite number of subsets (where the integer τ can be random or deterministic), computing the probability p_j of each subset j under $\mathcal{H}_0$, and measuring the discrepancy between these probabilities and empirical frequencies from realizations simulated by the RNG.

A special case that immediately comes to mind is to take $\tau = t$ (a constant) and cut the interval $[0, 1)$ into d equal segments for some positive integer d, in order to partition the hypercube $[0, 1)^t$ into $k = d^t$ subcubes of volume $1/k$. We then generate n points $\mathbf{u}_i = (u_{ti}, \ldots, u_{ti+t-1}) \in [0, 1)^t$, for $i = 0, \ldots, n-1$, and count the number N_j of points falling in subcube j, for $j = 0, \ldots, k-1$. Any measure of distance (or divergence) between the numbers N_j and their expectations n/k can define a test statistic X. The tests thus defined are generally called *serial tests* of uniformity (Knuth 1998; L'Ecuyer et al. 2002). They can be *sparse* (if $n/k \ll 1$), or *dense* (if $n/k \gg 1$), or somewhere in between. There are also *overlapping* versions, where the points are defined by $\mathbf{u}_i = (u_i, \ldots, u_{i+t-1})$ for $i = 0, \ldots, n-1$ (they have overlapping coordinates).

Special instances for which the distribution under $\mathcal{H}_0$ is well-known are the chi-square, the (negative) empirical entropy, and the number of collisions (L'Ecuyer and Hellekalek 1998; L'Ecuyer et al. 2002; Read and Cressie 1988). For the latter, the test statistic X is the number of times a point falls in a subcube that already

had a point in it. Its distribution under $\mathcal{H}_0$ is approximately Poisson with mean $\lambda_1 = n^2/(2k)$, if n is large and λ_1 not too large.

A variant is the *birthday spacings* test, defined as follows (Knuth 1998; L'Ecuyer and Simard 2001; Marsaglia 1985). Let $I_{(1)} \le \cdots \le I_{(n)}$ be the numbers of the subcubes that contain the points, sorted by increasing order. Define the *spacings* $S_j = I_{(j+1)} - I_{(j)}$, for $j = 1, \ldots, n-1$, and let X be the number of collisions between these spacings. Under $\mathcal{H}_0$, X is approximately Poisson with mean $\lambda_2 = n^3/(4k)$, if n is large and λ_2 not too large.

Consider now a MRG, for which Ψ_t has a regular lattice structure. Because of this regularity the points of Ψ_t will tend to be more evenly distributed among the subcubes than random points. For a well-chosen k and large enough n, we expect the collision test to detect this: it is likely that there will be too few collisions. In fact, the same applies to any RNG whose set Ψ_t is very evenly distributed. When a birthday spacings test with a very large k is applied to a MRG, the numbers of the subcubes that contain one point of Ψ_t tend to be too evenly spaced and the test detects this by finding too many collisions.

These specific interactions between the test and the structure of the RNG lead to systematic patterns in the p-values of the tests. To illustrate this, suppose that we take k slightly larger than the cardinality of Ψ_t (so $k \approx \rho$) and that due to the excessive regularity, no collision is observed in the collision test. The left p-value will then be $p_{\mathrm{L}} \approx P[X \le 0 \mid X \sim \mathrm{Poisson}(\lambda_1)] = \exp[-n^2/(2k)]$. For this p-value to be smaller than a given ϵ, we need a sample size n proportional to the square root of the period ρ. And after that, p_{L} decreases exponentially fast in n^2.

Extensive experiments with LCGs, MRGs, and LFSR generators confirms that this is actually what happens with these RNGs (L'Ecuyer 2001; L'Ecuyer and Hellekalek 1998; L'Ecuyer et al. 2002). For example, if we take $\epsilon = 10^{-15}$ and define n_0 as the minimal sample size n for which $p_{\mathrm{L}} < \epsilon$, we find that $n_0 \approx 16\rho^{1/2}$ (plus some noise) for LCGs that behave well in the spectral test as well as for LFSR generators. For the birthday spacings test, the rule for LCGs is $n_0 \approx 16\rho^{1/3}$ instead (L'Ecuyer and Simard 2001). So to be safe with respect to these tests, the period ρ must be so large that generating more than $\rho^{1/3}$ numbers is practically unfeasible. This certainly disqualifies all LCGs with modulus smaller than 2^{100} or so.

Other types of tests for RNGs include tests based on the closest pairs of points among n points generated in the hypercube, tests based on random walks on the real line or over the integers, tests based on the linear complexity of a binary sequence, tests based on the simulation of dice or poker hands, and many others (Gentle 2003; Knuth 1998; L'Ecuyer and Simard 2007; Marsaglia 1996; Rukhin et al. 2001; Vattulainen et al. 1995).

When testing RNGs, there is no specific alternative hypothesis to $\mathcal{H}_0$. Different tests are needed to detect different types of departures from $\mathcal{H}_0$. The *TestU01* library of L'Ecuyer and Simard (2007) implements a large collection of tests in the C language, and also provides specific test suites with preselected parameters and sample sizes. Some of these suites are designed for i.i.d. $U(0, 1)$ output sequences

and others for strings of bits. Other (smaller) test suites for RNGs are DIEHARD (Marsaglia 1996) and the NIST suite (Rukhin et al. 2001).

3.7 Available Software and Recommendations

Applying standard statistical test suites to RNGs found in popular software (statistical and simulation software, spreadsheets, system libraries, etc.) reveals that many of them are surprisingly poor and fail the tests spectacularly (L'Ecuyer 2001; L'Ecuyer and Simard 2007). There is no good reason to use these poor RNGs, because several good ones are available that are fast, portable, and pass these test suites with flying colors.

The RNG I use most of the time is the combined MRG `MRG32k3a` from L'Ecuyer (1999a). A convenient object-oriented software package with multiple streams and substreams of random numbers, based on this generator, is described in L'Ecuyer et al. (2002) and is available in Java, C, and C++, at http://www.iro.umontreal.ca/~lecuyer. This tool has been included recently in several software products, including MATLAB, SAS, R, Arena, Automod, ns3, and many more. `MRG32k3a` is not the fastest RNG available, but it is very robust and reliable. A faster alternative is `MGR31k3p` from L'Ecuyer and Touzin (2000). Other good combined MRGs, some for 64-bit computers, are available in L'Ecuyer (1999a). Even faster ones are the combined LFSRs, Mersenne twisters, and WELL generators proposed in L'Ecuyer (1999c), L'Ecuyer and Panneton (2002), Matsumoto and Nishimura (1998), Nishimura (2000), and Panneton et al. (2006). When speed is a concern, I personally use `LFSR113` or `LFSR258` from L'Ecuyer (1999c). Software tools that provide multiple streams and substreams with most of these generators (except the ones with very large state) are available in the SSJ library (L'Ecuyer 2008).

3.8 Non-Uniform Random Variate Generation

Like for the uniform case, non-uniform variate generation often involves approximations and compromises. The first requirement is, of course, *correctness*. This does not mean that the generated random variate X must always have *exactly* the required distribution, because this would sometimes be much too costly or even impossible. But we must have a *good approximation* and, preferably, some understanding of the quality of that approximation. *Robustness* is also important: when the accuracy depends on the parameters of the distribution, it must be good *uniformly* over the entire range of parameter values that we are interested in.

The method must also be *efficient* both in terms of speed and memory usage. Often, it is possible to increase the speed by using more memory (e.g, for larger precomputed tables) or by relaxing the accuracy requirements. Some methods need

a one-time setup to compute constants and construct tables. The setup time can be significant but may be well worth spending if it is amortized by a large number of subsequent calls to the generator. For example, it makes sense to invest in a more extensive setup if we plan to make a million calls to a given generator than if we expert to make only a few calls, assuming that this investment can improve the speed of the generator sufficiently.

In general, compromises must be made between simplicity of the algorithm, quality of the approximation, robustness with respect to the distribution parameters, and efficiency (generation speed, memory requirements, and setup time).

In many situations, compatibility with variance reduction techniques is another important issue (Asmussen and Glynn 2007; Bratley et al. 1987; Law and Kelton 2000). We may be willing to sacrifice the speed of the generator to preserve inversion, because the gain in efficiency obtained via the variance reduction methods may more than compensate (sometimes by orders of magnitude) for the slightly slower generator.

3.8.1 Inversion

The inversion method, defined in the introduction, should be the method of choice for generating non-uniform random variates in a majority of situations. The fact that $X = F^{-1}(U)$ is a monotone (non-decreasing) function of U makes this method compatible with important variance reductions techniques such as common random numbers, antithetic variates, Latin hypercube sampling, and randomized quasi-Monte Carlo methods (Bratley et al. 1987; Law and Kelton 2000; L'Ecuyer and Lemieux 2000; L'Ecuyer et al. 2009).

For some distributions, an analytic expression can be obtained for the inverse distribution function F^{-1} and inversion can be easily implemented. As an example, consider the *Weibull* distribution function with parameters $\alpha > 0$ and $\beta > 0$, defined by $F(x) = 1 - \exp[-(x/\beta)^{\alpha}]$ for $x > 0$. It is easy to see that $F^{-1}(U) = \beta[-\ln(1-U)]^{1/\alpha}$. For $\alpha = 1$, we have the special case of the exponential distribution with mean β.

For an example of a simple discrete distribution, suppose that $P[X = i] = p_i$ where $p_0 = 0.6$, $p_1 = 0.3$, $p_2 = 0.1$, and $p_i = 0$ elsewhere. The inversion method in this case will return 0 if $U < 0.6$, 1 if $0.6 \le U < 0.9$, and 2 if $U \ge 0.9$. For the discrete uniform distribution over $\{0, \ldots, k-1\}$, return $X = \lfloor kU \rfloor$. As another example, let X have the *geometric* distribution with parameter p, so $P[X = x] = p(1-p)^x$ for $x = 0, 1, 2, \ldots$, where $0 < p < 1$. Then, $F(x) = 1 - (1-p)^{\lfloor x+1 \rfloor}$ for $x \ge 0$ and one can show that $X = F^{-1}(U) = \lceil \ln(1-U)/\ln(1-p) \rceil - 1$.

For other distributions (e.g., the normal, Student, chi-square) there is no closed-form expression for F^{-1} but good numerical approximations are available (Bratley et al. 1987; Gentle 2003; Hörmann et al. 2004; Marsaglia et al. 1994). When the distribution has only scale and location parameters, we need to approximate F^{-1} only for a standardized version of the distribution. For the normal distribution, for

example, it suffices to have an efficient method for evaluating the inverse distribution function of a $N(0, 1)$ random variable Z, since a normal with mean μ and variance σ^2 can be generated by $X = \sigma Z + \mu$.

When shape parameters are involved (e.g., the gamma and beta distributions), things are more complicated because F^{-1} then depends on the parameters in a more fundamental manner.

Hörmann and Leydold (2003) propose a general adaptive and automatic method that constructs a highly accurate Hermite interpolation method of F^{-1}. In a one-time setup, their method produces tables for the interpolation points and coefficients. Random variate generation using these tables is then quite fast.

A less efficient but simpler way of implementing inversion when a method is available for computing F is via binary search (Cheng 1998). If the density is also available and if it is unimodal with known mode, a Newton-Raphson iteration method can advantageously replace the binary search (Cheng 1998; Devroye 1986).

To implement inversion for general discrete distributions, sequential search and binary search with look-up tables are the standard methods (Bratley et al. 1987; Cheng 1998). For a discrete distribution over the values $x_1 < \cdots < x_k$, one first tabulates the pairs $(x_i, F(x_i))$, where $F(x_i) = P[X \le x_i]$ for $i = 1, \ldots, k$. To generate X, it then suffices to generate $U \sim U(0, 1)$, find $I = \min\{i \mid F(x_i) \ge U\}$, and return $X = x_I$. The following algorithms do that.

Sequential search (needs $O(k)$ iterations in the worst case);
generate $U \sim U(0, 1)$; let $i = 1$;
while $F(x_i) < U$ do $i = i + 1$;
return x_i.

Binary search (needs $O(\log k)$ iterations in the worst case);
generate $U \sim U(0, 1)$; let $L = 0$ and $R = k$;
while $L < R - 1$ do
$m = \lfloor (L + R)/2 \rfloor$;
if $F(x_m) < U$ then $L = m$ else $R = m$;
/* Invariant: at this stage, the index I is in $\{L + 1, \ldots, R\}$. */
return x_R.

These algorithms can be modified in many different ways. For example, if $k = \infty$, in the binary search, one can start with an arbitrary value of R, double it until $F(x_R) \ge U$, and start the algorithm with this R and $L = R/2$. Of course, only a finite portion of the table (a portion that contains most of the probability mass) would be precomputed in this case, the other values can be computed only when needed. This can also be done if k is finite but large.

Another class of techniques use indexing or buckets to speed up the search (Bratley et al. 1987; Chen and Asau 1974; Devroye 1986). For example, one can partition the interval $(0, 1)$ into c subintervals of equal sizes and use (pre-tabulated) initial values of (L, R) that depend on the subinterval in which U falls. For the subinterval $[j/c, (j + 1)/c)$ the values of L and R would be $L_j = F^{-1}(j/c)$ and $R_j = F^{-1}((j + 1)/c)$, for $j = 0, \ldots, c - 1$. The subinterval number that

corresponds to a given U is simply $J = \lfloor cU \rfloor$. Once we know that subinterval, we can search it by linear of binary search. With a larger value of c the search is faster (on the average) but the setup is more costly and a larger amount of memory is needed. So a compromise must be made depending on the situation (e.g., the value of k, the number of variates we expect to generate, etc.). For $c = 1$, we recover the basic sequential and binary search algorithms given above. A well-implemented indexed search with a large enough c is competitive with the alias method (described in the next paragraph). A combined indexed/binary search algorithm is given below. An easy adaptation gives the combined indexed/sequential search, which is generally preferable when k/c is small, because it has smaller overhead.

Indexed search (combined with binary search);
generate $U \sim U(0,1)$; let $J = \lfloor cU \rfloor$, $L = L_J$, and $R = R_J$;
while $L < R - 1$ do
$m = \lfloor (L + R)/2 \rfloor$;
if $F(x_m) < U$ then $L = m$ else $R = m$;
return x_R.

These search methods are also useful for piecewise-linear (or piecewise-polynomial) distribution functions. Essentially, it suffices to add an interpolation step at the end of the algorithm, after the appropriate linear (or polynomial) piece has been determined (Bratley et al. 1987).

Finally, the stochastic model itself can sometimes be selected in a way that makes inversion easier. For example, one can fit a parametric, highly-flexible, and easily computable inverse distribution function F^{-1} to the data, directly or indirectly (Nelson and Yamnitsky 1998).

There are situations where *speed* is important and where non-inversion methods are appropriate. In forthcoming subsections, we outline the main non-inversion methods.

3.8.2 The Alias Method

Sequential and binary search require $O(k)$ and $O(\log k)$ time, respectively, in the worst case, to generate a random variate X by inversion over the finite set $\{x_1, \ldots, x_k\}$. The *alias method* (Walker 1977) can generate such a X in $O(1)$ time per variate, after a table setup that takes $O(k)$ time and space. On the other hand, it does not implement inversion, i.e., the transformation from U to X is not monotone.

To explain the idea, consider a bar diagram of the distribution, where each index i has a bar of height $p_i = P[X = x_i]$. The idea is to "equalize" the bars so that they all have height $1/k$, by cutting-off bar pieces and transferring them to other bars. This is done in a way that in the new diagram, each bar i contains one piece of size q_i (say) from the original bar i and one piece of size $1/k - q_i$ from another bar whose index j, denoted $A(i)$, is called the *alias* value of i. The setup procedure initializes

two tables, A and R, where $A(i)$ is the alias value of i and $R(i) = (i-1)/k + q_i$. See Devroye (1986) and Law and Kelton (2000) for the details. To generate X, we generate $U \sim U[0,1]$, define $I = \lceil kU \rceil$, and return $X = x_I$ if $U < R(I)$ and $X = x_{A(I)}$ otherwise.

There is a version of the alias method for continuous distributions, called the *acceptance-complement* method (Devroye 1986; Gentle 2003; Kronmal and Peterson 1984). The idea is to decompose the density f of the target distribution as the convex combination of two densities f_1 and f_2, $f = wf_1 + (1-w)f_2$ for some real number $w \in (0,1)$, in a way that $wf_1 \le g$ for some other density g and so that it is easy to generate from g and f_2. The algorithm works as follows: Generate X from density g and $U \sim U(0,1)$; if $Ug(X) \le wf_1(Y)$ return X, otherwise generate a new X from density f_2 and return it.

3.8.3 Kernel Density Estimation and Generation

Instead of selecting a parametric distribution that is hard to invert and estimating the parameters, one can estimate the density via a *kernel density estimation* method for which random variate generation is very easy (Devroye 1986; Hörmann et al. 2004). In the case of a Gaussian kernel, for example, on can generate variates simply by selecting one observation at random from the data and adding random noise generated form a normal distribution with mean zero. However, this method *is not* equivalent to inversion. Because of the added noise, selecting a larger observation does not necessarily guarantee a larger value for the generated variate.

3.8.4 The Rejection Method

Suppose we want to generate X from a complicated density f. In fact f may be known only up to a multiplicative constant $\kappa > 0$, i.e., we know only κf. If we know f, we may just take $\kappa = 1$. We select another density r such that $\kappa f(x) \le t(x) \stackrel{\text{def}}{=} ar(x)$ for all x for some constant a, and such that generating variates Y from the density r is easy. The function t is called a *hat function* or *majorizing function*. By integrating this inequality with respect to x on both sides, we find that $\kappa \le a$. The following *rejection* method generates X with density f (Devroye 1986; Evans and Swartz 2000; von Neumann 1951):

Rejection method;
repeat
  generate Y from the density r and $U \sim U(0,1)$, independent;
until $Ut(Y) \le \kappa f(Y)$;
return $X = Y$.

The number R of turns into the "repeat" loop is one plus a geometric random variable with parameter κ/a, so $E[R] = a/\kappa$. Thus, we want $a/\kappa \ge 1$ to be as

small as possible, i.e., we want to minimize the area between κf and t. There is generally a compromise between bringing a/κ close to 1 and keeping r simple.

When κf is expensive to compute, we can also use *squeeze functions* q_1 and q_2 that are faster to evaluate and such that $q_1(x) \le \kappa f(x) \le q_2(x) \le t(x)$ for all x. To verify the condition $Ut(Y) \le \kappa f(Y)$, we first check if $Ut(Y) \le q_1(Y)$, in which case we accept Y immediately, otherwise we check if $Ut(Y) \ge q_2(Y)$, in which case we reject Y immediately. The value of $\kappa f(Y)$ must be computed only when $Ut(Y)$ falls between the two squeezes. Sequences of embedded squeezes can also be used, where the primary ones are the least expensive to compute, the secondary ones are a little more expensive but closer to κf, etc.

It is common practice to transform the density f by a smooth increasing function T (e.g., $T(x) = \log x$ or $T(x) = -x^{-1/2}$) selected so that it is easier to construct good hat and squeeze functions (often piecewise linear) for the transformed density $T(f(\cdot))$. By transforming back to the original scale, we get hat and squeeze functions for f. This is the *transformed density rejection* method, which has several variants and extensions (Devroye 1986; Evans and Swartz 2000; Hörmann et al. 2004).

The rejection method works for discrete distributions as well; it suffices to replace densities by probability mass functions.

3.8.5 Thinning for Point Processes with Time-Varying Rates

Thinning is a cousin of acceptance-rejection, often used for generating events from a non-homogeneous Poisson process. Suppose the process has rate $\lambda(t)$ at time t, with $\lambda(t) \le \bar{\lambda}$ for all t, where $\bar{\lambda}$ is a finite constant. One can generate Poisson *pseudo-arrivals* at constant rate $\bar{\lambda}$ by generating interarrival times that are i.i.d. exponentials of mean $1/\bar{\lambda}$. Then, a pseudo-arrival at time t is accepted (becomes an arrival) with probability $\lambda(t)/\bar{\lambda}$ (i.e., if $U \le \lambda(t)/\bar{\lambda}$, where U is an independent $U[0,1]$), and rejected with probability $1 - \lambda(t)/\bar{\lambda}$. Non-homogeneous Poisson processes can also be generated by inversion (Bratley et al. 1987). The idea is to apply a nonlinear transformation to the time scale to make the process homogeneous with rate 1 in the new time scale. Arrival times are generated in this new time scale (which is easy), and then transformed back to the original time scale. The method can be adapted to other types of point processes with time-varying rates.

3.8.6 The Ratio-of-Uniforms Method

If f is a density over the real-line, κ an arbitrary positive constant, and the pair (U, V) has the uniform distribution over the set

$$\mathcal{C} = \left\{(u, v) \in \mathbb{R}^2 \text{ such that } 0 \le u \le \sqrt{\kappa f(v/u)}\right\},$$

then V/U has density f (Devroye 1986; Gentle 2003; Kinderman and Monahan 1977). This interesting property can be exploited to generate X with density f: generate (U, V) uniformly over $\mathcal{C}$ and return $X = V/U$. This is the *ratio-of-uniforms* method. The key issue is how do we generate a point uniformly over $\mathcal{C}$. In the cases where this can be done efficiently, we immediately have an efficient way of generating X.

The most frequent approach for generating (U, V) uniformly over $\mathcal{C}$ is the rejection method: Define a region $\mathcal{C}_2$ that contains $\mathcal{C}$ and in which it is easy to generate a point uniformly (for example, a rectangular box or a polygonal region). To generate X, repeat: generate (U, V) uniformly over $\mathcal{C}_2$, until it belongs to $\mathcal{C}$. Then return $X = V/U$. If there is another region $\mathcal{C}_1$ contained in $\mathcal{C}$ and for which it is very fast to check if a point (U, V) is in $\mathcal{C}_1$, this $\mathcal{C}_1$ can also be used as a squeeze to accelerate the verification that the point belongs to $\mathcal{C}$. Several special cases and refinements are described in Devroye (1986), Gentle (2003), Leydold (2000), and other references given there.

3.8.7 Composition and Convolution

Suppose F is a convex combination of several distributions, i.e., $F(x) = \sum_j p_j F_j(x)$, or more generally $F(x) = \int F_y(x) dH(y)$. To generate from F, one can generate $J = j$ with probability p_j (or Y from H), then generate X from F_J (or F_Y). This method, called the *composition algorithm*, is useful for generating from *compound* distributions such as the hyperexponential or from compound Poisson processes. It is also frequently used to design specialized algorithms for generating from complicated densities. The idea is to partition the area under the complicated density into pieces, where piece j has surface p_j. To generate X, first select a piece (choose piece j with probability p_j), then draw a random point uniformly over that piece and project it to the horizontal axis. If the partition is defined so that it is fast and easy to generate from the large pieces, then X will be returned very quickly most of the time. The rejection method with a squeeze is often used to generate from some of the pieces.

A dual method to composition is the *convolution method*, which can be used when $X = Y_1 + Y_2 + \cdots + Y_n$, where the Y_i's are independent with specified distributions. With this method, one just generates the Y_i's and sum up. This requires at least n uniforms. Examples of random variables that can be expressed as sums like this include the hypoexponential, Erlang, and binomial distributions.

3.8.8 Other Special Techniques

Specialized and sometimes very elegant techniques have been designed for commonly used distributions such as the Poisson distribution with small mean, the

normal (e.g., the Box-Muller and the polar methods), for generating points uniformly on a k-dimensional sphere, for generating random permutations, and so on. Details can be found, e.g., in Bratley et al. (1987), Cheng (1998), Devroye (1986), Fishman (1996), Gentle (2003). Many of those methods are based on a clever multivariate change of variables, defined so that the random variates or random vectors in the new coordinates are much easier to generate. In the Box-Muller and Polar methods, for example, a pair of independent standard normals is generated in polar coordinates, and then transformed back into rectangular coordinates.

Recently, there has been an effort in developing *automatic* or *black box* algorithms for generating variates from an arbitrary (known) density, and reliable software that implements these methods (Hörmann and Leydold 2000; Hörmann et al. 2004; Leydold 2009).

3.8.9 Multivariate Distributions

Inversion does not directly apply to generate a d-dimensional random vector $\mathbf{X} = (X_1, \ldots, X_d)^\mathsf{T}$, because the inverse of its distribution function is not well defined. In some cases, one can generate the first coordinate X_1 by inversion from its marginal distribution, then generate X_2 by inversion from its marginal distribution conditional on X_1, then generate X_3 by inversion from its marginal distribution conditional on (X_1, X_2), and so on. But this is not always possible and convenient.

Simple and elegant methods are available for certain classes of distributions. For example, if $\mathbf{X}$ has a *multinormal distribution* with mean vector μ and covariance matrix Σ, then one can decompose $\Sigma = \mathbf{A}\mathbf{A}^\mathsf{T}$ for some matrix $\mathbf{A}$, generate a vector $\mathbf{Z}$ of d independent standard normal random variable (with mean 0 and variance 1), usually by inversion, and return $\mathbf{X} = \mu + \mathbf{A}\mathbf{Z}$. One way to decompose Σ is via the Cholesky decomposition, for which $\mathbf{A}$ is lower triangular, but there are many other possibilities, including the eigendecomposition as in principal component analysis. The choice of decomposition can have a large impact on the variance reduction in the context of randomized quasi-Monte Carlo integration, by concentrating more of the variance on the first few uniform random numbers that are generated L'Ecuyer (2009).

Multivariate normals are often used to generate vectors from other distributions. For example, to generate a random point on a d-dimensional sphere of radius r centered at zero, one can generate a vector $\mathbf{Z}$ of independent standard normals (this amounts to generating a random direction), then normalize its length to r. More generally, by putting $\mathbf{X} = R\,\mathbf{Z}/\|\mathbf{Z}\|$ where R has an arbitrary distribution over $(0, \infty)$, one generates a vector with a *radially symmetric* distribution. As a special case, if R has the Student distribution, $\mathbf{X}$ is multivariate Student. As a further generalization, let $\mathbf{X} = \mu + R\,\mathbf{A}\mathbf{Z}/\|\mathbf{Z}\|$ where $\mathbf{Z}$ is multinormal in k dimensions and $\mathbf{A}$ is a $d \times k$ matrix. This $\mathbf{X}$ has an *elliptic multivariate* distribution.

A richer class of multivariate distributions are defined via *copula methods* (Asmussen and Glynn 2007; Hörmann et al. 2004; Nelsen 1999). Start with an arbitrary d-dimensional distribution function G with continuous marginals G_j, generate $\mathbf{Y} = (Y_1, \ldots, Y_d)^\mathsf{T}$ from G, and let $\mathbf{U} = (U_1, \ldots, U_d) = (G_1(Y_1), \ldots, G_d(Y_d))^\mathsf{T}$. These U_j have the uniform distribution over $(0, 1)$, but they are not independent in general. The distribution function of $\mathbf{U}$ is the *copula associated with* G and it specifies the dependence structure of the vector $\mathbf{U}$. In fact, any multivariate distribution function over $(0, 1)^d$ with uniform marginals is a copula. To generate $\mathbf{X} = (X_1, \ldots, X_d)^\mathsf{T}$ with arbitrary marginal distribution functions F_j and dependence structure specified by this copula, put $X_j = F_j^{-1}(U_j)$ for each j. A popular choice for G is the multinormal distribution with standard normal marginals; then $\mathbf{Y}$ and $\mathbf{U}$ are easy to generate, and one can select the correlation matrix of $\mathbf{Y}$ to approximate a target correlation (or rank correlation) matrix for $\mathbf{X}$. It can usually match the correlations pretty well. But to approximate the whole dependence structure in general, a much richer variety of copulas is required (Asmussen and Glynn 2007; Hörmann et al. 2004; Nelsen 1999).

The rejection method extends easily to the multivariate case. For a target d-dimensional density f known up to the multiplicative constant κ, pick a d-dimensional density r such that $\kappa f(\mathbf{x}) \le a r(\mathbf{x})$ for all $\mathbf{x}$ and some constant a, and such that sampling random vectors $\mathbf{Y}$ from r is easy. To generate $\mathbf{X}$ with density f, generate $\mathbf{Y}$ from r and U uniform over $(0, 1)$ independent of $\mathbf{Y}$, until $U a r(\mathbf{Y}) \le \kappa f(\mathbf{Y})$, and return $\mathbf{X} = \mathbf{Y}$.

There are many situations where one wishes to generate random vectors $\mathbf{X}$ from quite complicated distributions and no efficient method is available to do it exactly. One very important approach that often permit one to generate $\mathbf{X}$ approximately from the target distribution is the *Markov chain Monte Carlo* (MCMC) method. In a nutshell, it constructs an artificial Markov chain whose steady-state distribution is the target distribution of $\mathbf{X}$, and runs the Markov chain for a large number of steps, until it is deemed sufficiently close to steady-state. Then the state of the chain has a distribution close to the target one. MCMC is covered elsewhere in this handbook.

Acknowledgements This work has been supported by the Natural Sciences and Engineering Research Council of Canada (NSERC) and a Canada Research Chair to the author. Wolfgang Hörmann, Josef Leydold, François Panneton, and Richard Simard made helpful comments and corrections on an earlier draft. The author has been asked to write chapters on Random Number Generation for several handbooks and encyclopedia over the years. Inevitably, there is a large amount of duplication between these chapters.

References

Aiello, W., Rajagopalan, S., Venkatesan, R.: Design of practical and provably good random number generators. J. Algorithm. **29**(2), 358–389 (1998)

Asmussen, S., Glynn, P.W. Stochastic simulation, Springer, New York (2007)

Blum, L., Blum, M., Schub, M.: A simple unpredictable pseudo-random number generator. SIAM J. Comput. **15**(2), 364–383 (1986)
Bratley, P., Fox, B.L., Schrage, L.E.: A Guide to Simulation. (2nd edn.), Springer, New York, NY (1987)
Brown, M., Solomon, H.: On combining pseudorandom number generators. Ann. Stat. **1**, 691–695 (1979)
Chen, H.C., Asau, Y.: On generating random variates from an empirical distribution. AIEE Trans. **6**, 163–166 (1974)
Cheng, R.C.H.: Random variate generation. In: Banks, J. (eds.) Handbook of Simulation, pp. 139–172. Wiley (1998); chapter 5.
Collings, B.J.: Compound random number generators. J. Am. Stat. Assoc. **82**(398), 525–527 (1987)
Conway, J.H., Sloane, N.J.A.: Sphere packings, lattices and groups. (3rd edn.) Grundlehren der Mathematischen Wissenschaften 290. Springer, New York (1999)
Couture, R., L'Ecuyer, P.: On the lattice structure of certain linear congruential sequences related to AWC/SWB generators. Math. Comput. **62**(206), 798–808 (1994)
Couture, R., L'Ecuyer, P.: Orbits and lattices for linear random number generators with composite moduli. Math. Comput. **65**(213), 189–201 (1996)
Couture, R., L'Ecuyer, P.: Distribution properties of multiply-with-carry random number generators. Math. Comput. **66**(218), 591–607 (1997)
Couture, R., L'Ecuyer, P.: Lattice computations for random numbers. Math. Comput. **69**(230), 757–765 (2000)
Deng, L.-Y.: Efficient and portable multiple recursive generators of large order. ACM Trans. Model. Comput. Simulat. **15**(1), 1–13 (2005)
Deng, L.-Y., George, E.O.: Generation of uniform variates from several nearly uniformly distributed variables. Comm. Stat. **B19**(1), 145–154 (1990)
Deng, L.-Y., Lin, D.K.J.: Random number generation for the new century. Am. Stat. **54**(2), 145–150 (2000)
Deng, L.-Y., Xu, H.: A system of high-dimensional, efficient, long-cycle and portable uniform random number generators. ACM Trans. Model. Comput. Simulat. **13**(4), 299–309 (2003)
Devroye, L.: Non-uniform Random Variate Generation, Springer, New York, NY (1986)
Devroye, L.: Nonuniform random variate generation. In: Simulation, Henderson, S.G., Nelson, B.L. (eds.) Handbooks in Operations Research and Management Science, pp. 83–121. Elsevier, Amsterdam, Netherlands (2006); Chapter 4.
Eichenauer-Herrmann, J.: Pseudorandom number generation by nonlinear methods. Int. Stat. Rev. **63**, 247–255 (1995)
Eichenauer-Herrmann, J., Herrmann, E., Wegenkittl, S.: A survey of quadratic and inversive congruential pseudorandom numbers. In: Hellekalek, P., Larcher, G., Niederreiter, H., Zinterhof, P. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 1996, Lecture Notes in Statistics, vol. 127, pp. 66–97. Springer, New York, NY (1998)
Evans, M., Swartz, T.: Approximating Integrals via Monte Carlo and Deterministic Methods, Oxford University Press, Oxford, UK (2000)
Fincke, U., Pohst, M.: Improved methods for calculating vectors of short length in a lattice, including a complexity analysis. Math. Comput. **44**, 463–471 (1985)
Fishman, G.S.: Monte Carlo: Concepts, Algorithms, and applications. Springer Series in Operations Research. Springer, New York, NY (1996)
Gentle, J.E.: Random Number Generation and Monte Carlo methods. (2nd edn.), Springer, New York, NY (2003)
Goresky, M., Klapper, A.: Efficient multiply-with-carry random number generators with maximal period. ACM Trans. Model. Comput. Simulat. **13**(4), 310–321 (2003)
Hellekalek, P., Wegenkittl, S.: Empirical evidence concerning AES. ACM Trans. Model. Comput. Simulat. **13**(4), 322–333 (2003)

Hörmann, W., Leydold, J.: Dec. Automatic random variate generation for simulation input. In: Joines, J.A., Barton, R.R., Kang, K., Fishwick, P.A. (eds.) In: Proceedings of the 2000 Winter Simulation Conference, pp. 675–682. IEEE Press, Pistacaway, NJ (2000)
Hörmann, W., Leydold, J.: Continuous random variate generation by fast numerical inversion. ACM Trans. Model. Comput. Simulat. **13**(4), 347–362 (2003)
Hörmann, W., Leydold, J., Derflinger, G.: Automatic Nonuniform Random Variate Generation. Springer, Berlin (2004)
Kinderman, A.J., Monahan, J.F.: Computer generation of random variables using the ratio of uniform deviates. ACM Trans. Math. Software **3**, 257–260 (1977)
Knuth, D.E.: The art of Computer Programming, Seminumerical Algorithms, vol. 2, (3rd edn.) Addison-Wesley, Reading, MA (1998)
Kronmal, R.A., Peterson, A.V.: An Acceptance-complement Analogue of the Mixture-plus-acceptance-rejection Method for Generating Random Variables. ACM Trans. Math. Software **10**, 271–281 (1984)
Lagarias, J.C.: Pseudorandom numbers. Stat. Sci. **8**(1), 31–39 (1993)
Law, A.M., Kelton, W.D.: Simulation Modeling and Analysis. (3rd edn.), McGraw-Hill, New York, NY (2000)
L'Ecuyer, P.: Random numbers for simulation. Comm. ACM **33**(10), 85–97 (1990)
L'Ecuyer, P.: Uniform random number generation. Ann. Oper. Res. **53**, 77–120 (1994)
L'Ecuyer, P.: Combined multiple recursive random number generators. Oper. Res. **44**(5), 816–822 (1996a)
L'Ecuyer, P.: Maximally equidistributed combined Tausworthe generators. Math. Comput. **65**(213), 203–213 (1996b)
L'Ecuyer, P.: Bad lattice structures for vectors of non-successive values produced by some linear recurrences. INFORMS J. Comput. **9**(1), 57–60 (1997)
L'Ecuyer, P.: Random number generation. In: Banks, J. (eds.) Handbook of Simulation, pp. 93–137. Wiley (1998); chapter 4.
L'Ecuyer, P.: Good parameters and implementations for combined multiple recursive random number generators. Oper. Res. **47**(1), 159–164 (1999a)
L'Ecuyer, P.: Tables of linear congruential generators of different sizes and good lattice structure. Math. Comput. **68**(225), 249–260 (1999b)
L'Ecuyer, P.: Tables of maximally equidistributed combined LFSR generators. Math. Comput. **68**(225), 261–269 (1999c)
L'Ecuyer, P.: Software for uniform random number generation: Distinguishing the good and the bad. In: Proceedings of the 2001 Winter Simulation Conference, pp. 95–105. IEEE Press, Pistacaway, NJ (2001)
L'Ecuyer, P.: Polynomial integration lattices. In: Niederreiter, H. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2002, pp. 73–98. Springer, Berlin (2004)
L'Ecuyer, P.: SSJ: A Java library for stochastic simulation. Software user's guide, available at http://www.iro.umontreal.ca/~lecuyer (2008)
L'Ecuyer, P.: Quasi-Monte Carlo methods with applications in finance. Finance Stochast. **13**(3), 307–349 (2009)
L'Ecuyer, P., Andres, T.H.: A random number generator based on the combination of four LCGs. Math. Comput. Simul. **44**, 99–107 (1997)
L'Ecuyer, P., Blouin, F., Couture, R.: A search for good multiple recursive random number generators. ACM Trans. Model. Comput. Simulat. **3**(2), 87–98 (1993)
L'Ecuyer, P., Côté, S.: Implementing a random number package with splitting facilities. ACM Trans. Math. Software **17**(1), 98–111 (1991)
L'Ecuyer, P., Couture, R.: An implementation of the lattice and spectral tests for multiple recursive linear random number generators. INFORMS J. Comput. **9**(2), 206–217 (1997)
L'Ecuyer, P., Granger-Piché, J.: Combined generators with components from different families. Math. Comput. Simul. **62**, 395–404 (2003)

L'Ecuyer, P., Hellekalek, P.: Random number generators: Selection criteria and testing. In: Hellekalek, P., Larcher, G. (eds.) Random and Quasi-Random Point Sets, Lecture Notes in Statistics, vol. 138, pp. 223–265. Springer New York, NY (1998)
L'Ecuyer, P., Lemieux, C.: Variance reduction via lattice rules. Manag. Sci. **46**(9), 1214–1235 (2000)
L'Ecuyer, P., Lemieux, C.: Recent advances in randomized quasi-Monte Carlo methods. In: Dror, M., L'Ecuyer, P., Szidarovszky, F. (eds.) Modeling Uncertainty: An Examination of Stochastic Theory, Methods, and Applications, pp. 419–474. Kluwer Academic, Boston (2002)
L'Ecuyer, P., Mandjes, M., Tuffin, B.: Importance sampling and rare event simulation. In: Rubino, G., Tuffin, B. (eds.) Rare Event Simulation Using Monte Carlo Methods, 17–38. Wiley (2009); Chapter 2.
L'Ecuyer, P., Panneton, F.: Construction of equidistributed generators based on linear recurrences modulo 2. In: Fang, K.-T., Hickernell, F.J., Niederreiter, H. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2000, pp. 318–330. Springer, Berlin (2002)
L'Ecuyer, P., Panneton, F.: $\mathbf{F}_2$-linear random number generators. In: Alexopoulos, C., Goldsman, D.J.R. (eds.) Advancing the Frontiers of Simulation: A Festschrift in Honor of George Samuel Fishman, Wilson, pp. 169–193. Springer, New York (2009)
L'Ecuyer, P., Proulx, R.: Dec. About polynomial-time "unpredictable" generators. In: Proceedings of the 1989 Winter Simulation Conference, pp. 467–476: IEEE Press, New York (1989)
L'Ecuyer, P., Simard, R.: Beware of linear congruential generators with multipliers of the form $a = \pm 2^q \pm 2^r$. ACM Trans. Math. Software **25**(3), 367–374 (1999)
L'Ecuyer, P., Simard, R.: On the performance of birthday spacings tests for certain families of random number generators. Math. Comput. Simul. **55**(1–3), 131–137 (2001)
L'Ecuyer, P., Simard, R.: TestU01: A C library for empirical testing of random number generators. ACM Trans. Math. Software **33**(4), Article 22 (2007)
L'Ecuyer, P., Simard, R., Chen, E.J., Kelton, W.D.: An object-oriented random-number package with many long streams and substreams. Oper. Res. **50**(6), 1073–1075 (2002)
L'Ecuyer, P., Simard, R., Wegenkittl, S.: Sparse serial tests of uniformity for random number generators. SIAM J. Sci. Comput. **24**(2), 652–668 (2002)
L'Ecuyer, P., Tezuka, S.: Structural properties for two classes of combined random number generators. Math. Comput. **57**(196), 735–746 (1991)
L'Ecuyer, P., Touzin, R.: Fast combined multiple recursive generators with multipliers of the form $a = \pm 2^q \pm 2^r$. In: Joines, J.A., Barton, R.R., Kang, K., Fishwick, P.A. (eds.) Proceedings of the 2000 Winter Simulation Conference, pp. 683–689. IEEE Press, Pistacaway, NJ (2000)
L'Ecuyer, P., Touzin, R.: On the Deng-Lin random number generators and related methods. Stat. Comput. **14**, 5–9 (2004)
Leeb, H.: Random numbers for computer simulation. Master's thesis, University of Salzburg (1995)
Lemieux, C., L'Ecuyer, P.: Randomized polynomial lattice rules for multivariate integration and simulation. SIAM J. Sci. Comput. **24**(5), 1768–1789 (2003)
Leydold, J.: Automatic sampling with the ratio-of-uniform method. ACM Trans. Math. Software **26**(1), 78–98 (2000)
Leydold, J.: UNU.RAN—universal non-uniform random number generators. Available at http://statmath.wu.ac.at/unuran/ (2009)
Luby, M.: Pseudorandomness and cryptographic applications. Princeton: Princeton University Press (1996)
Lüscher, M.: A portable high-quality random number generator for lattice field theory simulations. Comput. Phys. Comm. **79**, 100–110 (1994)
Marsaglia, G.: A current view of random number generators. In Computer Science and Statistics, Sixteenth Symposium on the Interface, pp. 3–10. Elsevier Science Publishers, North-Holland, Amsterdam (1985)
Marsaglia, G.: The Marsaglia random number CDROM including the DIEHARD battery of tests of randomness. See http://stat.fsu.edu/pub/diehard (1996)

Marsaglia, G., Zaman, A.: A new class of random number generators. Ann. Appl. Probab. **1**, 462–480 (1991)
Marsaglia, G., Zaman, A., Marsaglia, J.C.W.: Rapid evaluation of the inverse normal distribution function. Stat. Probab. Lett. **19**, 259–266 (1994)
Matsumoto, M., Kurita, Y.: Twisted GFSR generators. ACM Trans. Model. Comput. Simul. **2**(3), 179–194 (1992)
Matsumoto, M., Kurita, Y.: Twisted GFSR generators II. ACM Trans. Model. Comput. Simul. **4**(3), 254–266 (1994)
Matsumoto, M., Kurita, Y.: Strong deviations from randomness in m-sequences based on trinomials. ACM Trans. Model. Comput. Simul. **6**(2), 99–106 (1996)
Matsumoto, M., Nishimura, T.: Mersenne twister: A 623-dimensionally equidistributed uniform pseudo-random number generator. ACM Trans. Model. Comput. Simul. **8**(1), 3–30 (1998)
Nelsen, R.B.: An introduction to copulas, Lecture Notes in Statistics. vol. 139, Springer, New York, NY (1999)
Nelson, B.L., Yamnitsky, M.: Input modeling tools for complex problems. In: Proceedings of the 1998 Winter Simulation Conference, pp. 105–112. IEEE Press, Piscataway, NJ (1998)
Niederreiter, H.: Random number generation and quasi-Monte Carlo methods, SIAM CBMS-NSF Regional Conference Series in Applied Mathematics. vol. 63 SIAM, Philadelphia, PA (1992)
Niederreiter, H.: The multiple-recursive matrix method for pseudorandom number generation. Finite Fields Appl. **1**, 3–30 (1995)
Niederreiter, H., Shparlinski, I.E.: Recent advances in the theory of nonlinear pseudorandom number generators. In: Fang, K.-T., Hickernell, F.J., Niederreiter, H. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2000, pp. 86–102. Springer, Berlin (2002)
Nishimura, T.: Tables of 64-bit Mersenne twisters. ACM Trans. Model. Comput. Simul. **10**(4), 348–357 (2000)
Panneton, F., L'Ecuyer, P.: Random number generators based on linear recurrences in F_{2^w}. In: Niederreiter, H. (eds.) Monte Carlo and Quasi-Monte Carlo Methods 2002, pp. 367–378. Springer, Berlin (2004)
Panneton, F., L'Ecuyer, P.: On the xorshift random number generators. ACM Trans. Model. Comput. Simul. **15**(4), 346–361 (2005)
Panneton, F., L'Ecuyer, P., Matsumoto, M.: Improved long-period generators based on linear recurrences modulo 2. ACM Trans. Math. Software **32**(1), 1–16 (2006)
Read, T.R.C., Cressie, N.A.C.: Goodness-of-fit statistics for discrete multivariate data. Springer Series in Statistics. Springer, New York, NY (1988)
Rukhin, A., Soto, J., Nechvatal, J., Smid, M., Barker, E., Leigh, S., Levenson, M., Vangel, M., Banks, D., Heckert, A., Dray, J., Vo, S.: A statistical test suite for random and pseudorandom number generators for cryptographic applications. NIST special publication 800-22, National Institute of Standards and Technology (NIST), Gaithersburg, MD, USA (2001); See http://csrc.nist.gov/rng/.
Tausworthe, R.C.: Random numbers generated by linear recurrence modulo two. Math. Comput. **19**, 201–209 (1965)
Tezuka, S.: Uniform random numbers: Theory and practice. Kluwer Academic Publishers, Norwell, MA (1995)
Tezuka, S., L'Ecuyer, P.: Efficient and portable combined Tausworthe random number generators. ACM Trans. Model. Comput. Simul. **1**(2), 99–112 (1991)
Tezuka, S., L'Ecuyer, P., Couture, R.: On the add-with-carry and subtract-with-borrow random number generators. ACM Trans. Model. Comput. Simul. **3**(4), 315–331 (1994)
Tootill, J.P.R., Robinson, W.D., Eagle, D.J.: An asymptotically random Tausworthe sequence. J. ACM **20**, 469–481 (1973)
Vattulainen, I., Ala-Nissila, T., Kankaala, K.: Physical models as tests of randomness. Phys. Rev. E **52**(3), 3205–3213 (1995)
von Neumann, J.: Various techniques used in connection with random digits. In: A.S.H. et al. (eds.) The Monte Carlo Method, vol. 12, pp. 36–38. National Bureau of Standards, Applied Mathematics Series (1951)

Walker, A.J.: An efficient method for generating discrete random variables with general distributions. ACM Trans. Math. Software **3**, 253–256 (1977)

Wang, D., Compagner, A.: On the use of reducible polynomials as random number generators. Math. Comput. **60**, 363–374 (1993)

Wegenkittl, S., Matsumoto, M.: Getting rid of correlations among pseudorandom numbers: Discarding versus tempering. ACM Trans. Model. Comput. Simul. **9**(3), 282–294 (1999)

Wu, P.-C.: Multiplicative, congruential random number generators with multiplier $\pm 2^{k_1} \pm 2^{k_2}$ and modulus $2^p - 1$. ACM Trans. Math. Software **23**(2), 255–265 (1997)

Chapter 4
Markov Chain Monte Carlo Technology

Siddhartha Chib

4.1 Introduction

In the past fifteen years computational statistics has been enriched by a powerful, somewhat abstract method of generating variates from a target probability distribution that is based on Markov chains whose stationary distribution is the probability distribution of interest. This class of methods, popularly referred to as Markov chain Monte Carlo methods, or simply MCMC methods, have been influential in the modern practice of Bayesian statistics where these methods are used to summarize the posterior distributions that arise in the context of the Bayesian prior-posterior analysis (Besag et al. 1995; Chib and Greenberg 1995; Gelfand and Smith 1990; Smith and Roberts 1993; Tanner and Wong 1987; Tierney 1994, 1996; Carlin and Louis 2000; Chen et al. 2000; Chib 2001; Congdon 2001; Gammerman 1997; Gelman et al. 2003; Gilks et al. 1996; Liu 2001; Robert 2001; Robert and Casella 1999; Tanner 1996). MCMC methods have proved useful in practically all aspects of Bayesian inference, for example, in the context of prediction problems and in the computation of quantities, such as the marginal likelihood, that are used for comparing competing Bayesian models.

A central reason for the widespread interest in MCMC methods is that these methods are extremely general and versatile and can be used to sample univariate and multivariate distributions when other methods (for example classical methods that produce independent and identically distributed draws) either fail or are difficult to implement. The fact that MCMC methods produce dependent draws causes no substantive complications in summarizing the target distribution. For example, if $\{\boldsymbol{\psi}^{(1)}, \ldots, \boldsymbol{\psi}^{(M)}\}$ are draws from a (say continuous) target distribution $\pi(\boldsymbol{\psi})$, where $\boldsymbol{\psi} \in \Re^d$, then the expectation of $h(\boldsymbol{\psi})$ under π can be estimated by the average

S. Chib (✉)
Olin Business School, Washington University in St. Louis
St. Louis, MO 63130, USA
e-mail: chib@wustl.edu

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_4,

$$M^{-1}\sum_{j=1}^{M} h\left(\boldsymbol{\psi}^{(j)}\right) \ , \tag{4.1}$$

as in the case of random samples, because suitable laws of large numbers for Markov chains can be used to show that

$$M^{-1}\sum_{j=1}^{M} h\left(\boldsymbol{\psi}^{(j)}\right) \rightarrow \int_{\Re^d} h(\boldsymbol{\psi})\pi(\boldsymbol{\psi})\mathrm{d}\boldsymbol{\psi} \ ,$$

as the simulation sample size M becomes large.

Another reason for the interest in MCMC methods is that, somewhat surprisingly, it is rather straightforward to construct one or more Markov chains whose limiting invariant distribution is the desired target distribution. One way to construct the appropriate Markov chain is by a method called the Metropolis method which was introduced by Metropolis et al. (1953) in connection with work related to the hydrogen bomb project. In this method, the Markov chain simulation is constructed by a recursive two step process. Given the current iterate $\boldsymbol{\psi}^{(j)}$, a proposal value $\boldsymbol{\psi}'$ is drawn from a distribution $q(\boldsymbol{\psi}^{(j)}, \cdot)$, such that $\boldsymbol{\psi}'$ is symmetrically distributed about the current value $\boldsymbol{\psi}^{(j)}$. In the second step, this proposal value is accepted as the next iterate $\boldsymbol{\psi}^{(j+1)}$ of the Markov chain with probability

$$\alpha\left(\boldsymbol{\psi}^{(j)}, \boldsymbol{\psi}'\right) = \min\left\{1, \frac{\pi(\boldsymbol{\psi}')}{\pi\left(\boldsymbol{\psi}^{(j)}\right)}\right\} \ .$$

If the proposal value is rejected, then $\boldsymbol{\psi}^{(j+1)}$ is taken to be the current value. The method is simple to implement, even in multivariate settings, and was widely used by physicists in computational statistical mechanics and quantum field theory to sample the coordinates of a point in phase space. In those settings, and in subsequent statistical problems, it is helpful that the method can be implemented without knowledge of the normalizing constant of π since that constant cancels in the determination of the probability $\alpha(\boldsymbol{\psi}^{(j)}, \boldsymbol{\psi}')$.

The requirement that the proposal distribution be symmetric about the current value was relaxed by Hastings (1970). The resulting method, commonly called the Metropolis–Hastings (M–H) method, relies on the same two steps of the Metropolis method except that the probability of move is given by

$$\alpha\left(\boldsymbol{\psi}^{(j)}, \boldsymbol{\psi}'\right) = \min\left\{1, \frac{\pi\left(\boldsymbol{\psi}'\right)}{\pi\left(\boldsymbol{\psi}^{(j)}\right)}\frac{q\left(\boldsymbol{\psi}', \boldsymbol{\psi}^{(j)}\right)}{q\left(\boldsymbol{\psi}^{(j)}, \boldsymbol{\psi}'\right)}\right\}$$

which clearly reduces to the Metropolis probability of move when the proposal distribution is symmetric in its arguments. Starting with an arbitrary value $\boldsymbol{\psi}^{(0)}$ in the support of the target distributions, iterations of this two step process produce the

(correlated) sequence of values

$$\{\boldsymbol{\psi}^{(0)}, \boldsymbol{\psi}^{(1)}, \boldsymbol{\psi}^{(2)}, \ldots\} \ .$$

Typically, a certain number of values (say n_0) at the start of this sequence are discarded and the subsequent (say) M values are used as variates from the target distribution.

In applications when the dimension of $\boldsymbol{\psi}$ is large it may be preferable to construct the Markov chain simulation by first grouping the variables $\boldsymbol{\psi}$ into smaller blocks. For simplicity suppose that two blocks are adequate and that $\boldsymbol{\psi}$ is written as $(\boldsymbol{\psi}_1, \boldsymbol{\psi}_2)$, with $\boldsymbol{\psi}_k \in \Omega_k \subseteq \Re^{d_k}$. In that case, the M–H chain can be constructed by:

- Updating $\boldsymbol{\psi}_1$ given $\left(\boldsymbol{\psi}_1^{(j)}, \boldsymbol{\psi}_2^{(j)}\right)$ to produce $\boldsymbol{\psi}_1^{(j)}$ and then
- Updating $\boldsymbol{\psi}_2$ given $\left(\boldsymbol{\psi}_1^{(j+1)}, \boldsymbol{\psi}_2^{(j)}\right)$ to produce $\boldsymbol{\psi}_2^{(j+1)}$,

which completes one cycle through two sub-moves. Chib and Greenberg (1995) who emphasized and highlighted such M–H chains have referred to them as multiple-block M–H algorithms.

Despite the long vintage of the M–H method, the contemporary interest in MCMC methods was sparked by work on a related MCMC method, the Gibbs sampling algorithm. The Gibbs sampling algorithm is one of the simplest Markov chain Monte Carlo algorithms and has its origins in the work of Besag (1974) on spatial lattice systems, Geman and Geman (1984) on the problem of image processing, and Tanner and Wong (1987) on missing data problems. The paper by Gelfand and Smith (1990) helped to demonstrate the value of the Gibbs algorithm for a range of problems in Bayesian analysis. In the Gibbs sampling method, the Markov chain is constructed by simulating the conditional distributions that are implied by $\pi(\boldsymbol{\psi})$. In particular, if $\boldsymbol{\psi}$ is split into two components $\boldsymbol{\psi}_1$ and $\boldsymbol{\psi}_2$, then the Gibbs method proceeds through the recursive sampling of the conditional distributions $\pi(\boldsymbol{\psi}_1|\boldsymbol{\psi}_2)$ and $\pi(\boldsymbol{\psi}_2|\boldsymbol{\psi}_1)$, where the most recent value of $\boldsymbol{\psi}_2$ is used in the first simulation and the most recent value of $\boldsymbol{\psi}_1$ in the second simulation. This method is most simple to implement when each conditional distribution is a known distribution that is easy to sample. As we show below, the Gibbs sampling method is a special case of the multiple block M–H algorithm.

4.1.1 Organization

The rest of the chapter is organized as follows. In Sect. 4.2 we summarize the relevant Markov chain theory that justifies simulation by MCMC methods. In particular, we provide the conditions under which discrete-time and continuous state space Markov chains satisfy a law of large numbers and a central limit theorem. The M–H algorithm is discussed in Sect. 4.3 followed by the Gibbs sampling algorithm

in Sect. 4.4. Section 4.5 deals with MCMC methods with latent variables and Sect. 4.6 with ways of estimating the marginal densities based on the MCMC output. Issues related to sampler performance are considered in Sect. 4.7 and strategies for improving the mixing of the Markov chains in Sect. 4.8. Section 4.9 concludes with brief comments about new and emerging directions in MCMC methods.

4.2 Markov Chains

Markov chain Monte Carlo is a method to sample a given multivariate distribution π^* by constructing a suitable Markov chain with the property that its limiting, invariant distribution, is the target distribution π^*. In most problems of interest, the distribution π^* is absolutely continuous and, as a result, the theory of MCMC methods is based on that of Markov chains on continuous state spaces outlined, for example, in Nummelin (1984) and Meyn and Tweedie (1993). Tierney (1994) is the fundamental reference for drawing the connections between this elaborate Markov chain theory and MCMC methods. Basically, the goal of the analysis is to specify conditions under which the constructed Markov chain converges to the invariant distribution, and conditions under which sample path averages based on the output of the Markov chain satisfy a law of large numbers and a central limit theorem.

4.2.1 Definitions and Results

A Markov chain is a collection of random variables (or vectors) $\boldsymbol{\Phi} = \{\boldsymbol{\Phi}_i : i \in T\}$ where $T = \{0, 1, 2, \ldots\}$. The evolution of the Markov chain on a space $\Omega \subseteq \Re^p$ is governed by the transition kernel

$$\begin{aligned} P(\boldsymbol{x}, A) &\equiv \Pr(\boldsymbol{\Phi}_{i+1} \in A | \boldsymbol{\Phi}_i = \boldsymbol{x}, \boldsymbol{\Phi}_j,\ j < i) \\ &= \Pr(\boldsymbol{\Phi}_{i+1} \in A | \boldsymbol{\Phi}_i = \boldsymbol{x}) \,, \quad \boldsymbol{x} \in \Omega \,, \quad A \subset \Omega \,, \end{aligned}$$

where the second line embodies the Markov property that the distribution of each succeeding state in the sequence, given the current and the past states, depends only on the current state.

Generally, the transition kernel in Markov chain simulations has both a continuous and discrete component. For some function $p(\boldsymbol{x}, \boldsymbol{y}) : \Omega \times \Omega \to \Re^+$, the kernel can be expressed as

$$P(\boldsymbol{x}, d\boldsymbol{y}) = p(\boldsymbol{x}, \boldsymbol{y}) d\boldsymbol{y} + r(\boldsymbol{x}) \delta_{\boldsymbol{x}}(d\boldsymbol{y}) \,, \tag{4.2}$$

where $p(\boldsymbol{x}, \boldsymbol{x}) = 0$, $\delta_{\boldsymbol{x}}(d\boldsymbol{y}) = 1$ if $\boldsymbol{x} \in d\boldsymbol{y}$ and 0 otherwise, $r(\boldsymbol{x}) = 1 - \int_\Omega p(\boldsymbol{x}, \boldsymbol{y})\, d\boldsymbol{y}$. This transition kernel specifies that transitions from $\boldsymbol{x}$ to $\boldsymbol{y}$ occur according to $p(\boldsymbol{x}, \boldsymbol{y})$ and transitions from $\boldsymbol{x}$ to $\boldsymbol{x}$ occur with probability $r(\boldsymbol{x})$.

The transition kernel is thus the distribution of $\boldsymbol{\Phi}_{i+1}$ given that $\boldsymbol{\Phi}_i = \boldsymbol{x}$. The nth step ahead transition kernel is given by

$$P^{(n)}(\boldsymbol{x}, A) = \int_{\Omega} P(\boldsymbol{x}, d\,\boldsymbol{y})\, P^{(n-1)}(\boldsymbol{y}, A) \ ,$$

where $P^{(1)}(\boldsymbol{x}, d\,\boldsymbol{y}) = P(\boldsymbol{x}, d\,\boldsymbol{y})$ and

$$P(\boldsymbol{x}, A) = \int_{A} P(\boldsymbol{x}, d\,\boldsymbol{y}) \ . \tag{4.3}$$

The goal is to find conditions under which the nth iterate of the transition kernel converges to the invariant distribution π^* as $n \to \infty$. The invariant distribution is one that satisfies

$$\pi^*(d\,\boldsymbol{y}) = \int_{\Omega} P(\boldsymbol{x}, d\,\boldsymbol{y})\pi(\boldsymbol{x})\, d\,\boldsymbol{x} \ , \tag{4.4}$$

where π is the density of π^* with respect to the Lebesgue measure. The invariance condition states that if $\boldsymbol{\Phi}_i$ is distributed according to π^*, then all subsequent elements of the chain are also distributed as π^*. Markov chain samplers are invariant by construction and therefore the existence of the invariant distribution does not have to be checked.

A Markov chain is reversible if the function $p(\boldsymbol{x}, \boldsymbol{y})$ in (4.2) satisfies

$$f(\boldsymbol{x})p(\boldsymbol{x}, \boldsymbol{y}) = f(\boldsymbol{y})p(\boldsymbol{y}, \boldsymbol{x}) \ , \tag{4.5}$$

for a density $f(\cdot)$. If this condition holds, it can be shown that $f(\cdot) = \pi(\cdot)$ and has π^* as an invariant distribution (Tierney 1994). To verify this we evaluate the right hand side of (4.4):

$$\begin{aligned}
\int P(\boldsymbol{x}, A)\pi(\boldsymbol{x})\, d\,\boldsymbol{x} &= \int \left\{ \int_A p(\boldsymbol{x}, \boldsymbol{y})\, d\,\boldsymbol{y} \right\} \pi(\boldsymbol{x})\, d\,\boldsymbol{x} + \int r(\boldsymbol{x})\delta_{\boldsymbol{x}}(A)\pi(\boldsymbol{x})\, d\,\boldsymbol{x} \\
&= \int_A \left\{ \int p(\boldsymbol{x}, \boldsymbol{y})\pi(\boldsymbol{x})\, d\,\boldsymbol{x} \right\} d\,\boldsymbol{y} + \int_A r(\boldsymbol{x})\pi(\boldsymbol{x})\, d\,\boldsymbol{x} \\
&= \int_A \left\{ \int p(\boldsymbol{y}, \boldsymbol{x})\pi(\boldsymbol{y})\, d\,\boldsymbol{x} \right\} d\,\boldsymbol{y} + \int_A r(\boldsymbol{x})\pi(\boldsymbol{x})\, d\,\boldsymbol{x} \\
&= \int_A (1 - r(\boldsymbol{y}))\pi(\boldsymbol{y})\, d\,\boldsymbol{y} + \int_A r(\boldsymbol{x})\pi(\boldsymbol{x})\, d\,\boldsymbol{x} \\
&= \int_A \pi(\boldsymbol{y})\, d\,\boldsymbol{y} \ .
\end{aligned} \tag{4.6}$$

A minimal requirement on the Markov chain for it to satisfy a law of large numbers is the requirement of π^*-irreducibility. This means that the chain is able to visit all sets with positive probability under π^* from any starting point in Ω. Formally, a Markov chain is said to be π^*-irreducible if for every $\boldsymbol{x} \in \Omega$,

$$\pi^*(A) > 0 \Rightarrow P(\boldsymbol{\Phi}_i \in A | \boldsymbol{\Phi}_0 = \boldsymbol{x}) > 0$$

for some $i \geq 1$. If the space Ω is connected and the function $p(\boldsymbol{x}, \boldsymbol{y})$ is positive and continuous, then the Markov chain with transition kernel given by (4.3) and invariant distribution π^* is π^*-irreducible.

Another important property of a chain is aperiodicity, which ensures that the chain does not cycle through a finite number of sets. A Markov chain is aperiodic if there exists no partition of $\Omega = (D_0, D_1, \ldots, D_{p-1})$ for some $p \geq 2$ such that $P(\boldsymbol{\Phi}^i \in D_{i \bmod (p)} | \boldsymbol{\Phi}_0 \in D_0) = 1$ for all i.

These definitions allow us to state the following results from Tierney (1994) which form the basis for Markov chain Monte Carlo methods. The first of these results gives conditions under which a strong law of large numbers holds and the second gives conditions under which the probability density of the Mth iterate of the Markov chain converges to its unique, invariant density.

Theorem 1. *Suppose $\{\boldsymbol{\Phi}_i\}$ is a π^*-irreducible Markov chain with transition kernel $P(\cdot, \cdot)$ and invariant distribution π^*, then π^* is the unique invariant distribution of $P(\cdot, \cdot)$ and for all π^*-integrable real-valued functions h,*

$$\frac{1}{M} \sum_{i=1}^{M} h(\boldsymbol{\Phi}_i) \rightarrow \int h(\boldsymbol{x}) \pi(\boldsymbol{x}) d\boldsymbol{x} \quad as \quad M \rightarrow \infty \ , \ a.s.$$

Theorem 2. *Suppose $\{\boldsymbol{\Phi}_i\}$ is a π^*-irreducible, aperiodic Markov chain with transition kernel $P(\cdot, \cdot)$ and invariant distribution π^*. Then for π^*-almost every $\boldsymbol{x} \in \Omega$, and all sets A*

$$\parallel P^M(\boldsymbol{x}, A) - \pi^*(A) \parallel \rightarrow 0 \quad as \quad M \rightarrow \infty \ ,$$

where $\parallel \cdot \parallel$ denotes the total variation distance.

A further strengthening of the conditions is required to obtain a central limit theorem for sample-path averages. A key requirement is that of an ergodic chain, i.e., chains that are irreducible, aperiodic and positive Harris-recurrent (for a definition of the latter, see Tierney (1994)). In addition, one needs the notion of geometric ergodicity. An ergodic Markov chain with invariant distribution π^* is a geometrically ergodic if there exists a non-negative real-valued function (bounded in expectation under π^*) and a positive constant $r < 1$ such that

$$\parallel P^M(\boldsymbol{x}, A) - \pi^*(A) \parallel \leq C(\boldsymbol{x}) r^n$$

for all $\boldsymbol{x}$ and all n and sets A. Chan and Ledolter (1995) show that if the Markov chain is ergodic, has invariant distribution π^*, and is geometrically ergodic, then for all L^2 measurable functions h, taken to be scalar-valued for simplicity, and any initial distribution, the distribution of $\sqrt{M}(\hat{h}_M - \mathrm{E}h)$ converges weakly to a normal distribution with mean zero and variance $\sigma_h^2 \geq 0$, where

$$\hat{h}_M = \frac{1}{M}\sum_{i=1}^{M} h(\boldsymbol{\Phi}_i)$$

$$\mathrm{E}h = \int h(\boldsymbol{\Phi})\pi(\boldsymbol{\Phi})d\boldsymbol{\Phi}$$

and

$$\sigma_h^2 = \mathrm{Var}\, h(\boldsymbol{\Phi}_0) + 2\sum_{k=1}^{\infty} \mathrm{Cov}\ [\{h(\boldsymbol{\Phi}_0), h(\boldsymbol{\Phi}_k)\}]\ . \tag{4.7}$$

4.2.2 *Computation of Numerical Accuracy and Inefficiency Factor*

The square root of σ_h^2 is the numerical standard error of $\hat{h}_M$. To describe estimators of σ_h^2 that are consistent in M, let $Z_i = h(\boldsymbol{\Phi}_i)$ $(i \leq M)$. Then, due to the fact that $\{Z_i\}$ is a dependent sequence

$$\begin{aligned}
\mathrm{Var}\left(\hat{h}_M\right) &= M^{-2}\sum_{j,k} \mathrm{Cov}\,(Z_j, Z_k) \\
&= s^2 M^{-2}\sum_{j,k=1}^{M} \rho_{|j-k|} \\
&= s^2 M^{-1}\left\{1 + 2\sum_{s=1}^{M}\left(1 - \frac{s}{M}\right)\rho_s\right\}\ ,
\end{aligned}$$

where s^2 is the sample variance of $\{Z_i\}$ and ρ_s is the estimated autocorrelation at lag s (see Ripley 1987, Chap. 6). If $\rho_s > 0$ for each s, then this variance is larger than s^2/M which is the variance under independence. Another estimate of the variance can be found by consistently estimating the spectral density f of $\{Z_i\}$ at frequency zero and using the fact that $\mathrm{Var}\,(\hat{h}_M) = \tau^2/M$, where $\tau^2 = 2\pi f(0)$. Finally, a traditional approach to finding the variance is by the method of batch means. In this approach, the data $(Z_1, \ldots, Z_M)$ is divided into k batches of length m with means $B_i = m^{-1}[Z_{(i-1)m+1} + \ldots + Z_{im}]$ and the variance of $\hat{h}_M$ estimated as

$$\text{Var}\left(\hat{h}_M\right) = \frac{1}{k(k-1)} \sum_{i=1}^{k} \left(B_i - \bar{B}\right)^2 , \tag{4.8}$$

where the batch size m is chosen to ensure that the first order serial correlation of the batch means is less than 0.05.

Given the numerical variance it is common to calculate the inefficiency factor, which is also called the autocorrelation time, defined as

$$\kappa_{\hat{h}} = \frac{\text{Var}\left(\hat{h}_M\right)}{s^2/M} . \tag{4.9}$$

This quantity is interpreted as the ratio of the numerical variance of $\hat{h}_M$ to the variance of $\hat{h}_M$ based on independent draws, and its inverse is the relative numerical efficiency defined in Geweke (1992). Because independence sampling produces an autocorrelation time that is theoretically equal to one and Markov chain sampling produces autocorrelation times that are bigger than one, the inefficiency factor serves to quantify the relative efficiency loss in the computation of $\hat{h}_M$ from correlated versus independent samples.

4.3 Metropolis–Hastings Algorithm

This powerful algorithm provides a general approach for producing a correlated sequence of draws from the target density that may be difficult to sample by a classical independence method. The goal is to simulate the d-dimensional distribution $\pi^*(\boldsymbol{\psi})$, $\boldsymbol{\psi} \in \Psi \subseteq \Re^d$ that has density $\pi(\boldsymbol{\psi})$ with respect to some dominating measure. To define the algorithm, let $q(\boldsymbol{\psi}, \boldsymbol{\psi}')$ denote a source density for a candidate draw $\boldsymbol{\psi}'$ given the current value ψ in the sampled sequence. The density $q(\boldsymbol{\psi}, \boldsymbol{\psi}')$ is referred to as the proposal or candidate generating density. Then, the M–H algorithm is defined by two steps: a first step in which a proposal value is drawn from the candidate generating density and a second step in which the proposal value is accepted as the next iterate in the Markov chain according to the probability $\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}')$, where

$$\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}') = \begin{cases} \min\left[\dfrac{\pi(\boldsymbol{\psi}')q(\boldsymbol{\psi}', \boldsymbol{\psi})}{\pi(\boldsymbol{\psi})q(\boldsymbol{\psi}, \boldsymbol{\psi}')}, 1\right] & \text{if} \quad \pi(\boldsymbol{\psi})q(\boldsymbol{\psi}, \boldsymbol{\psi}') > 0 ; \\ 1 & \text{otherwise} . \end{cases} \tag{4.10}$$

If the proposal value is rejected, then the next sampled value is taken to be the current value. In algorithmic form, the simulated values are obtained by the following recursive procedure.

Algorithm 1 Metropolis–Hastings

1. Specify an initial value $\boldsymbol{\psi}^{(0)}$:
2. Repeat for $j = 1, 2, \ldots, M$.

   (a) Propose
   $$\boldsymbol{\psi}' \sim q\left(\boldsymbol{\psi}^{(j)}, \cdot\right)$$

   (b) Let
   $$\boldsymbol{\psi}^{(j+1)} = \begin{cases} \boldsymbol{\psi}' & \text{if} \quad Unif(0,1) \leq \alpha\left(\boldsymbol{\psi}^{(j)}, \boldsymbol{\psi}'\right) \ ; \\ \boldsymbol{\psi}^{(j)} & \text{otherwise} \ . \end{cases}$$

3. Return the values $\left\{\boldsymbol{\psi}^{(1)}, \boldsymbol{\psi}^{(2)}, \ldots, \boldsymbol{\psi}^{(M)}\right\}$.

Typically, a certain number of values (say n_0) at the start of this sequence are discarded after which the chain is assumed to have converged to it invariant distribution and the subsequent draws are taken as approximate variates from π. Because theoretical calculation of the burn-in is not easy it is important that the proposal density is chosen to ensure that the chain makes large moves through the support of the invariant distribution without staying at one place for many iterations. Generally, the empirical behavior of the M–H output is monitored by the autocorrelation time of each component of $\boldsymbol{\psi}$ and by the acceptance rate, which is the proportion of times a move is made as the sampling proceeds.

One should observe that the target density appears as a ratio in the probability $\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}')$ and therefore the algorithm can be implemented without knowledge of the normalizing constant of $\pi(\cdot)$. Furthermore, if the candidate-generating density is symmetric, i.e. $q(\boldsymbol{\psi}, \boldsymbol{\psi}') = q(\boldsymbol{\psi}', \boldsymbol{\psi})$, the acceptance probability only contains the ratio $\pi(\boldsymbol{\psi}')/\pi(\boldsymbol{\psi})$; hence, if $\pi(\boldsymbol{\psi}') \geq \pi(\boldsymbol{\psi})$, the chain moves to $\boldsymbol{\psi}'$, otherwise it moves with probability given by $\pi(\boldsymbol{\psi}')/\pi(\boldsymbol{\psi})$. The latter is the algorithm originally proposed by Metropolis et al. (1953). This version of the algorithm is illustrated in Fig. 4.1.

Different proposal densities give rise to specific versions of the M–H algorithm, each with the correct invariant distribution π. One family of candidate-generating densities is given by $q(\boldsymbol{\psi}, \boldsymbol{\psi}') = q(\boldsymbol{\psi}' - \boldsymbol{\psi})$. The candidate $\boldsymbol{\psi}'$ is thus drawn according to the process $\boldsymbol{\psi}' = \boldsymbol{\psi} + z$, where z follows the distribution q. Since the candidate is equal to the current value plus noise, this case is called a random walk M–H chain. Possible choices for q include the multivariate normal density and the multivariate-t. The random walk M–H chain is perhaps the simplest version of the M–H algorithm (and was the one used by Metropolis et al. 1953) and popular in applications. One has to be careful, however, in setting the variance of z; if it is too large it is possible that the chain may remain stuck at a particular value for many iterations while if it is too small the chain will tend to make small moves and move inefficiently through the support of the target distribution. In both cases the generated draws that will be highly serially correlated. Note that when q is symmetric, $q(z) = q(-z)$ and the probability of move only contains the

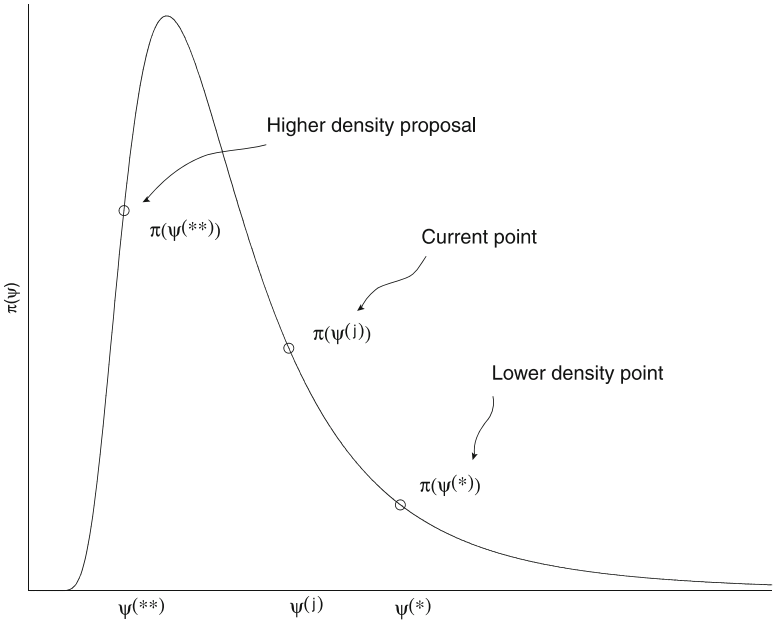


Fig. 4.1 Original Metropolis algorithm: higher density proposal is accepted with probability one and the lower density proposal with probability α

ratio $\pi(\boldsymbol{\psi}')/\pi(\boldsymbol{\psi})$. As mentioned earlier, the same reduction occurs if $q(\boldsymbol{\psi}, \boldsymbol{\psi}') = q(\boldsymbol{\psi}', \boldsymbol{\psi})$.

Hastings (1970) considers a second family of candidate-generating densities that are given by the form $q(\boldsymbol{\psi}, \boldsymbol{\psi}') = q(\boldsymbol{y})$. Tierney (1994) refers to this as an independence M–H chain because, in contrast to the random walk chain, the candidates are drawn independently of the current location $\boldsymbol{\psi}$. In this case, the probability of move becomes

$$\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}') = \min\left\{\frac{w(\boldsymbol{\psi}')}{w(\boldsymbol{\psi})}, 1\right\} \ ;$$

where $w(\boldsymbol{\psi}) = \pi(\boldsymbol{\psi})/q(\boldsymbol{\psi})$ is the ratio of the target and proposal densities. For this method to work and not get stuck in the tails of π, it is important that the proposal density have thicker tails than π. A similar requirement is placed on the importance sampling function in the method of importance sampling (Geweke 1989). In fact, Mengersen and Tweedie (1996) show that if $w(\boldsymbol{\psi})$ is uniformly bounded then the resulting Markov chain is ergodic.

Chib and Greenberg (1994, 1995) discuss a way of formulating proposal densities in the context of time series autoregressive-moving average models that has a bearing on the choice of proposal density for the independence M–H chain. They suggest matching the proposal density to the target at the mode by a multivariate normal or

multivariate-t distribution with location given by the mode of the target and the dispersion given by inverse of the Hessian evaluated at the mode. Specifically, the parameters of the proposal density are taken to be

$$\begin{aligned} \boldsymbol{m} &= \arg\max \log \pi(\boldsymbol{\psi}) \quad \text{and} \\ \boldsymbol{V} &= \tau \left\{ -\frac{\partial^2 \log \pi(\boldsymbol{\psi})}{\partial \boldsymbol{\psi}\, \partial \boldsymbol{\psi}'} \right\}^{-1}_{\boldsymbol{\psi}=\hat{\psi}} , \end{aligned} \tag{4.11}$$

where τ is a tuning parameter that is adjusted to control the acceptance rate. The proposal density is then specified as $q(\boldsymbol{\psi}') = f(\boldsymbol{\psi}'|\boldsymbol{m}, \boldsymbol{V})$, where f is some multivariate density. This may be called a tailored M–H chain.

Another way to generate proposal values is through a Markov chain version of the accept-reject method. In this version, due to Tierney (1994), and considered in detail by Chib and Greenberg (1995), a pseudo accept-reject step is used to generate candidates for an M–H algorithm. Suppose $c > 0$ is a known constant and $h(\boldsymbol{\psi})$ a source density. Let $C = \{\boldsymbol{\psi} : \pi(\boldsymbol{\psi}) \leq ch(\boldsymbol{\psi})\}$ denote the set of value for which $ch(\boldsymbol{\psi})$ dominates the target density and assume that this set has high probability under π^*. Given $\boldsymbol{\psi}^{(n)} = \boldsymbol{\psi}$, the next value $\boldsymbol{\psi}^{(n+1)}$ is obtained as follows: First, a candidate value $\boldsymbol{\psi}'$ is obtained, independent of the current value $\boldsymbol{\psi}$, by applying the accept-reject algorithm with $ch(\cdot)$ as the "pseudo dominating" density. The candidates $\boldsymbol{\psi}'$ that are produced under this scheme have density $q(\boldsymbol{\psi}') \propto \min\{\pi(\boldsymbol{\psi}'), ch(\boldsymbol{\psi}')\}$. If we let $w(\boldsymbol{\psi}) = c^{-1}\pi(\boldsymbol{\psi})/h(\boldsymbol{\psi})$ then it can be shown that the M–H probability of move is given by

$$\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}') = \begin{cases} 1 & \text{if} \quad \boldsymbol{\psi} \in C \\ 1/w(\boldsymbol{\psi}) & \text{if} \quad \boldsymbol{\psi} \notin C\,, \quad \boldsymbol{\psi}' \in C \\ \min\{w(\boldsymbol{\psi}')/w(\boldsymbol{\psi}), 1\} & \text{if} \quad \boldsymbol{\psi} \notin C\,, \quad \boldsymbol{\psi}' \notin C \end{cases} . \tag{4.12}$$

4.3.1 Convergence Results

In the M–H algorithm the transition kernel of the chain is given by

$$P(\boldsymbol{\psi}, d\boldsymbol{\psi}') = q(\boldsymbol{\psi}, \boldsymbol{\psi}')\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}')\, d\boldsymbol{\psi}' + r(\boldsymbol{\psi})\delta_{\boldsymbol{\psi}}(d\boldsymbol{\psi}') , \tag{4.13}$$

where $\delta_{\boldsymbol{\psi}}(d\boldsymbol{\psi}') = 1$ if $\boldsymbol{\psi} \in d\boldsymbol{\psi}'$ and 0 otherwise and

$$r(\boldsymbol{\psi}) = 1 - \int_{\Omega} q(\boldsymbol{\psi}, \boldsymbol{\psi}')\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}')\, d\boldsymbol{\psi}' .$$

Thus, transitions from $\boldsymbol{\psi}$ to $\boldsymbol{\psi}'$ ($\boldsymbol{\psi}' \neq \boldsymbol{\psi}$) are made according to the density

$$p(\boldsymbol{\psi}, \boldsymbol{\psi}') \equiv q(\boldsymbol{\psi}, \boldsymbol{\psi}')\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}'), \quad \boldsymbol{\psi} \neq \boldsymbol{\psi}'$$

while transitions from $\boldsymbol{\psi}$ to $\boldsymbol{\psi}$ occur with probability $r(\boldsymbol{\psi})$. In other words, the density function implied by this transition kernel is of mixed type,

$$K(\boldsymbol{\psi}, \boldsymbol{\psi}') = q(\boldsymbol{\psi}, \boldsymbol{\psi}')\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}') + r(\boldsymbol{\psi})\delta_{\boldsymbol{\psi}}(\boldsymbol{\psi}') \,, \tag{4.14}$$

having both a continuous and discrete component, where now, with change of notation, $\delta_{\boldsymbol{\psi}}(\boldsymbol{\psi}')$ is the Dirac delta function defined as $\delta_{\boldsymbol{\psi}}(\boldsymbol{\psi}') = 0$ for $\boldsymbol{\psi}' \neq \boldsymbol{\psi}$ and $\int_{\Omega} \delta_{\boldsymbol{\psi}}(\boldsymbol{\psi}')d\boldsymbol{\psi}' = 1$.

Chib and Greenberg (1995) provide a way to derive and interpret the probability of move $\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}')$. Consider the proposal density $q(\boldsymbol{\psi}, \boldsymbol{\psi}')$. This proposal density q is not likely to be reversible for π (if it were then we would be done and M–H sampling would not be necessary). Without loss of generality, suppose that $\pi(\boldsymbol{\psi})q(\boldsymbol{\psi}, \boldsymbol{\psi}') > \pi(\boldsymbol{\psi}')q(\boldsymbol{\psi}', \boldsymbol{\psi})$ implying that the rate of transitions from $\boldsymbol{\psi}$ to $\boldsymbol{\psi}'$ exceed those in the reverse direction. To reduce the transitions from $\boldsymbol{\psi}$ to $\boldsymbol{\psi}'$ one can introduce a function $0 \leq \alpha(\boldsymbol{\psi}, \boldsymbol{\psi}') \leq 1$ such that $\pi(\boldsymbol{\psi})q(\boldsymbol{\psi}, \boldsymbol{\psi}')\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}') = \pi(\boldsymbol{\psi}')q(\boldsymbol{\psi}', \boldsymbol{\psi})$. Solving for $\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}')$ yields the probability of move in the M–H algorithm. This calculation reveals the important point that the function $p(\boldsymbol{\psi}, \boldsymbol{\psi}') = q(\boldsymbol{\psi}, \boldsymbol{\psi}')\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}')$ is reversible by construction, i.e., it satisfies the condition

$$q(\boldsymbol{\psi}, \boldsymbol{\psi}')\alpha(\boldsymbol{\psi}, \boldsymbol{\psi}')\pi(\boldsymbol{\psi}) = q(\boldsymbol{\psi}', \boldsymbol{\psi})\alpha(\boldsymbol{\psi}', \boldsymbol{\psi})\pi(\boldsymbol{\psi}') \,. \tag{4.15}$$

It immediately follows, therefore, from the argument in (4.6) that the M–H kernel has $\pi(\boldsymbol{\psi})$ as its invariant density.

It is not difficult to provide conditions under which the Markov chain generated by the M–H algorithm satisfies the conditions of Propositions 1–2. The conditions of Proposition 1 are satisfied by the M–H chain if $q(\boldsymbol{\psi}, \boldsymbol{\psi}')$ is positive for $(\boldsymbol{\psi}, \boldsymbol{\psi}')$ and continuous and the set $\boldsymbol{\psi}$ is connected. In addition, the conditions of Proposition 2 are satisfied if q is not reversible (which is the usual situation) which leads to a chain that is aperiodic. Conditions for ergodicity, required for use of the central limit theorem, are satisfied if in addition π is bounded. Other similar conditions are provided by Robert and Casella (1999).

4.3.2 Example

To illustrate the M–H algorithm, consider the binary response data in Table 4.1, taken from Fahrmeir and Tutz (1997), on the occurrence or non-occurrence of infection following birth by caesarean section. The response variable y is one if the caesarean birth resulted in an infection, and zero if not. There are three covariates: x_1, an indicator of whether the caesarean was non-planned; x_2, an indicator of whether risk factors were present at the time of birth and x_3, an indicator of whether

Table 4.1 Caesarean infection data

Y (1/0)	x_1	x_2	x_3
11/87	1	1	1
1/17	0	1	1
0/2	0	0	1
23/3	1	1	0
28/30	0	1	0
0/9	1	0	0
8/32	0	0	0

antibiotics were given as a prophylaxis. The data in the table contains information from 251 births. Under the column of the response, an entry such as 11/87 means that there were 98 deliveries with covariates $(1, 1, 1)$ of whom 11 developed an infection and 87 did not.

Suppose that the probability of infection for the ith birth ($i \leq 251$) is

$$\Pr(y_i = 1|\boldsymbol{x}_i, \boldsymbol{\beta}) = \Phi(\boldsymbol{x}_i'\boldsymbol{\beta}) \,, \tag{4.16}$$

$$\boldsymbol{\beta} \sim N_4(0, 5\boldsymbol{I}_4) \,, \tag{4.17}$$

where $\boldsymbol{x}_i = (1, x_{i1}, x_{i2}, x_{i3})^\top$ is the covariate vector, $\boldsymbol{\beta} = (\beta_0, \beta_1, \beta_2, \beta_3)$ is the vector of unknown coefficients, Φ is the cdf of the standard normal random variable and $\boldsymbol{I}_4$ is the four-dimensional identity matrix. The target posterior density, under the assumption that the outcomes $\boldsymbol{y} = (y_1, y_2, \ldots, y_{251})$ are conditionally independent, is

$$\pi(\boldsymbol{\beta}|\boldsymbol{y}) \propto \pi(\boldsymbol{\beta}) \prod_{i=1}^{251} \Phi\left(\boldsymbol{x}_i^\top \boldsymbol{\beta}\right)^{y_i} \left\{1 - \Phi\left(\boldsymbol{x}_i^\top \boldsymbol{\beta}\right)\right\}^{(1-y_i)} \,,$$

where $\pi(\boldsymbol{\beta})$ is the density of the $N(0, 10\boldsymbol{I}_4)$ distribution.

Random Walk Proposal Density

To define the proposal density, let

$$\hat{\boldsymbol{\beta}} = (-1.093022 \;\; 0.607643 \;\; 1.197543 \;\; -1.904739)^\top$$

be the MLE found using the Newton–Raphson algorithm and let

$$\boldsymbol{V} = \begin{pmatrix} 0.040745 & -0.007038 & -0.039399 & 0.004829 \\ & 0.073101 & -0.006940 & -0.050162 \\ & & 0.062292 & -0.016803 \\ & & & 0.080788 \end{pmatrix}$$

Table 4.2 Caesarean data: Prior-posterior summary based on 5000 draws (beyond a burn-in of 100 cycles) from the random-walk M–H algorithm

	Prior		Posterior			
	Mean	Std dev	Mean	Std dev	Lower	Upper
β_0	0.000	3.162	−1.110	0.224	−1.553	−0.677
β_1	0.000	3.162	0.612	0.254	0.116	1.127
β_2	0.000	3.162	1.198	0.263	0.689	1.725
β_3	0.000	3.162	−1.901	0.275	−2.477	−1.354

be the symmetric matrix obtained by inverting the negative of the Hessian matrix (the matrix of second derivatives) of the log-likelihood function evaluated at $\hat{\boldsymbol{\beta}}$. Now generate the proposal values by the random walk:

$$\begin{aligned} \boldsymbol{\beta} &= \boldsymbol{\beta}^{(j-1)} + \boldsymbol{\varepsilon}^{(j)} \\ \boldsymbol{\varepsilon}^{(j)} &\sim \mathrm{N}_4(\mathbf{0}, \boldsymbol{V}) \ , \end{aligned} \tag{4.18}$$

which leads to the original Metropolis method. From a run of 5000 iterations of the algorithm beyond a burn-in of a 100 iterations we get the prior-posterior summary that is reported in Table 4.2, which contains the first two moments of the prior and posterior and the 2.5th (lower) and 97.5th (upper) percentiles of the marginal densities of $\boldsymbol{\beta}$.

As expected, both the first and second covariates increase the probability of infection while the third covariate (the antibiotics prophylaxis) reduces the probability of infection.

To get an idea of the form of the posterior density we plot in Fig. 4.1 the four marginal posterior densities. The density plots are obtained by smoothing the histogram of the simulated values with a Gaussian kernel. In the same plot we also report the autocorrelation functions (correlation against lag) for each of the sampled parameter values. The autocorrelation plots provide information of the extent of serial dependence in the sampled values. Here we see that the serial correlations start out high but decline to almost zero by lag twenty.

Tailored Proposal Density

To see the difference in results, the M–H algorithm is next implemented with a tailored proposal density. In this scheme one utilizes both $\hat{\boldsymbol{\beta}}$ and $\boldsymbol{V}$ that were defined above. We let the proposal density be $f_T(\boldsymbol{\beta}|\hat{\boldsymbol{\beta}}, \boldsymbol{V}, 15)$, a multivariate-$t$ density with fifteen degrees of freedom. This proposal density is similar to the random-walk proposal except that the distribution is centered at the fixed point $\hat{\boldsymbol{\beta}}$. The prior-posterior summary based on 5,000 draws of the M–H algorithm with this proposal density is given in Table 4.3. We see that the marginal posterior moments are similar to those in Table 4.1. The marginal posterior densities are

Table 4.3 Caesarean data: Prior-posterior summary based on 5,000 draws (beyond a burn-in of 100 cycles) from the tailored M–H algorithm

	Prior		Posterior			
	Mean	Std dev	Mean	Std dev	Lower	Upper
β_0	0.000	3.162	−1.080	0.220	−1.526	−0.670
β_1	0.000	3.162	0.593	0.249	0.116	1.095
β_2	0.000	3.162	1.181	0.254	0.680	1.694
β_3	0.000	3.162	−1.889	0.266	−2.421	−1.385

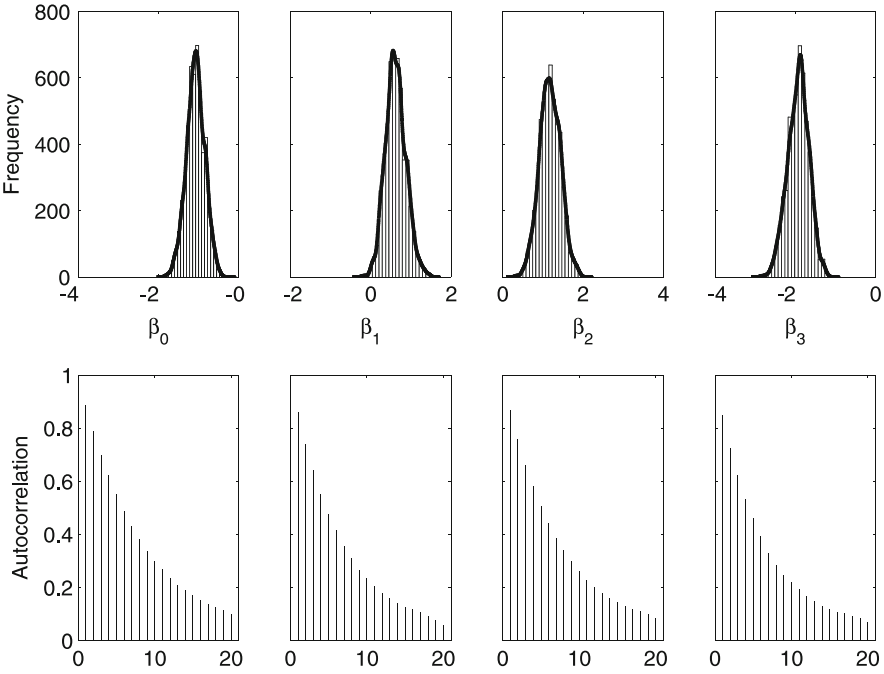


Fig. 4.2 Caesarean data with random-walk M–H algorithm: Marginal posterior densities (*top panel*) and autocorrelation plot (*bottom panel*)

reported in the top panel of Fig. 4.2. These are virtually identical to those computed using the random-walk M–H algorithm. The most notable difference is in the serial correlation plots which decline much more quickly to zero indicating that the algorithm is mixing well. The same information is revealed by the inefficiency factors which are much closer to one than those from the previous algorithm.

The message from this analysis is that the two proposal densities produce similar results, with the differences appearing only in the autocorrelation plots (and inefficiency factors) of the sampled draws (Fig. 4.3).

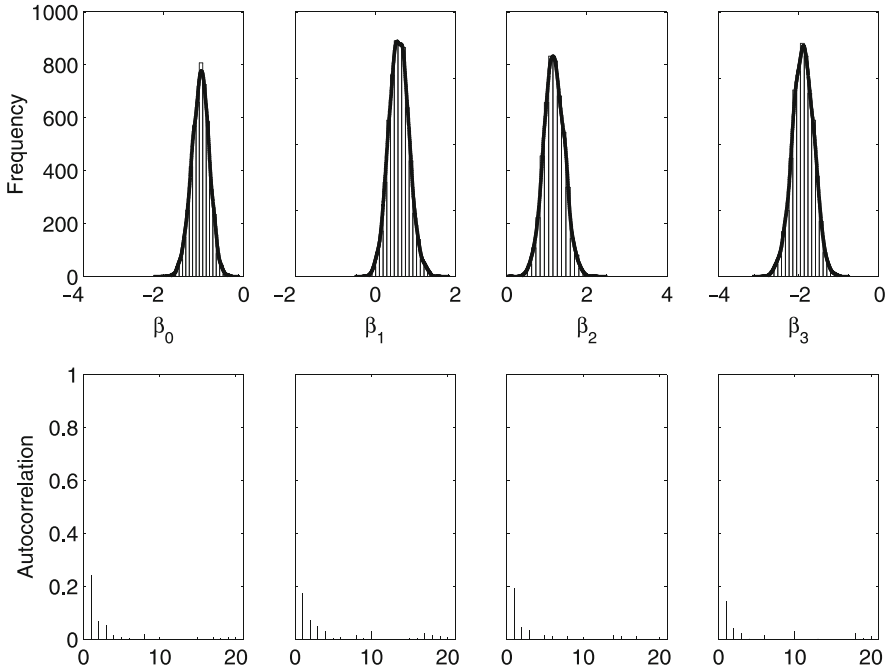


Fig. 4.3 Caesarean data with tailored M–H algorithm: Marginal posterior densities (*top panel*) and autocorrelation plot (*bottom panel*)

4.3.3 Multiple-Block M–H Algorithm

In applications when the dimension of $\boldsymbol{\psi}$ is large, it can be difficult to construct a single block M–H algorithm that converges rapidly to the target density. In such cases, it is helpful to break up the variate space into smaller blocks and to then construct a Markov chain with these smaller blocks. Suppose, for illustration, that $\boldsymbol{\psi}$ is split into two vector blocks $(\boldsymbol{\psi}_1, \boldsymbol{\psi}_2)$. For example, in a regression model, one block may consist of the regression coefficients and the other block may consist of the error variance. Next, for each block, let

$$q_1(\boldsymbol{\psi}_1, \boldsymbol{\psi}_1'|\boldsymbol{\psi}_2)\ ; \quad q_2(\boldsymbol{\psi}_2, \boldsymbol{\psi}_2'|\boldsymbol{\psi}_1)\ ,$$

denote the corresponding proposal density. Here each proposal density q_k is allowed to depend on the data and the current value of the remaining block. Also define (by analogy with the single-block case)

$$\alpha(\boldsymbol{\psi}_1, \boldsymbol{\psi}_1'|\boldsymbol{\psi}_2) = \min\left\{1, \frac{\pi(\boldsymbol{\psi}_1'|\boldsymbol{\psi}_2)q_1(\boldsymbol{\psi}_1', \boldsymbol{\psi}_1|\boldsymbol{\psi}_2)}{\pi(\boldsymbol{\psi}_1|\boldsymbol{\psi}_2)q_1(\boldsymbol{\psi}_1, \boldsymbol{\psi}_1'|\boldsymbol{\psi}_2)}\right\}\ , \tag{4.19}$$

and

$$\alpha(\boldsymbol{\psi}_2, \boldsymbol{\psi}_2' | \boldsymbol{y}, \boldsymbol{\psi}_1) = \min\left\{1, \frac{\pi(\boldsymbol{\psi}_2' | \boldsymbol{\psi}_1) q_2(\boldsymbol{\psi}_2', \boldsymbol{\psi}_2 | \boldsymbol{\psi}_1)}{\pi(\boldsymbol{\psi}_2 | \boldsymbol{\psi}_1) q_2(\boldsymbol{\psi}_2, \boldsymbol{\psi}_2' | \boldsymbol{\psi}_1)}\right\} , \tag{4.20}$$

as the probability of move for block $\boldsymbol{\psi}_k$ $(k = 1, 2)$ conditioned on the other block. The conditional densities

$$\pi(\boldsymbol{\psi}_1 | \boldsymbol{\psi}_2) \quad \text{and} \quad \pi(\boldsymbol{\psi}_2 | \boldsymbol{\psi}_1)$$

that appear in these functions are called the *full conditional densities*. By Bayes theorem each is proportional to the joint density. For example,

$$\pi(\boldsymbol{\psi}_1 | \boldsymbol{\psi}_2) \propto \pi(\boldsymbol{\psi}_1, \boldsymbol{\psi}_2) ,$$

and, therefore, the probabilities of move in (4.19) and (4.20) can be expressed equivalently in terms of the kernel of the joint posterior density $\pi(\boldsymbol{\psi}_1, \boldsymbol{\psi}_2)$ because the normalizing constant of the full conditional density (the norming constant in the latter expression) cancels in forming the ratio.

With these inputs, one sweep of the multiple-block M–H algorithm is completed by updating each block, say sequentially in fixed order, using a M–H step with the above probabilities of move, given the most current value of the other block.

Algorithm 2 Multiple-Block Metropolis–Hastings

1. Specify an initial value $\boldsymbol{\psi}^{(0)} = \left(\boldsymbol{\psi}_1^{(0)}, \boldsymbol{\psi}_2^{(0)}\right)$:
2. Repeat for $j = 1, 2, \ldots, n_0 + M$.
   (a) Repeat for $k = 1, 2$
      I. Propose a value for the kth block, conditioned on the previous value of kth block, and the current value of the other block $\boldsymbol{\psi}_{-k}$:

$$\boldsymbol{\psi}_k' \sim q_k\left(\boldsymbol{\psi}_k^{(j-1)}, \cdot | \boldsymbol{\psi}_{-k}\right) .$$

      II. Calculate the probability of move

$$\alpha_k\left(\boldsymbol{\psi}_k^{(j-1)}, \boldsymbol{\psi}_k' | \boldsymbol{y}, \boldsymbol{\psi}_{-k}\right) = \min\left\{1, \frac{\pi\left(\boldsymbol{\psi}_k' | \boldsymbol{\psi}_{-k}\right) q_k\left(\boldsymbol{\psi}_k', \boldsymbol{\psi}_k^{(j-1)} | \boldsymbol{\psi}_{-k}\right)}{h\left(\boldsymbol{\psi}_k^{(j-1)} | \boldsymbol{\psi}_{-k}\right) q_k\left(\boldsymbol{\psi}_k^{(j-1)}, \boldsymbol{\psi}_k' | \boldsymbol{\psi}_{-k}\right)}\right\} .$$

      III. Update the kth block as

$$\boldsymbol{\psi}_k^{(j)} = \begin{cases} \boldsymbol{\psi}_k' & \text{with prob} \quad \alpha_k\left(\boldsymbol{\psi}_k^{(j-1)}, \boldsymbol{\psi}_k' | \boldsymbol{\psi}_{-k}\right) \\ \boldsymbol{\psi}_k^{(j-1)} & \text{with prob} \quad 1 - \alpha_k\left(\boldsymbol{\psi}_k^{(j-1)}, \boldsymbol{\psi}_k' | \boldsymbol{\psi}_{-k}\right) \end{cases} .$$

3. Return the values $\left\{\boldsymbol{\psi}^{(n_0+1)}, \boldsymbol{\psi}^{(n_0+2)}, \ldots, \boldsymbol{\psi}^{(n_0+M)}\right\}$.

The extension of this method to more than two blocks is straightforward.

The transition kernel of the resulting Markov chain is given by the product of transition kernels

$$P(\boldsymbol{\psi}, d\boldsymbol{\psi}') = \prod_{k=1}^{2} P_k\left(\boldsymbol{\psi}_k, d\boldsymbol{\psi}'_k | \boldsymbol{\psi}_{-k}\right) \tag{4.21}$$

This transition kernel is not reversible, as can be easily checked, because under fixed sequential updating of the blocks updating in the reverse order never occurs. The multiple-block M–H algorithm, however, satisfies the weaker condition of invariance. To show this, we utilize the fact that each sub-move satisfies local reversibility (Chib and Jeliazkov 2001) and therefore the transition kernel $P_1(\boldsymbol{\psi}_1, d\boldsymbol{\psi}_1|\boldsymbol{\psi}_2)$ has $\pi^*_{1|2}(\cdot|\boldsymbol{\psi}_2)$ as its local invariant distribution with density $\pi^*_{1|2}(\cdot|\boldsymbol{\psi}_2)$, i.e.,

$$\pi^*_{1|2}(d\boldsymbol{\psi}_1|\boldsymbol{\psi}_2) = \int P_1(\boldsymbol{\psi}_1, d\boldsymbol{\psi}_1|\boldsymbol{\psi}_2)\pi_{1|2}(\boldsymbol{\psi}_1|\boldsymbol{\psi}_2)\, d\boldsymbol{\psi}_1 \ . \tag{4.22}$$

Similarly, the conditional transition kernel $P_2(\boldsymbol{\psi}_2, d\boldsymbol{\psi}_2|\boldsymbol{\psi}_1)$ has $\pi^*_{2|1}(\cdot|\boldsymbol{\psi}_1)$ as its invariant distribution, for a given value of $\boldsymbol{\psi}_1$. Then, the kernel formed by multiplying the conditional kernels is invariant for $\pi^*(\cdot,\cdot)$:

$$\begin{aligned}
&\iint P_1(\boldsymbol{\psi}_1, d\boldsymbol{\psi}'_1|\boldsymbol{\psi}_2) P_2(\boldsymbol{\psi}_2, d\boldsymbol{\psi}'_2|\boldsymbol{\psi}'_1)\pi(\boldsymbol{\psi}_1, \boldsymbol{\psi}_2)\, d\boldsymbol{\psi}_1\, d\boldsymbol{\psi}_2 \\
&= \int P_2(\boldsymbol{\psi}_2, d\boldsymbol{\psi}'_2|\boldsymbol{\psi}'_1)\left[\int P_1(\boldsymbol{\psi}_1, d\boldsymbol{\psi}'_1|\boldsymbol{\psi}_2)\pi_{1|2}(\boldsymbol{\psi}_1|\boldsymbol{\psi}_2)\, d\boldsymbol{\psi}_1\right]\pi_2(\boldsymbol{\psi}_2)\, d\boldsymbol{\psi}_2 \\
&= \int P_2(\boldsymbol{\psi}_2, d\boldsymbol{\psi}'_2|\boldsymbol{\psi}'_1)\pi^*_{1|2}(d\boldsymbol{\psi}'_1|\boldsymbol{\psi}_2)\pi_2(\boldsymbol{\psi}_2)\, d\boldsymbol{\psi}_2 \\
&= \int P_2(\boldsymbol{\psi}_2, d\boldsymbol{\psi}'_2|\boldsymbol{\psi}'_1)\frac{\pi_{2|1}(\boldsymbol{\psi}_2|\boldsymbol{\psi}'_1)\pi^*_1(d\boldsymbol{\psi}'_1)}{\pi_2(\boldsymbol{\psi}_2)}\pi_2(\boldsymbol{\psi}_2)\, d\boldsymbol{\psi}_2 \\
&= \pi^*_1(d\boldsymbol{\psi}'_1)\int P_2(\boldsymbol{\psi}_2, d\boldsymbol{\psi}'_2|\boldsymbol{\psi}'_1)\pi_{2|1}(\boldsymbol{\psi}_2|\boldsymbol{\psi}'_1)\, d\boldsymbol{\psi}_2 \\
&= \pi^*_1(d\boldsymbol{\psi}'_1)\pi^*_{2|1}(d\boldsymbol{\psi}'_2|\boldsymbol{\psi}'_1) \\
&= \pi^*(d\boldsymbol{\psi}'_1, d\boldsymbol{\psi}'_2)\ ,
\end{aligned}$$

where the third line follows from (4.22), the fourth from Bayes theorem, the sixth from assumed invariance of P_2, and the last from the law of total probability.

The implication of this result is that it allows us to take draws in succession from each of the kernels, instead of having to run each to convergence for every value of the conditioning variable.

Remark 1. Versions of either random-walk or tailored proposal densities can be used in this algorithm, analogous to the single-block case. For example, Chib and Greenberg (1995) determine the proposal densities q_k by tailoring to $\pi(\boldsymbol{\psi}_k, \boldsymbol{\psi}_{-k})$ in which case the proposal density is not fixed but varies across iterations. An

important special case occurs if each proposal density is taken to be the full conditional density of that block. Specifically, if we set

$$q_1\left(\boldsymbol{\psi}_1^{(j-1)}, \boldsymbol{\psi}_1' | \boldsymbol{\psi}_2\right) = \pi(\boldsymbol{\psi}_1' | \boldsymbol{\psi}_2) \ ,$$

and

$$q_2\left(\boldsymbol{\psi}_2^{(j-1)}, \boldsymbol{\psi}_2' | \boldsymbol{\psi}_1\right) = \pi(\boldsymbol{\psi}_2' | \boldsymbol{\psi}_1) \ ,$$

then an interesting simplification occurs. The probability of move (for the first block) becomes

$$\begin{aligned} \alpha_1\left(\boldsymbol{\psi}_1^{(j-1)}, \boldsymbol{\psi}_1' | \boldsymbol{\psi}_2\right) &= \min\left\{1, \frac{\pi\left(\boldsymbol{\psi}_1' | \boldsymbol{\psi}_2\right) \pi\left(\boldsymbol{\psi}_1^{(j-1)} | \boldsymbol{\psi}_2\right)}{\pi\left(\boldsymbol{\psi}_1^{(j-1)} | \boldsymbol{\psi}_2\right) \pi\left(\boldsymbol{\psi}_1' | \boldsymbol{\psi}_2\right)}\right\} \\ &= 1 \ , \end{aligned}$$

and similarly for the second block, implying that if proposal values are drawn from their full conditional densities then the proposal values are accepted with probability one. This special case of the multiple-block M–H algorithm (in which *each* block is proposed using its full conditional distribution) is called the Gibbs sampling algorithm.

4.4 The Gibbs Sampling Algorithm

The Gibbs sampling algorithm is one of the simplest Markov chain Monte Carlo algorithms. It was introduced by Geman and Geman (1984) in the context of image processing and then discussed in the context of missing data problems by Tanner and Wong (1987). The paper by Gelfand and Smith (1990) helped to demonstrate the value of the Gibbs algorithm for a range of problems in Bayesian analysis.

4.4.1 The Algorithm

To define the Gibbs sampling algorithm, let the set of full conditional distributions be

$$\left\{\pi(\boldsymbol{\psi}_1 | \boldsymbol{\psi}_2, \ldots, \boldsymbol{\psi}_p); \ \pi(\boldsymbol{\psi}_2 | \boldsymbol{\psi}_1, \boldsymbol{\psi}_3, \ldots, \boldsymbol{\psi}_p); \ \ldots, \pi(\boldsymbol{\psi}_p | \boldsymbol{\psi}_1, \ldots, \boldsymbol{\psi}_{d-1})\right\} \ .$$

Now one cycle of the Gibbs sampling algorithm is completed by simulating $\{\boldsymbol{\psi}_k\}_{k=1}^p$ from these distributions, recursively refreshing the conditioning variables. When $d = 2$ one obtains the two block Gibbs sampler that appears in Tanner and

Wong (1987). The Gibbs sampler in which each block is revised in fixed order is defined as follows.

Algorithm 3 Gibbs Sampling

1. Specify an initial value $\boldsymbol{\psi}^{(0)} = \left(\boldsymbol{\psi}_1^{(0)}, \ldots, \boldsymbol{\psi}_p^{(0)}\right)$:
2. Repeat for $j = 1, 2, \ldots, M$.

   Generate $\boldsymbol{\psi}_1^{(j+1)}$ from $\pi\left(\boldsymbol{\psi}_1 | \boldsymbol{\psi}_2^{(j)}, \boldsymbol{\psi}_3^{(j)}, \ldots, \boldsymbol{\psi}_p^{(j)}\right)$

   Generate $\boldsymbol{\psi}_2^{(j+1)}$ from $\pi\left(\boldsymbol{\psi}_2 | \boldsymbol{\psi}_1^{(j+1)}, \boldsymbol{\psi}_3^{(j)}, \ldots, \boldsymbol{\psi}_p^{(j)}\right)$

   $\vdots$

   Generate $\boldsymbol{\psi}_p^{(j+1)}$ from $\pi\left(\boldsymbol{\psi}_p | \boldsymbol{\psi}_1^{(j+1)}, \ldots, \boldsymbol{\psi}_{p-1}^{(j+1)}\right)$.

3. Return the values $\left\{\boldsymbol{\psi}^{(1)}, \boldsymbol{\psi}^{(2)}, \ldots, \boldsymbol{\psi}^{(M)}\right\}$.

It follows that the transition density of moving from $\boldsymbol{\psi}_k^{(j)}$ to $\boldsymbol{\psi}_k^{(j+1)}$ is given by

$$\pi\left(\boldsymbol{\psi}_k | \boldsymbol{\psi}_1^{(j+1)}, \ldots, \boldsymbol{\psi}_{k-1}^{(j+1)}, \boldsymbol{\psi}_{k+1}^{(j)}, \ldots, \boldsymbol{\psi}_p^{(j)}\right)$$

since when the kth block is reached, the previous $(k-1)$ blocks have been updated. Thus, the transition density of the chain, under the maintained assumption that π is absolutely continuous, is given by the product of transition kernels for each block:

$$K(\boldsymbol{\psi}, \boldsymbol{\psi}') = \prod_{k=1}^{p} \pi\left(\boldsymbol{\psi}_k | \boldsymbol{\psi}_1^{(j+1)}, \ldots, \boldsymbol{\psi}_{k-1}^{(j+1)}, \boldsymbol{\psi}_{k+1}^{(j)}, \ldots, \boldsymbol{\psi}_p^{(j)}\right) . \tag{4.23}$$

To illustrate the manner in which the blocks are revised, we consider a two block case, each with a single component, and trace out in Fig. 4.4 a possible trajectory of the sampling algorithm. The contours in the plot represent the joint distribution of $\boldsymbol{\psi}$ and the labels "(0)", "(1)" etc., denote the simulated values. Note that one iteration of the algorithm is completed after both components are revised. Also notice that each component is revised along the direction of the coordinate axes. This feature can be a source of problems if the two components are highly correlated because then the contours get compressed and movements along the coordinate axes tend to produce small moves. We return to this issue below.

4.4.2 Invariance of the Gibbs Markov Chain

The Gibbs transition kernel is invariant by construction. This is a consequence of the fact that the Gibbs algorithm is a special case of the multiple-block M–H algorithm

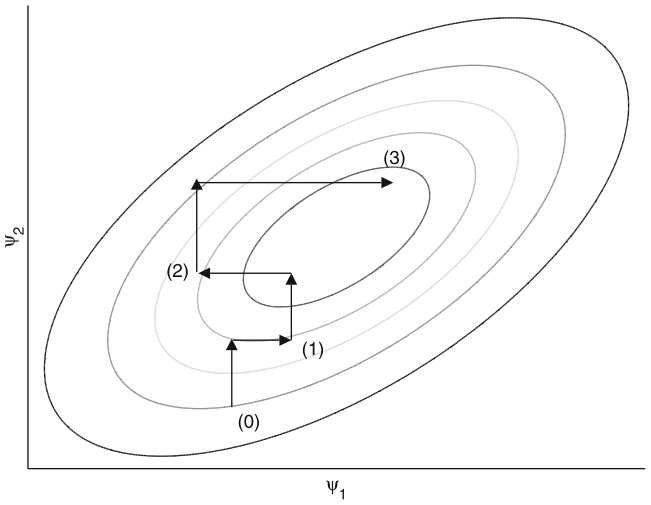


Fig. 4.4 Gibbs sampling algorithm in two dimensions starting from an initial point and then completing three iterations

which is invariant, as was established in the last section. Invariance can also be confirmed directly. Consider for simplicity a two block sampler with transition kernel density

$$K(\boldsymbol{\psi}, \boldsymbol{\psi}') = \pi(\boldsymbol{\psi}_1'|\boldsymbol{\psi}_2)\pi(\boldsymbol{\psi}_2'|\boldsymbol{\psi}_1') \, .$$

To check invariance we have to show that

$$\begin{aligned} &\int K(\boldsymbol{\psi}, \boldsymbol{\psi}')\pi(\boldsymbol{\psi}_1, \boldsymbol{\psi}_2) d\boldsymbol{\psi}_1 d\boldsymbol{\psi}_2 \\ &\quad = \int \pi(\boldsymbol{\psi}_1'|\boldsymbol{\psi}_2)\pi(\boldsymbol{\psi}_2'|\boldsymbol{\psi}_1')\pi(\boldsymbol{\psi}_1, \boldsymbol{\psi}_2) d\boldsymbol{\psi}_1 d\boldsymbol{\psi}_2 \end{aligned}$$

is equal to $\pi(\boldsymbol{\psi}_1', \boldsymbol{\psi}_2')$. This holds because $\pi(\boldsymbol{\psi}_2'|\boldsymbol{\psi}_1')$ comes out of the integral, and the integral over $\boldsymbol{\psi}_1$ and $\boldsymbol{\psi}_2$ produces $\pi(\boldsymbol{\psi}_1')$. This calculation can be extended to any number of blocks. It may be noted that the Gibbs Markov chain is not reversible. Reversible Gibbs samplers are discussed by Liu et al. (1995).

4.4.3 Sufficient Conditions for Convergence

Under rather general conditions, the Markov chain generated by the Gibbs sampling algorithm converges to the target density as the number of iterations become large. Formally, if we let $K(\boldsymbol{\psi}, \boldsymbol{\psi}')$ represent the transition density of the Gibbs algorithm and let $K^{(M)}(\boldsymbol{\psi}_0, \boldsymbol{\psi}')$ be the density of the draw $\boldsymbol{\psi}'$ after M iterations given the

starting value $\boldsymbol{\psi}_0$, then

$$\left\| K^{(M)}\left(\boldsymbol{\psi}^{(0)}, \boldsymbol{\psi}'\right) - \pi(\boldsymbol{\psi}')\right\| \to 0\,, \quad \text{as} \quad M \to \infty\,. \tag{4.24}$$

Roberts and Smith (1994) (see also Chan 1993) have shown that the conditions of Proposition 2 are satisfied under the following conditions: (1) $\pi(\boldsymbol{\psi}) > 0$ implies there exists an open neighborhood $N_{\boldsymbol{\psi}}$ containing $\boldsymbol{\psi}$ and $\epsilon > 0$ such that, for all $\boldsymbol{\psi}' \in N_{\boldsymbol{\psi}}$, $\pi(\boldsymbol{\psi}') \geq \epsilon > 0$; (2) $\int f(\boldsymbol{\psi})\, d\boldsymbol{\psi}_k$ is locally bounded for all k, where $\boldsymbol{\psi}_k$ is the kth block of parameters; and (3) the support of $\boldsymbol{\psi}$ is arc connected.

These conditions are satisfied in a wide range of problems.

4.4.4 Example: Simulating a Truncated Multivariate Normal

Consider the question of sampling a trivariate normal distribution truncated to the positive orthant. In particular, suppose that the target distribution is

$$\begin{aligned}\pi(\boldsymbol{\psi}) &= \frac{1}{\Pr(\boldsymbol{\psi} \in A)} f_N(\boldsymbol{\mu}, \boldsymbol{\Sigma}) I(\boldsymbol{\psi} \in A) \\ &\propto f_N(\boldsymbol{\mu}, \boldsymbol{\Sigma}) I(\boldsymbol{\psi} \in A)\end{aligned}$$

where $\boldsymbol{\mu} = (0.5, 1, 1.5)'$, $\boldsymbol{\Sigma}$ is in equi-correlated form with units on the diagonal and 0.7 on the off-diagonal, $A = (0, \infty) \times (0, \infty) \times (0, \infty)$ and $\Pr(\boldsymbol{\psi} \in A)$ is the normalizing constant which is difficult to compute. In this case, the Gibbs sampler is defined with the blocks ψ_1, ψ_2, ψ_3 and the full conditional distributions

$$\pi(\psi_1|\psi_2, \psi_3)\,; \quad \pi(\psi_2|\psi_1, \psi_3)\,; \quad \pi(\psi_3|\psi_1, \psi_2)\,,$$

where each of the these full conditional distributions is univariate truncated normal restricted to the interval $(0, \infty)$:

$$\begin{aligned}\pi(\psi_k|\boldsymbol{\psi}_{-k}) \propto f_N\left(\psi_k|\mu_k + \boldsymbol{C}_k'\boldsymbol{\Sigma}_{-k}^{-1}\left(\boldsymbol{\psi}_{-k} - \boldsymbol{\mu}_{-k}\right), \Sigma_k\right. \\ \left. -\boldsymbol{C}_k'\boldsymbol{\Sigma}_{-k}^{-1}\boldsymbol{C}_k\right) I(\psi_k \in (0, \infty))\,,\end{aligned} \tag{4.25}$$

$\boldsymbol{C}_k = Cov(\psi_k, \boldsymbol{\psi}_{-k})$, $\boldsymbol{\Sigma}_{-k} = Var(\boldsymbol{\psi}_{-k})$ and $\boldsymbol{\mu}_{-k} = E(\boldsymbol{\psi}_{-k})$. Figure 4.5 gives the marginal distribution of each component of ψ_k from a Gibbs sampling run of $M = 10000$ iterations with a burn-in of 100 cycles. The figures includes both the histograms of the sampled values and the Rao–Blackwellized estimates of the marginal densities (see Sect. 4.6 below) based on the averaging of (4.25) over the simulated values of $\boldsymbol{\psi}_{-k}$. The agreement between the two density estimates is close. In the bottom panel of Fig. 4.5 we plot the autocorrelation function of the sampled draws. The rapid decline in the autocorrelations for higher lags indicates that the sampler is mixing well.

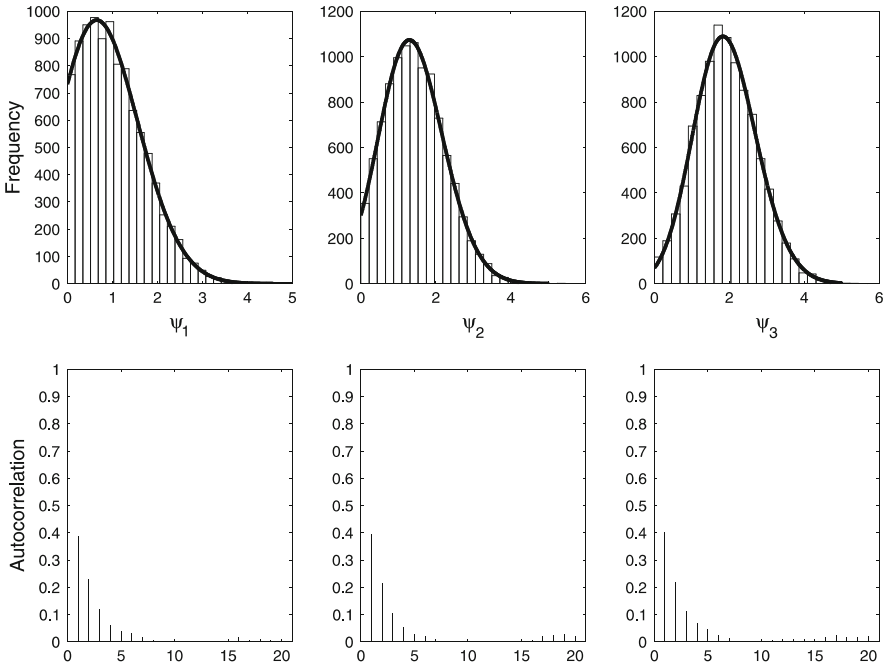


Fig. 4.5 Marginal distributions of ψ in truncated multivariate normal example (*top panel*). Histograms of the sampled values and Rao–Blackwellized estimates of the densities are shown. Autocorrelation plots of the Gibbs MCMC chain are in the *bottom panel*. Graphs are based on 10,000 iterations following a burn-in of 500 cycles

4.5 MCMC Sampling with Latent Variables

In designing MCMC simulations, it is sometimes helpful to modify the target distribution by introducing latent variables or auxiliary variables into the sampling. This idea was called data augmentation by Tanner and Wong (1987) in the context of missing data problems. Slice sampling, which we do not discuss in this chapter, is a particular way of introducing auxiliary variables into the sampling, for example see Damien et al. (1999).

To fix notations, suppose that z denotes a vector of latent variables and let the modified target distribution be $\pi(\boldsymbol{\psi}, z)$. If the latent variables are tactically introduced, the conditional distribution of $\boldsymbol{\psi}$ (or sub components of $\boldsymbol{\psi}$) given z may be easy to derive. Then, a multiple-block M–H simulation is conducted with the blocks $\boldsymbol{\psi}$ and z leading to the sample

$$\left(\boldsymbol{\psi}^{(n_0+1)}, z^{(n_0+1)}\right), \ldots, \left(\boldsymbol{\psi}^{(n_0+M)}, z^{(n_0+M)}\right) \sim \pi(\boldsymbol{\psi}, z) \ ,$$

where the draws on $\boldsymbol{\psi}$, ignoring those on the latent data, are from $\pi(\boldsymbol{\psi})$, as required.

To demonstrate this technique in action, we return to the probit regression example discussed in Sect. 4.3.2 to show how a MCMC sampler can be developed with the help of latent variables. The approach, introduced by Albert and Chib (1993), capitalizes on the simplifications afforded by introducing latent or auxiliary data into the sampling.

The model is rewritten as

$$\begin{aligned} z_i|\boldsymbol{\beta} &\sim \mathrm{N}(\boldsymbol{x}_i'\boldsymbol{\beta}, 1) \ , \\ y_i &= I[z_i > 0] \ , \quad i \le n \ , \\ \boldsymbol{\beta} &\sim \mathrm{N}_k(\boldsymbol{\beta}_0, \boldsymbol{B}_0) \ . \end{aligned} \tag{4.26}$$

This specification is equivalent to the probit regression model since

$$\Pr(y_i = 1|\boldsymbol{x}_i, \boldsymbol{\beta}) = \Pr(z_i > 0|\boldsymbol{x}_i, \boldsymbol{\beta}) = \Phi(\boldsymbol{x}_i'\boldsymbol{\beta}) \ .$$

Now the Albert–Chib algorithm proceeds with the sampling of the full conditional distributions

$$\boldsymbol{\beta}|\boldsymbol{y}, \{z_i\} \ ; \quad \{z_i\}|\boldsymbol{y}, \boldsymbol{\beta} \ ,$$

where both these distributions are tractable (i.e., requiring no M–H steps). In particular, the distribution of $\boldsymbol{\beta}$ conditioned on the latent data becomes independent of the observed data and has the same form as in the Gaussian linear regression model with the response data given by $\{z_i\}$ and is multivariate normal with mean $\hat{\boldsymbol{\beta}} = \boldsymbol{B}(\boldsymbol{B}_0^{-1}\boldsymbol{\beta}_0 + \sum_{i=1}^n \boldsymbol{x}_i z_i)$ and variance matrix $\boldsymbol{B} = (\boldsymbol{B}_0^{-1} + \sum_{i=1}^n \boldsymbol{x}_i\boldsymbol{x}_i')^{-1}$. Next, the distribution of the latent data conditioned on the data and the parameters factor into a set of n independent distributions with each depending on the data through y_i:

$$\{z_i\}|\boldsymbol{y}, \boldsymbol{\beta} \stackrel{d}{=} \prod_{i=1}^n z_i|y_i, \boldsymbol{\beta} \ ,$$

where the distribution $z_i|y_i, \boldsymbol{\beta}$ is the normal distribution $z_i|\boldsymbol{\beta}$ truncated by the knowledge of y_i; if $y_i = 0$, then $z_i \le 0$ and if $y_i = 1$, then $z_i > 0$. Thus, one samples z_i from $\mathcal{TN}_{(-\infty,0)}(\boldsymbol{x}_i'\boldsymbol{\beta}, 1)$ if $y_i = 0$ and from $\mathcal{TN}_{(0,\infty)}(\boldsymbol{x}_i'\boldsymbol{\beta}, 1)$ if $y_i = 1$, where $\mathcal{TN}_{(a,b)}(\mu, \sigma^2)$ denotes the $\mathcal{N}(\mu, \sigma^2)$ distribution truncated to the region (a, b).

The results, based on 5,000 MCMC draws beyond a burn-in of a 100 iterations, are reported in Fig. 4.4. The results are close to those presented above, especially to the ones from the tailored M–H chain (Fig. 4.6).

4.6 Estimation of Density Ordinates

We mention that if the full conditional densities are available, whether in the context of the multiple-block M–H algorithm or that of the Gibbs sampler, then the MCMC output can be used to estimate posterior marginal density functions (Gelfand and

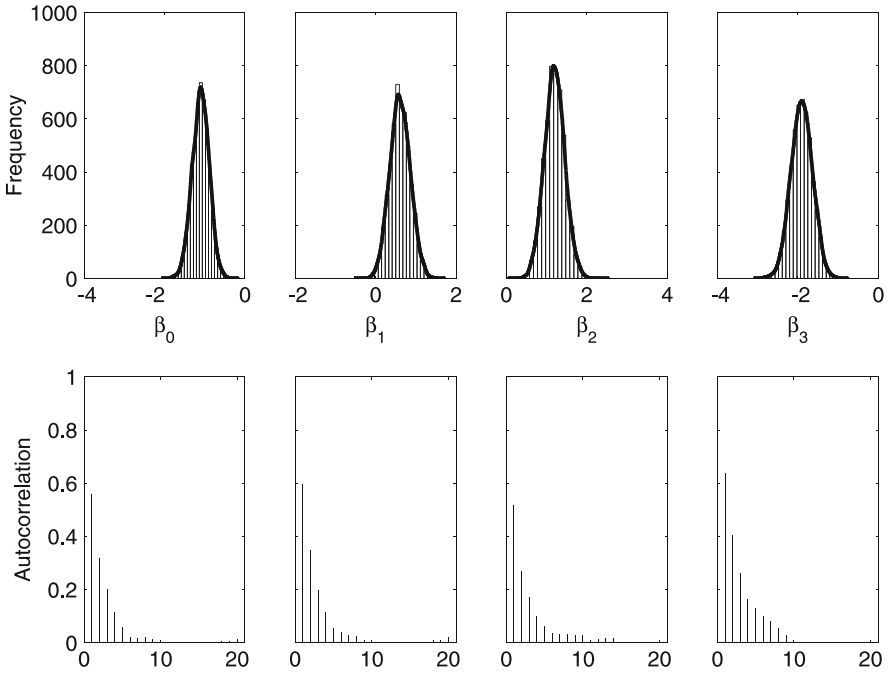


Fig. 4.6 Caesarean data with Albert–Chib algorithm: Marginal posterior densities (*top panel*) and autocorrelation plot (*bottom panel*)

Smith 1990; Tanner and Wong 1987). We exploit the fact that the marginal density of $\boldsymbol{\psi}_k$ at the point $\boldsymbol{\psi}_k^*$ is

$$\pi(\boldsymbol{\psi}_k^*) = \int \pi(\boldsymbol{\psi}_k^* | \boldsymbol{\psi}_{-k})\, \pi(\boldsymbol{\psi}_{-k}) d\boldsymbol{\psi}_{-k} \ ,$$

where as before $\boldsymbol{\psi}_{-k} = \boldsymbol{\psi} \backslash \boldsymbol{\psi}_k$. Provided the normalizing constant of $\pi\left(\boldsymbol{\psi}_k^* | \boldsymbol{\psi}_{-k}\right)$ is known, an estimate of the marginal density is available as an average of the full conditional density over the simulated values of $\boldsymbol{\psi}_{-k}$:

$$\hat{\pi}(\boldsymbol{\psi}_k^*) = M^{-1} \sum_{j=1}^{M} \pi\left(\boldsymbol{\psi}_k^* | \boldsymbol{\psi}_{-k}^{(j)}\right) \ .$$

Under the assumptions of Proposition 1,

$$M^{-1} \sum_{j=1}^{M} \pi\left(\boldsymbol{\psi}_k^* | \boldsymbol{\psi}_{-k}^{(j)}\right) \to \pi(\boldsymbol{\psi}_k^*) \ , \quad \text{as} \quad M \to \infty \ .$$

Gelfand and Smith (1990) refer to this approach as Rao–Blackwellization because of the connections with the Rao–Blackwell theorem in classical statistics. That connection is more clearly seen in the context of estimating (say) the mean of ψ_k, $E(\psi_k) = \int \psi_k \pi(\psi_k) d\psi_k$. By the law of the iterated expectation,

$$E(\psi_k) = E\{E(\psi_k|\psi_{-k})\}$$

and therefore the estimates

$$M^{-1}\sum_{j=1}^{M}\psi_k^j$$

and

$$M^{-1}\sum_{j=1}^{M}E\left(\psi_k|\psi_{-k}^{(j)}\right)$$

both converge to $E(\psi_k)$ as $M \to \infty$. Under iid sampling, and under Markov sampling provided some conditions are satisfied – see Liu et al. (1994), Casella and Robert (1996) and Robert and Casella (1999), it can be shown that the variance of the latter estimate is smaller than that of the former. Thus, it can help to average the conditional mean $E(\psi_k|\psi_{-k})$, if that were available, rather than average the draws directly. Gelfand and Smith (1990) appeal to this analogy to argue that the Rao–Blackwellized estimate of the density is preferable to that based on the method of kernel smoothing. Chib (1995) extends the Rao–Blackwellization approach to estimate reduced conditional ordinates defined as the density of ψ_k conditioned on one or more of the remaining blocks. Finally, Chen (1994) provides an importance weighted estimate of the marginal density for cases where the conditional posterior density does not have a known normalizing constant. Chen's estimator is based on the identity

$$\pi(\psi_k^*) = \int w(\psi_k|\psi_{-k})\frac{\pi(\psi_k^*,\psi_{-k})}{\pi(\psi_k,\psi_{-k})}\pi(\psi)d\psi \ ,$$

where $w(\psi_k|\psi_{-k})$ is a completely known conditional density whose support is equal to the support of the full conditional density $\pi(\psi_k|\psi_{-k})$. In this form, the normalizing constant of the full conditional density is not required and given a sample of draws $\{\psi^{(1)},\ldots,\psi^{(M)}\}$ from $\pi(\psi)$, a Monte Carlo estimate of the marginal density is given by

$$\hat{\pi}(\psi_k^*) = M^{-1}\sum_{j=1}^{M} w\left(\psi_k^{(j)}\middle|\psi_{-k}^{(j)}\right)\frac{\pi\left(\psi_k^*,\psi_{-k}^{(j)}\right)}{\pi\left(\psi_k^{(j)},\psi_{-k}^{(j)}\right)} \ .$$

Chen (1994) discusses the choice of the conditional density w. Since it depends on ψ_{-k}, the choice of w will vary from one sampled draw to the next.

4.7 Sampler Performance and Diagnostics

In implementing a MCMC method it is important to assess the performance of the sampling algorithm to determine the rate of mixing and the size of the burn-in, both having implications for the number of iterations required to get reliable answers. A large literature has emerged on these issues, for example, Robert (1995), Tanner (1996, Sect. 6.3), Cowles and Carlin (1996), Gammermann (1997, Sect. 5.4) and Robert and Casella (1999), but the ideas, although related in many ways, have not coalesced into a single prescription.

One approach for determining sampler performance and the size of the burn-in time is to employ analytical methods to the specified Markov chain, prior to sampling. This approach is exemplified in the work of, for example, Polson (1996), Roberts and Tweedie (1996) and Rosenthal (1995). Two factors have inhibited the growth and application of these methods. The first is that the calculations are difficult and problem-specific and, second, the upper bounds for the burn-in that emerge from such calculations are usually conservative.

At this time the more popular approach is to utilize the sampled draws to assess both the performance of the algorithm and its approach to the invariant distribution. Several such relatively informal methods are available. Gelfand and Smith (1990) recommend monitoring the evolution of the quantiles as the sampling proceeds. Another useful diagnostic, one that is perhaps the most direct, are autocorrelation plots (and autocorrelation times) of the sampled output. Slowly decaying correlations indicate problems with the mixing of the chain. It is also useful in connection with M–H Markov chains to monitor the acceptance rate of the proposal values with low rates implying "stickiness" in the sampled values and thus a slower approach to the invariant distribution.

Somewhat more formal sample-based diagnostics are summarized in the CODA routines provided by Best et al. (1995). Although these diagnostics often go under the name "convergence diagnostics" they are in principle approaches that detect lack of convergence. Detection of convergence based entirely on the sampled output, without analysis of the target distribution, is perhaps impossible. Cowles and Carlin (1996) discuss and evaluate thirteen such diagnostics (for example, those proposed by Geweke 1992; Raftery and Lewis 1992; Ritter and Tanner 1992; Gelman and Rubin 1992; Gelman and Rubin 1992; and Zellner and Min 1995, amongst others) without arriving at a consensus. Difficulties in evaluating these methods stem from the fact that some of these methods apply only to Gibbs Markov chains (for example, those of Ritter and Tanner 1992; and Zellner and Min 1995) while others are based on the output not just of a single chain but on that of multiple chains specifically run from "disparate starting values" as in the method of Gelman and Rubin (1992). Finally, some methods assess the behavior of univariate moment estimates (as in the approach of Geweke 1992; and Gelman and Rubin 1992) while others are concerned with the behavior of the entire transition kernel (as in Ritter and Tanner 1992; and Zellner and Min 1995).

4.8 Strategies for Improving Mixing

In practice, while implementing MCMC methods it is important to construct samplers that mix well, where mixing is measured by the autocorrelation time, because such samplers can be expected to converge more quickly to the invariant distribution. Over the years a number of different recipes for designing samplers with low autocorrelation times have been proposed although it may sometimes be difficult, because of the complexity of the problem, to apply any of these recipes.

4.8.1 Choice of Blocking

As a general rule, sets of parameters that are highly correlated should be treated as one block when applying the multiple-block M–H algorithm. Otherwise, it would be difficult to develop proposal densities that lead to large moves through the support of the target distribution.

Blocks can be combined by the method of composition. For example, suppose that $\boldsymbol{\psi}_1, \boldsymbol{\psi}_2$ and $\boldsymbol{\psi}_3$ denote three blocks and that the distribution $\boldsymbol{\psi}_1|\boldsymbol{\psi}_3$ is tractable (i.e., can be sampled directly). Then, the blocks $(\boldsymbol{\psi}_1, \boldsymbol{\psi}_2)$ can be collapsed by first sampling $\boldsymbol{\psi}_1$ from $\boldsymbol{\psi}_1|\boldsymbol{\psi}_3$ followed by $\boldsymbol{\psi}_2$ from $\boldsymbol{\psi}_2|\boldsymbol{\psi}_1, \boldsymbol{\psi}_3$. This amounts to a two block MCMC algorithm. In addition, if it is possible to sample $(\boldsymbol{\psi}_1, \boldsymbol{\psi}_2)$ marginalized over $\boldsymbol{\psi}_3$ then the number of blocks is reduced to one. Liu et al. (1994) discuss the value of these strategies in the context of a three-block Gibbs MCMC chains. Roberts and Sahu (1997) provide further discussion of the role of blocking in the context of Gibbs Markov chains used to sample multivariate normal target distributions.

4.8.2 Tuning the Proposal Density

As mentioned above, the proposal density in a M–H algorithm has an important bearing on the mixing of the MCMC chain. Fortunately, one has great flexibility in the choice of candidate generating density and it is possible to adapt the choice to the given problem. For example, Chib et al. (1998) develop and compare four different choices in longitudinal random effects models for count data. In this problem, each cluster (or individual) has its own random effects and each of these has to be sampled from an intractable target distribution. If one lets n denote the number of clusters, where n is typically large, say in excess of a thousand, then the number of blocks in the MCMC implementation is $n + 3$ (n for each of the random effect distributions, two for the fixed effects and one for the variance components matrix). For this problem, the multiple-block M–H algorithm requires $n + 1$ M–H steps within one iteration of the algorithm. Tailored proposal densities are therefore

computationally expensive but one can use a mixture of proposal densities where a less demanding proposal, for example a random walk proposal, is combined with the tailored proposal to sample each of the n random effect target distributions. Further discussion of mixture proposal densities is contained in Tierney (1994).

4.8.3 Other Strategies

Other approaches have also been discussed in the literature. Marinari and Parsi (1992) develop the simulated tempering method whereas Geyer and Thompson (1995) develop a related technique that they call the Metropolis-coupled MCMC method. Both these approaches rely on a series of transition kernels $\{K_1, \ldots, K_m\}$ where only K_1 has π^* as the stationary distribution. The other kernels have equilibrium distributions π_i, which Geyer and Thompson (1995) take to be $\pi_i(\boldsymbol{\psi}) = \pi(\boldsymbol{\psi})^{1/i}$, $i = 2, \ldots, m$. This specification produces a set of target distributions that have higher variance than π^*. Once the transition kernels and equilibrium distributions are specified then the Metropolis-coupled MCMC method requires that each of the m kernels be used in parallel. At each iteration, after the m draws have been obtained, one randomly selects two chains to see if the states should be swapped. The probability of swap is based on the M–H acceptance condition. At the conclusion of the sampling, inference is based on the sequence of draws that correspond to the distribution π^*. These methods promote rapid mixing because draws from the various "flatter" target densities have a chance of being swapped with the draws from the base kernel K_1. Thus, variates that are unlikely under the transition K_1 have a chance of being included in the chain, leading to more rapid exploration of the parameter space.

4.9 Concluding Remarks

In this survey we have provided an outline of Markov chain Monte Carlo methods. These methods provide a set of general recipes for sampling intractable multivariate distributions and have proved vital in the recent virtually revolutionary evolution and growth of Bayesian statistics. Refinements and extensions of these methods continue to occur. Two recent developments are the slice sampling method discussed by Mira and Tierney (2002), Damien et al. (1999) and Roberts and Rosenthal (1999) and the perfect sampling method proposed by Propp and Wilson (1996). The slice sampling method is based on the introduction of auxiliary uniform random variables to simplify the sampling and improve mixing while the perfect sampling method uses Markov chain coupling to generate an exact draw from the target distribution.

References

Albert, J., Chib, S.: Bayesian analysis of binary and polychotomous response data. J. Am. Stat. Assoc. **88**, 669–679 (1993)

Besag, J.: Spatial interaction and the statistical analysis of lattice systems (withdiscussion). J. Roy. Stat. Soc. B **36**, 192–236 (1974)

Besag, J., Green, E., Higdon, D., Mengersen, K.L.: Bayesian computation and stochastic systems (with discussion). Stat. Sci. **10**, 3–66 (1995)

Best, N.G., Cowles, M.K., Vines, S.K.: CODA: Convergence diagnostics and output analysis software for Gibbs sampling. Technical report, Cambridge MRC Biostatistics Unit (1995)

Carlin, B.P., Louis, T.: Bayes and Empirical Bayes Methods for Data Analysis, (2nd edn.), Chapman and Hall, London (2000)

Casella, G., Robert, C.P.: Rao–Blackwellization of sampling schemes. Biometrika **83**, 81–94 (1996)

Chan, K.S.: Asymptotic behavior of the Gibbs sampler. J. Am. Stat. Assoc. **88**, 320–326 (1993)

Chan, K.S., Ledolter, J.: Monte Carlo EM estimation for time series models involving counts. J. Am. Stat. Assoc. **90**, 242–252 (1995)

Chen, M-H.: Importance-weighted marginal Bayesian posterior density estimation. J. Am. Stat. Assoc. **89**, 818–824 (1994)

Chen, M-H., Shao, Qi-M., Ibrahim, J.G.: Monte Carlo Methods in Bayesian Computation. Springer, New York (2000)

Chib, S.: Marginal likelihood from the Gibbs output. J. Am. Stat. Assoc. **90**, 1313–1321 (1995)

Chib, S.: Markov Chain Monte Carlo Methods: Computation and Inference. In: Heckman, J.J., Leamer, E. (eds.) Handbook of Econometrics, Vol. 5, pp. 3569–3649. North Holland, Amsterdam (2001)

Chib, S., Greenberg, E.: Bayes inference for regression models with ARMA(p, q) errors. J. Econometrics **64**, 183–206 (1994)

Chib, S., Greenberg, E.: Understanding the Metropolis–Hastings algorithm. Am. Stat. **49**, 327–335 (1995)

Chib, S., Greenberg, E.: Markov chain Monte Carlo simulation methods in econometrics. Economet. Theor. **12**, 409–431 (1996)

Chib, S., Greenberg, E., Winklemann, R.: Posterior simulation and Bayes factors in panel count data models. J. Econometrics **86**, 33–54 (1998)

Chib, S., Jeliazkov, I.: Marginal likelihood from the Metropolis–Hastings output. J. Am. Stat. Assoc. **96**, 270–281 (2001)

Congdon, P.: Bayesian Statistical Modeling. Wiley, Chicester (20011)

Cowles, M.K., Carlin, B.: Markov chain Monte Carlo convergence diagnostics: A comparative review. J. Am. Stat. Assoc. **91**, 883–904 (1996)

Damien, P., Wakefield, J., Walker, S.: Gibbs Sampling for Bayesian nonconjugate and hierarchical models using auxiliary variables. J. Roy. Stat. Soc. B **61**, 331–344 (1999)

Gammerman, D.: Markov chain Monte Carlo: Stochastic Simulation for Bayesian Inference. Chapman and Hall, London (1997)

Gelfand, A.E., Smith, A.F.M.: Sampling-based approaches to calculating marginal densities. J. Am. Stat. Assoc. 85: 398–409 (1990)

Gelman, A., Rubin, D.B.: Inference from iterative simulation using multiple sequences. Stat. Sci. **4**, 457–472 (1992)

Gelman, A., Meng, X.L., Stern, H.S., Rubin, D.B.: Bayesian Data Analysis, (2nd edn.), Chapman and Hall, London (2003)

Geman, S., Geman, D.: Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. IEEE Trans. Pattern Anal. Mach. Intell. **12**, 609–628 (1984)

Geweke, J.: Bayesian inference in econometric models using Monte Carlo integration. Econometrica **57**, 1317–1340 (1989)

Geweke, J.: Evaluating the accuracy of sampling-based approaches to the calculation of posterior moments. In: Bernardo, J.M., Berger, J.O., Dawid, A.P., Smith, A.F.M. (eds.) Bayesian Statistics, pp. 169–193, Oxford University Press, New York (1992)
Geyer, C., Thompson, E.A.: Annealing markov chain monte carlo with applications to ancestral inference. J. Am. Stat. Assoc. **90**, 909–920 (1995)
Gilks, W.R., Richardson, S., Spiegelhalter, D.J.: Markov Chain Monte Carlo in Practice, Chapman and Hall, London (1996)
Hastings, W.K.: Monte Carlo sampling methods using Markov chains and their applications. Biometrika **57**, 97–109 (1970)
Liu, J.S.: Monte Carlo Strategies in Scientific Computing, Springer, New York (2001)
Liu, J.S., Wong, W.H., Kong, A.: Covariance structure of the gibbs sampler with applications to the comparisons of estimators and data augmentation schemes. Biometrika **81**, 27–40 (1994)
Liu, J.S., Wong, W.H., Kong, A.: Covariance structure and convergence rate of the Gibbs sampler with various scans. J. Roy. Stat. Soc. B **57**, 157–169 (1995)
Marinari, E., Parsi, G.: Simulated tempering: A new Monte Carlo scheme. Europhys. Lett. **19**, 451–458 (1992)
Mengersen, K.L., Tweedie, R.L.: Rates of convergence of the Hastings and Metropolis algorithms. Ann. Stat. **24**, 101–121 (1996)
Metropolis, N., Rosenbluth, A.W., Rosenbluth, M.N., Teller, A.H., Teller, E.: Equations of state calculations by fast computing machines. J. Chem. Phys. **21**, 1087–1092 (1953)
Meyn, S.P., Tweedie, R.L.: Markov Chains and Stochastic Stability. Springer, London (1993)
Mira, A., Tierney, L.: Efficiency and convergence properties of slice samplers. Scand. J. Stat. **29**, 1–12 (2002)
Nummelin, E.: General Irreducible Markov Chains and Non-negative Operators. Cambridge University Press, Cambridge (1984)
Polson, N.G.: Convergence of Markov Chain Monte Carlo algorithms. In: Bernardo, J.M., Berger, J.O., Dawid, A.P., Smith, A.F.M. (eds.) Proceedings of the Fifth Valencia International Conference on Bayesian Statistics. pp. 297–323. Oxford University Press, Oxford (1996)
Propp, J.G., Wilson, D.B.: Exact sampling with coupled Markov chains and applications to statistical mechanics. Random Struct. Algorithm. **9**, 223–252 (1996)
Raftery, A.E., Lewis, S.M.: How many iterations in the Gibbs sampler? In: Bernardo, J.M., Berger, J.O., Dawid, A.P., Smith, A.F.M. (eds.) Proceedings of the Fourth Valencia International Conference on Bayesian Statistics. pp. 763–774. Oxford University Press, New York (1992)
Ripley, B.: Stochastic Simulation. Wiley, New York (1987)
Ritter, C, Tanner, M.A.: Facilitating the gibbs sampler: The gibbs stopper and the Griddy-Gibbs sampler. J. Am. Stat. Assoc. **87**, 861–868 (1992)
Robert C.P.: Convergence control methods for Markov chain Monte Carlo algorithms. Stat. Sci. **10**, 231–253 (1995)
Robert, C.P.: Bayesian Choice. (2nd ed.), Springer, New York (2001)
Robert, C.P., Casella, G.: Monte Carlo Statistical Methods. Springer, New York (1999)
Roberts, G.O., Rosenthal, J.S.: Convergence of slice sampler Markov chains. J. Roy. Stat. Soc. B **61**, 643–660 (1999)
Roberts, G.O., Sahu, S.K.: Updating schemes, correlation structure, blocking, and parametization for the Gibbs sampler. J. Roy. Stat. Soc. B **59**, 291–317 (1997)
Roberts, G.O., Smith, A.F.M.: Some simple conditions for the convergence of the Gibbs sampler and Metropolis–Hastings algorithms. Stochas. Process. Appls. **49**, 207–216 (1994)
Roberts, G.O., Tweedie, R.L.: Geometric convergence and central limit theorems for multidimensional Hastings and Metropolis algorithms. Biometrika **83**, 95–110 (1996)
Rosenthal, J.S.: Minorization conditions and convergence rates for Markov chain Monte Carlo. J. Am. Stat. Assoc. **90**, 558–566 (1995)
Smith, A.F.M., Roberts, G.O.: Bayesian computation via the Gibbs sampler and related Markov chain Monte Carlo methods. J. Roy. Stat. Soc. B **55**, 3–24 (1993)
Tanner, M.A.: Tools for Statistical Inference, (3rd edn.), Springer, New York (1996)

Tanner, M.A., Wong, W.H.: The calculation of posterior distributions by data augmentation. J. Am. Stat. Assoc. **82**, 528–549 (1987)
Tierney, L.: Markov chains for exploring posterior distributions (with discussion). Ann. Stat. **22**, 1701–1762 (1994)
Zellner, A., Min, C.: Gibbs sampler convergence criteria. J. Am. Stat. Assoc. **90**, 921–927 (1995)

Chapter 5
Numerical Linear Algebra

Lenka Čížková and Pavel Čížek

Many methods of computational statistics lead to matrix-algebra or numerical-mathematics problems. For example, the least squares method in linear regression reduces to solving a system of linear equations, see Chap. III.8. The principal components method is based on finding eigenvalues and eigenvectors of a matrix, see Chap. III.6. Nonlinear optimization methods such as Newton's method often employ the inversion of a Hessian matrix. In all these cases, we need numerical linear algebra.

Usually, one has a data matrix $\boldsymbol{X}$ of (explanatory) variables, and in the case of regression, a data vector $\boldsymbol{y}$ for dependent variable. Then the matrix defining a system of equations, being inverted or decomposed typically corresponds to $\boldsymbol{X}$ or $\boldsymbol{X}^\top \boldsymbol{X}$. We refer to the matrix being analyzed as $\boldsymbol{A} = \{A_{ij}\}_{i=1,j=1}^{m,n} \in \mathbb{R}^{m\times n}$ and to its columns as $\boldsymbol{A}_k = \{A_{ik}\}_{i=1}^{m}, k = 1,\ldots,n$. In the case of linear equations, $\boldsymbol{b} = \{b_i\}_{i=1}^{n} \in \mathbb{R}^n$ represents the right-hand side throughout this chapter. Further, the eigenvalues and singular values of $\boldsymbol{A}$ are denoted by λ_i and σ_i, respectively, and the corresponding eigenvectors $\boldsymbol{g}_i, i = 1,\ldots,n$. Finally, we denote the $n\times n$ identity and zero matrices by $\boldsymbol{I_n}$ and $\boldsymbol{0_n}$, respectively.

In this chapter, we first study various matrix decompositions (Sect. 5.1), which facilitate numerically stable algorithms for solving systems of linear equations and matrix inversions. Next, we discuss specific direct and iterative methods for solving linear systems (Sects. 5.2 and 5.3). Further, we concentrate on finding eigenvalues and eigenvectors of a matrix in Sect. 5.4. Finally, we provide an overview of numerical methods for large problems with sparse matrices (Sect. 5.5).

Let us note that most of the mentioned methods work under specific conditions given in existence theorems, which we state without proofs. Unless said otherwise, the proofs can be found in Harville (1997), for instance. Moreover, implementations of the algorithms described here can be found, for example, in Anderson et al. (1999) and Press et al. (1992).

L. Čížková · P. Čížek (✉)
Department of Econometrics & Operations Research,
Tilburg University, Tilburg, The Netherlands
e-mail: P.Cizek@uvt.nl; lenka@lenka-photography.eu

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_5,

5.1 Matrix Decompositions

This section covers relevant matrix decompositions and basic numerical methods. Decompositions provide a numerically stable way to solve a system of linear equations, as shown already in Wampler (1970), and to invert a matrix. Additionally, they provide an important tool for analyzing the numerical stability of a system.

Some of most frequently used decompositions are the Cholesky, QR, LU, and SVD decompositions. We start with the Cholesky and LU decompositions, which work only with positive definite and nonsingular diagonally dominant square matrices, respectively (Sects. 5.1.1 and 5.1.2). Later, we explore more general and in statistics more widely used QR and SVD decompositions, which can be applied to any matrix (Sects. 5.1.3 and 5.1.4). Finally, we briefly describe the use of decompositions for matrix inversion, although one rarely needs to invert a matrix (Sect. 5.1.5). Monographs Gentle (1998), Harville (1997), Higham (2002) and Stewart (1998) include extensive discussions of matrix decompositions.

All mentioned decompositions allow us to transform a general system of linear equations to a system with an upper triangular, a diagonal, or a lower triangular coefficient matrix: $\boldsymbol{U}\boldsymbol{x} = \boldsymbol{b}$, $\boldsymbol{D}\boldsymbol{x} = \boldsymbol{b}$, or $\boldsymbol{L}\boldsymbol{x} = \boldsymbol{b}$, respectively. Such systems are easy to solve with a very high accuracy by back substitution, see Higham (1989). Assuming the respective coefficient matrix $\boldsymbol{A}$ has a full rank, one can find a solution of $\boldsymbol{U}\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{U} = \{U_{ij}\}_{i=1,j=1}^{n}$, by evaluating

$$x_i = U_{ii}^{-1}\left(b_i - \sum_{j=i+1}^{n} U_{ij}x_j\right) \tag{5.1}$$

for $i = n, \ldots, 1$. Analogously for $\boldsymbol{L}\boldsymbol{x} = \boldsymbol{b}$, where $\boldsymbol{L} = \{L_{ij}\}_{i=1,j=1}^{n}$, one evaluates for $i = 1, \ldots, n$

$$x_i = L_{ii}^{-1}\left(b_i - \sum_{j=1}^{i-1} L_{ij}x_j\right). \tag{5.2}$$

For discussion of systems of equations that do not have a full rank, see for example Higham (2002).

5.1.1 Cholesky Decomposition

The Cholesky factorization, first published by Benoit (1924), was originally developed to solve least squares problems in geodesy and topography. This factorization, in statistics also referred to as "square root method," is a triangular decomposition. Providing matrix $\boldsymbol{A}$ is positive definite, the Cholesky decomposition finds a triangular matrix $\boldsymbol{U}$ that multiplied by its own transpose leads back to matrix $\boldsymbol{A}$. That is, $\boldsymbol{U}$ can be thought of as a square root of $\boldsymbol{A}$.

Theorem 1. *Let matrix* $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ *be symmetric and positive definite. Then there exists a unique upper triangular matrix* $\boldsymbol{U}$ *with positive diagonal elements such that* $\boldsymbol{A} = \boldsymbol{U}^\top \boldsymbol{U}$.

The matrix $\boldsymbol{U}$ is called the Cholesky factor of $\boldsymbol{A}$ and the relation $\boldsymbol{A} = \boldsymbol{U}^\top \boldsymbol{U}$ is called the Cholesky factorization.

Obviously, decomposing a system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ to $\boldsymbol{U}^\top \boldsymbol{U}\boldsymbol{x} = \boldsymbol{b}$ allows us to solve two triangular systems: $\boldsymbol{U}^\top \boldsymbol{z} = \boldsymbol{b}$ for $\boldsymbol{z}$ and then $\boldsymbol{U}\boldsymbol{x} = \boldsymbol{z}$ for $\boldsymbol{x}$. This is similar to the original Gauss approach for solving a positive definite system of normal equations $\boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{x} = \boldsymbol{X}^\top \boldsymbol{b}$. Gauss solved the normal equations by a symmetry-preserving elimination and used the back substitution to solve for $\boldsymbol{x}$.

Let us now describe the algorithm for finding the Cholesky decomposition, which is illustrated on Fig. 5.1. One of the interesting features of the algorithm is that in the ith iteration we obtain the Cholesky decomposition of the ith leading principal minor of $\boldsymbol{A}$, $\{A_{kl}\}_{k=1,l=1}^{i,i}$.

Algorithm 4

```
for i=1 to n
    U_ii = ( A_ii − Σ_{k=1}^{i−1} U_ki^2 )^{1/2}
    for j=i+1 to n
        U_ij = ( A_ij − Σ_{k=1}^{i−1} U_ki U_kj ) / U_ii
    end
end
```

The Cholesky decomposition described in Algorithm 4 is a numerically stable procedure, see Martin et al. (1965) and Meinguet (1983), which can be at the same time implemented in a very efficient way. Computed values U_{ij} can be stored

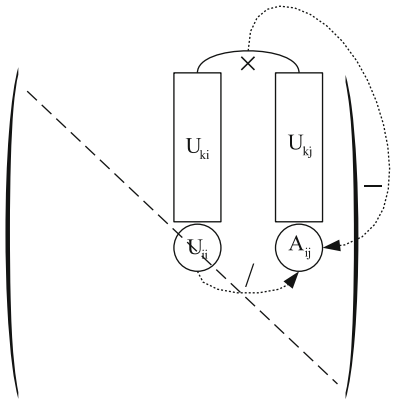


Fig. 5.1 Rowwise Cholesky algorithm

in place of original A_{ij}, and thus, no extra memory is needed. Moreover, let us note that while Algorithm 4 describes the rowwise decomposition ($\boldsymbol{U}$ is computed row by row), there are also a columnwise version and a version with diagonal pivoting, which is also applicable to semidefinite matrices. Finally, there are also modifications of the algorithm, such as blockwise decomposition, that are suitable for very large problems and parallelization; see Bjorck (1996), Gallivan et al. (1990) and Nool (1995).

5.1.2 LU Decomposition

The LU decomposition is another method reducing a square matrix $\boldsymbol{A}$ to a product of two triangular matrices (lower triangular $\boldsymbol{L}$ and upper triangular $\boldsymbol{U}$). Contrary to the Cholesky decomposition, it does not require a positive definite matrix $\boldsymbol{A}$, but there is no guarantee that $\boldsymbol{L} = \boldsymbol{U}^\top$.

Theorem 2. *Let the matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ satisfy following conditions:*

$$A_{11} \neq 0\,, \quad \det \begin{pmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{pmatrix} \neq 0\,, \quad \det \begin{pmatrix} A_{11} & A_{12} & A_{13} \\ A_{21} & A_{22} & A_{23} \\ A_{31} & A_{32} & A_{33} \end{pmatrix}$$

$$\neq 0\,, \quad \ldots\,, \quad \det \boldsymbol{A} \neq 0\,.$$

Then there exists a unique lower triangular matrix $\boldsymbol{L}$ with ones on a diagonal, a unique upper triangular matrix $\boldsymbol{U}$ with ones on a diagonal and a unique diagonal matrix $\boldsymbol{D}$ such that $\boldsymbol{A} = \boldsymbol{L}\boldsymbol{D}\boldsymbol{U}$.

Note that for any nonsingular matrix $\boldsymbol{A}$ there is always a row permutation $\boldsymbol{P}$ such that the permuted matrix $\boldsymbol{P}\boldsymbol{A}$ satisfies the assumptions of Theorem 2. Further, a more frequently used version of this theorem factorizes $\boldsymbol{A}$ to a lower triangular matrix $\boldsymbol{L}' = \boldsymbol{L}\boldsymbol{D}$ and an upper triangular matrix $\boldsymbol{U}' = \boldsymbol{U}$. Finally, Zou (1991) gave alternative conditions for the existence of the LU decomposition: $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ is nonsingular and $\boldsymbol{A}^\top$ is diagonally dominant (i.e., $|A_{ii}| \geq \sum_{i=1, i \neq j}^{n} |A_{ij}|$ for $j = 1, \ldots, n$).

Similarly to the Cholesky decomposition, the LU decomposition reduces solving a system of linear equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{L}\boldsymbol{U}\boldsymbol{x} = \boldsymbol{b}$ to solving two triangular systems: $\boldsymbol{L}\boldsymbol{z} = \boldsymbol{b}$, where $\boldsymbol{z} = \boldsymbol{U}\boldsymbol{x}$, and $\boldsymbol{U}\boldsymbol{x} = \boldsymbol{z}$.

Finding the LU decomposition of $\boldsymbol{A}$ is described in Algorithm 5. Since it is equivalent to solving a system of linear equations by the Gauss elimination, which searches just for $\boldsymbol{U}$ and ignores $\boldsymbol{L}$, we refer a reader to Sect. 5.2, where its advantages (e.g., easy implementation, speed) and disadvantages (e.g., numerical instability without pivoting) are discussed.

Algorithm 5

$\boldsymbol{L} = \boldsymbol{0}_n, \boldsymbol{U} = \boldsymbol{I}_n$
`for i = 1 to n`
  `for j = i to n`
    $L_{ji} = A_{ji} - \sum_{k=1}^{i-1} L_{jk} U_{ki}$
  `end`
  `for j = i + 1 to n`
    $U_{ij} = \left(A_{ij} - \sum_{k=1}^{i-1} L_{ik} U_{kj}\right) / L_{ii}$
  `end`
`end`

Finally, note that there are also generalizations of LU to non-square and singular matrices, such as rank revealing LU factorization; see Pan (2000) and Miranian and Gu (2003).

5.1.3 QR Decomposition

One of the most important matrix transformations is the QR decomposition. It splits a general matrix $\boldsymbol{A}$ to an orthonormal matrix $\boldsymbol{Q}$, that is, a matrix with columns orthogonal to each other and its Euclidian norm equal to 1, and to an upper triangular matrix $\boldsymbol{R}$. Thus, a suitably chosen orthogonal matrix $\boldsymbol{Q}$ will triangularize the given matrix $\boldsymbol{A}$.

Theorem 3. *Let matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}$ with $m \geq n$. Then there exist an orthonormal matrix $\boldsymbol{Q} \in \mathbb{R}^{m \times m}$ and an upper triangular matrix $\boldsymbol{R} \in \mathbb{R}^{n \times n}$ with nonnegative diagonal elements such that*

$$\boldsymbol{A} = \boldsymbol{Q} \begin{pmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{pmatrix}$$

(the QR decomposition of the matrix $\boldsymbol{A}$).

If $\boldsymbol{A}$ is a nonsingular square matrix, an even slightly stronger result can be obtained: uniqueness of the QR decomposition.

Theorem 4. *Let matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ be nonsingular. Then there exist a unique orthonormal matrix $\boldsymbol{Q} \in \mathbb{R}^{n \times n}$ and a unique upper triangular matrix $\boldsymbol{R} \in \mathbb{R}^{n \times n}$ with positive diagonal elements such that $\boldsymbol{A} = \boldsymbol{Q}\boldsymbol{R}$.*

The use of the QR decomposition for solving a system of equations $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{Q}\boldsymbol{R}\boldsymbol{x} = \boldsymbol{b}$ consists in multiplying the whole system by the orthonormal matrix $\boldsymbol{Q}^\top$, $\boldsymbol{Q}^\top\boldsymbol{Q} = \boldsymbol{I}$, and then solving the remaining upper triangular system $\boldsymbol{R}\boldsymbol{x} = \boldsymbol{Q}^\top\boldsymbol{b}$.

This method guarantees numerical stability by minimizing errors caused by machine roundoffs (see the end of this section for details).

The QR decomposition is usually constructed by finding one orthonormal vector (one column of $\boldsymbol{Q}$) after another. This can be achieved using the so-called elementary orthogonal transformations such as Householder reflections, Householder (1958), or Givens rotations, Givens (1958), that are described in the following subsections. These transformations are related to the solution of the following standard task.

Problem 1. Given a vector $\boldsymbol{x} \in \mathbb{R}^m$, $\boldsymbol{x} \neq 0$, find an orthogonal matrix $\boldsymbol{M} \in \mathbb{R}^{m \times m}$ such that $\boldsymbol{M}^\top \boldsymbol{x} = \|\boldsymbol{x}\|_2 \cdot \boldsymbol{e}_1$, where $\boldsymbol{e}_1 = (1, 0, \ldots, 0)^\top$ denotes the first unit vector.

In the rest of this section, we will first discuss how Householder reflections and Givens rotations can be used for solving Problem 1. Next, using these elementary results, we show how one can construct the QR decomposition. Finally, we briefly mention the Gram–Schmidt orthogonalization method, which also provides a way to find the QR decomposition.

Householder Reflections

The QR decomposition using Householder reflections (HR) was developed by Golub (1965). Householder reflection (or Householder transformation) is a matrix $\boldsymbol{P}$,

$$\boldsymbol{P} = \boldsymbol{I} - \frac{1}{c}\boldsymbol{u}\boldsymbol{u}^\top, \quad c = \frac{1}{2}\boldsymbol{u}^\top \boldsymbol{u}\ , \tag{5.3}$$

where $\boldsymbol{u}$ is a Householder vector. By definition, the matrix $\boldsymbol{P}$ is orthonormal and symmetric. Moreover, for any $\boldsymbol{x} \in \mathbb{R}^m$, it holds that

$$\boldsymbol{P}\boldsymbol{x} = \boldsymbol{x} - \frac{1}{c}\left(\boldsymbol{u}^\top \boldsymbol{x}\right)\boldsymbol{u}\ .$$

Therefore, to apply HR one does not need to explicitly compute the matrix $\boldsymbol{P}$ itself. Additionally, it holds $\boldsymbol{P}\boldsymbol{u} = -\boldsymbol{u}$ and $\boldsymbol{P}\boldsymbol{x} \in \text{span}\{\boldsymbol{x}, \boldsymbol{u}\}$. This means that HR reflects a vector $\boldsymbol{x}$ with respect to the hyperplane with normal vector, see Fig. 5.2. $\boldsymbol{u}$ (hence the name Householder reflection).

To solve Problem 1 using some HR, we search for a vector $\boldsymbol{u}$ such that $\boldsymbol{x}$ will be reflected to the x-axis. This holds for the following choice of $\boldsymbol{u}$:

$$\boldsymbol{u} = \boldsymbol{x} + s_1 \|\boldsymbol{x}\|_2 \cdot \boldsymbol{e}_1\ , \quad s_1 = 2I(x_1 \geq 0) - 1\ , \tag{5.4}$$

where $x_1 = \boldsymbol{x}^\top \boldsymbol{e}_1$ denotes the first element of the vector $\boldsymbol{x}$ and $I(\cdot)$ represents an indicator. For this reflection, it holds that c from (5.3) equals $\|x\|_2(\|x\|_2 + |x_1|)$ and $\boldsymbol{P}\boldsymbol{x} = -s_1\|\boldsymbol{x}\|_2 \cdot \boldsymbol{e}_1$ as one can verify by substituting (5.4) into (5.3).

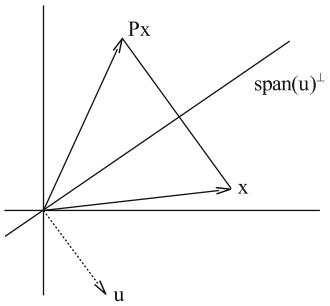


Fig. 5.2 Reflection with respect to the hyperplane with a normal vector $\boldsymbol{u}$

Givens Rotations

A Givens rotation (GR) in m dimensions (or Givens transformation) is defined by an orthonormal matrix $\boldsymbol{R}_{ij}(\alpha) \in \mathbb{R}^{m\times m}$,

$$\boldsymbol{R}_{ij}(\alpha) = \begin{pmatrix} 1 & 0 & \cdots & \cdots & \cdots & \cdots & 0 \\ 0 & \ddots & & & & & \vdots \\ \vdots & & c & & s & & \vdots \\ \vdots & & & \ddots & & & \vdots \\ \vdots & & -s & & c & & \vdots \\ \vdots & & & & & \ddots & 0 \\ 0 & \cdots & \cdots & \cdots & \cdots & 0 & 1 \end{pmatrix} \begin{matrix} \\ \\ i \\ \\ j \\ \\ \end{matrix}, \tag{5.5}$$

where $c = \cos\alpha$ and $s = \sin\alpha$ for $\alpha \in \mathbb{R}$ and $1 \le i < j \le n$. Thus, the rotation $\boldsymbol{R}_{ij}(\alpha)$ represents a plane rotation in the space spanned by the unit vectors $\boldsymbol{e}_i$ and $\boldsymbol{e}_j$ by an angle α. In two dimensions, rotation $\boldsymbol{R}_{12}(\alpha)$,

$$\boldsymbol{R}_{12}(\alpha) = \begin{pmatrix} c & s \\ -s & c \end{pmatrix}, \quad c = \cos\alpha, \quad s = \sin\alpha,$$

represents a clockwise rotation by an angle α; see Fig. 5.3.

Now, let us have a look at how GRs can be used for solving Problem 1. A GR of a vector $\boldsymbol{x} = (x_1, \ldots, x_m)^\top \in \mathbb{R}^m$ by an angle α results in $\boldsymbol{R}_{ij}(\alpha)\boldsymbol{x} = \boldsymbol{y} = (y_1, \ldots, y_m)^\top$ such that

$$y_k = \begin{cases} x_k & \text{for} \quad k \neq i, j, \\ cx_i + sx_j & \text{for} \quad k = i, \\ -sx_i + cx_j & \text{for} \quad k = j. \end{cases}$$

For a vector $\boldsymbol{x}$ with nonzero elements x_i or x_j, setting $d = (x_i^2 + x_j^2)^{1/2}$, $c = x_i/d$, $s = x_j/d$ leads to

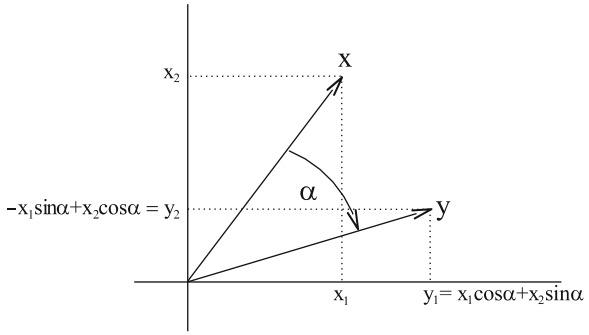


Fig. 5.3 Rotation of $\boldsymbol{x}$ in a plane by an angle α

$$\begin{pmatrix} c & s \\ -s & c \end{pmatrix} \begin{pmatrix} x_i \\ x_j \end{pmatrix} = \begin{pmatrix} d \\ 0 \end{pmatrix}.$$

Thus, using GR with this specific choice of c and s (referred further as $\boldsymbol{R}^0_{ij}$) implies that the jth component of the vector $\boldsymbol{x}$ vanishes. Similarly to HRs, it is not necessary to explicitly construct the whole matrix $\boldsymbol{P}$ to transform $\boldsymbol{x}$ since the rotation is fully described by only two numbers: c and s. This elementary rotation $\boldsymbol{R}^0_{ij}$ does not however constitute a solution to Problem 1 yet: we need to combine more of them.

The next step employs a simple fact that the pre- or postmultiplication of a vector $\boldsymbol{x}$ or a matrix $\boldsymbol{A}$ by any GR $\boldsymbol{R}_{ij}(\alpha)$ affects only the ith and jth rows and columns, respectively. Hence, one can combine several rotations without one rotation spoiling the result of another rotation. (Consequently, GRs are more flexible than HRs). Two typical ways how GRs are used for solving Problem 1 mentioned in Sect. 5.1.3 follow:

1. $\boldsymbol{R}^0_{1n}\boldsymbol{R}^0_{1,n-1}\ldots\boldsymbol{R}^0_{13}\boldsymbol{R}^0_{12}\boldsymbol{x} = d\boldsymbol{e}_1$. Here the kth component of the vector $\boldsymbol{x}$ vanishes after the Givens rotation $\boldsymbol{R}^0_{1k}$. The previously zeroed elements $x_2, \ldots, x_{k-1}$ are not changed because rotation $\boldsymbol{R}_{1k}$ affects only the first and kth component.
2. $\boldsymbol{R}^0_{12}\boldsymbol{R}^0_{23}\ldots\boldsymbol{R}^0_{n-1,n}\boldsymbol{x} = d\boldsymbol{e}_1$. Here the kth component vanishes by the rotation $R_{k-1,k}$.

Finally, there are several algorithms for computing the Givens rotations that improve over the straightforward evaluation of $\boldsymbol{R}^0_{ij}\boldsymbol{x}$. A robust algorithm minimizing the loss of precision is given in Bjorck (1996). An algorithm minimizing memory requirements was proposed by Stewart (1976). On the other hand, Gentleman (1973) and Hammarling (1974) proposed modifications aiming to minimize the number of arithmetic operations.

QR Decomposition by Householder Reflections or Givens Rotations

An appropriate combination of HRs or GRs, respectively, can be used to compute the QR decomposition of a given matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$, $m \geq n$, in a following

way. Let $\boldsymbol{Q_i}$, $i = 1, \ldots, n-1$, denote an orthonormal matrix in $\mathbb{R}^{m \times m}$ such that premultiplication of $\boldsymbol{B} = \boldsymbol{Q_{i-1}} \cdots \boldsymbol{Q_1} \boldsymbol{A}$ by $\boldsymbol{Q_i}$ can zero all elements in the ith column that are below the diagonal and such that the previous columns $1, \ldots, i-1$ are not affected at all. Such a matrix can be a blockwise diagonal matrix with blocks being the identity matrix $\boldsymbol{I_{i-1}}$ and a matrix $\boldsymbol{M}$ solving Problem 1 for the vector composed of elements in the ith column of $\boldsymbol{B}$ that lie on and below the diagonal. The first part $\boldsymbol{I_{i-1}}$ guarantees that the columns $1, \ldots, i-1$ of matrix $\boldsymbol{B}$ are not affected by multiplication, whereas the second block $\boldsymbol{M}$ transforms all elements in the ith column that are below the diagonal to zero. Naturally, matrix $\boldsymbol{M}$ can be found by means of HRs or GRs as described in previous paragraphs.

This way, we construct a series of matrices $\boldsymbol{Q_1}, \ldots, \boldsymbol{Q_n}$ such that

$$\boldsymbol{Q_n} \cdots \boldsymbol{Q_1} \boldsymbol{A} = \begin{pmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{pmatrix}.$$

Since all matrices $\boldsymbol{Q_1}, \ldots, \boldsymbol{Q_n}$ are orthonormal, $\boldsymbol{Q_t} = \boldsymbol{Q_n} \cdots \boldsymbol{Q_1}$ is also orthonormal and its inverse equals its transpose: $\boldsymbol{Q_t}^{-1} = \boldsymbol{Q_t}^{\top}$. Hence,

$$\boldsymbol{A} = (\boldsymbol{Q_n} \cdots \boldsymbol{Q_1})^{\top} \begin{pmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{pmatrix} = \boldsymbol{Q} \begin{pmatrix} \boldsymbol{R} \\ \boldsymbol{0} \end{pmatrix}$$

as described in Theorem 3.

We describe now the QR algorithm using HRs or GRs. Let $\boldsymbol{M}(\boldsymbol{x})$ denote the orthonormal matrix from Problem 1 constructed for a vector $\boldsymbol{x}$ by one of the discussed methods.

Algorithm 6

$\boldsymbol{Q} = \boldsymbol{I_m}$
$\boldsymbol{R} = \boldsymbol{A}$
`for i = 1 to n`
  $\boldsymbol{x} = \{R_{ki}\}_{k=i}^{m}$
  $\boldsymbol{Q_i} = \begin{pmatrix} \boldsymbol{I_{i-1}} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{M}(\boldsymbol{x}) \end{pmatrix}$
  $\boldsymbol{Q} = \boldsymbol{Q_i}\,\boldsymbol{Q}$
  $\boldsymbol{R} = \boldsymbol{Q_i}\,\boldsymbol{R}$
`end`
$\boldsymbol{Q} = \boldsymbol{Q}^{\top}$
$\boldsymbol{R} = \{R_{ij}\}_{i=1,j=1}^{n,n}$

There are also modifications of this basic algorithm employing pivoting for better numerical performance and even revealing rank of the system, see Hong and Tan (1992) and Higham (2000) for instance. An error analysis of the QR decomposition by HRs and GRs are given by Gentleman (1975) and Higham (2000), respectively.

Gram–Schmidt Orthogonalization

Given a nonsingular matrix $\boldsymbol{A} \in \mathbb{R}^{m \times n}, m \geq n$, the Gram–Schmidt orthogonalization constructs a matrix $\boldsymbol{Q}$ such that the columns of $\boldsymbol{Q}$ are orthonormal to each other and span the same space as the columns of $\boldsymbol{A}$. Thus, $\boldsymbol{A}$ can be expressed as $\boldsymbol{Q}$ multiplied by another matrix $\boldsymbol{R}$, whereby the Gram–Schmidt orthogonalization process (GS) ensures that $\boldsymbol{R}$ is an upper triangular matrix. Consequently, GS can be used to construct the QR decomposition of a matrix $\boldsymbol{A}$. A survey of GS variants and their properties is given by Bjorck (1994).

The classical Gram–Schmidt (CGS) process constructs the orthonormal basis stepwise. The first column $\boldsymbol{Q}_1$ of $\boldsymbol{Q}$ is simply normalized $\boldsymbol{A}_1$. Having constructed a orthonormal base $\boldsymbol{Q}_{1:k} = \{\boldsymbol{Q}_1, \ldots, \boldsymbol{Q}_k\}$, the next column $\boldsymbol{Q}_{k+1}$ is proportional to $\boldsymbol{A}_{k+1}$ minus its projection to the space span$\{\boldsymbol{Q}_{1:k}\}$. Thus, $\boldsymbol{Q}_{k+1}$ is by its definition orthogonal to span$\{\boldsymbol{Q}_{1:k}\}$, and at the same time, the first k columns of $\boldsymbol{A}$ and $\boldsymbol{Q}$ span the same linear space. The elements of the triangular matrix $\boldsymbol{R}$ from Theorem 3 are then coordinates of the columns of $\boldsymbol{A}$ given the columns of $\boldsymbol{Q}$ as a basis.

Algorithm 7

```
for i = 1 to n
    for j = 1 to i - 1
        R_ji = Q_j^⊤ A_i
    end
    Q_i = A_i − Σ_{j=1}^{i−1} R_ji Q_j
    R_ii = (Q_i^⊤ Q_i)^{1/2}
    Q_i = Q_i / R_ii
end
```

Similarly to many decomposition algorithms, also CGS allows a memory efficient implementation since the computed orthonormal columns of $\boldsymbol{Q}$ can rewrite the original columns of $\boldsymbol{A}$. Despite this feature and mathematical correctness, the CGS algorithm does not always behave well numerically because numerical errors can very quickly accumulate. For example, an error made in computing $\boldsymbol{Q}_1$ affects $\boldsymbol{Q}_2$, errors in both of these terms (although caused initially just by an error in $\boldsymbol{Q}_1$) adversely influence $\boldsymbol{Q}_3$ and so on. Fortunately, there is a modified Gram–Schmidt (MGS) procedure, which prevents such an error accumulation by subtracting linear combinations of $\boldsymbol{Q}_k$ directly from $\boldsymbol{A}$ before constructing following orthonormal vectors. (Surprisingly, MGS is historically older than CGS.)

Apart from this algorithm (the row version of MGS), there are also a column version of MGS by Bjorck (1994) and MGS modifications employing iterative orthogonalization and pivoting by Dax (2000). Numerical superiority of MGS over CGS was experimentally established already by Rice (1966). This result is also theoretically supported by the GS error analysis in Bjorck (1994), who uncovered numerical equivalence of the QR decompositions done by MGS and HRs.

Algorithm 8

```
for i = 1 to n
    Q_i = A_i
    R_ii = (Q_i^T Q_i)^{1/2}
    Q_i = Q_i / R_ii
    for j = i + 1 to n
        R_ji = Q_i^T A_j
        A_j = A_j − R_ij Q_i
    end
end
```

5.1.4 Singular Value Decomposition

The singular value decomposition (SVD) plays an important role in numerical linear algebra and in many statistical techniques as well. Using two orthonormal matrices, SVD can diagonalize any matrix $\boldsymbol{A}$ and the results of SVD can tell a lot about (numerical) properties of the matrix. (This is closely related to the eigenvalue decomposition: any symmetric square matrix $\boldsymbol{A}$ can be diagonalized, $\boldsymbol{A} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^\top$, where $\boldsymbol{D}$ is a diagonal matrix containing the eigenvalues of $\boldsymbol{A}$ and $\boldsymbol{V}$ is an orthonormal matrix.)

Theorem 5. *Let $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ be a matrix of rank r. Then there exist orthonormal matrices $\boldsymbol{U} \in \mathbb{R}^{m\times m}$ and $\boldsymbol{V} \in \mathbb{R}^{n\times n}$ and a diagonal matrix $\boldsymbol{D} \in \mathbb{R}^{m\times n}$, with the diagonal elements $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_r > \sigma_{r+1} = \ldots = \sigma_{\min\{m,n\}} = 0$, such that $\boldsymbol{A} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top$.*

Numbers $\sigma_1, \ldots, \sigma_{\min\{m,n\}}$ represent the singular values of $\boldsymbol{A}$. Columns $\boldsymbol{U}_i$ and $\boldsymbol{V}_i$ of matrices $\boldsymbol{U}$ and $\boldsymbol{V}$ are called the left and right singular vectors of $\boldsymbol{A}$ associated with singular value σ_i, respectively, because $\boldsymbol{A}\boldsymbol{V}_i = \sigma_i \boldsymbol{U}_i$ and $\boldsymbol{U}_i^\top \boldsymbol{A} = \sigma_i \boldsymbol{V}_i^\top, i = 1, \ldots, \min\{m, n\}$.

Similarly to the QR decomposition, SVD offers a numerically stable way to solve a system of linear equations. Given a system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{U}\boldsymbol{D}\boldsymbol{V}^\top\boldsymbol{x} = \boldsymbol{b}$, one can transform it to $\boldsymbol{U}^\top\boldsymbol{A}\boldsymbol{x} = \boldsymbol{D}\boldsymbol{V}^\top\boldsymbol{x} = \boldsymbol{U}^\top\boldsymbol{b}$ and solve it in two trivial steps: first, finding a solution $\boldsymbol{z}$ of $\boldsymbol{D}\boldsymbol{z} = \boldsymbol{U}^\top\boldsymbol{b}$, and second, setting $\boldsymbol{x} = \boldsymbol{V}\boldsymbol{z}$, which is equivalent to $\boldsymbol{V}^\top\boldsymbol{x} = \boldsymbol{z}$.

On the other hand, the power of SVD lies in its relation to many important matrix properties; see Trefethen and Bau (1997), for instance. First of all, the singular values of a matrix $\boldsymbol{A}$ are equal to the (positive) square roots of the eigenvalues of $\boldsymbol{A}^\top\boldsymbol{A}$ and $\boldsymbol{A}\boldsymbol{A}^\top$, whereby the associated left and right singular vectors are identical with the corresponding eigenvectors. Thus, one can compute the eigenvalues of $\boldsymbol{A}^\top\boldsymbol{A}$ directly from the original matrix $\boldsymbol{A}$. Second, the number of nonzero singular values equals the rank of a matrix. Consequently, SVD can be used to find an effective rank of a matrix, to check a near singularity and to

compute the condition number of a matrix. That is, it allows to assess conditioning and sensitivity to errors of a given system of equations. Finally, let us note that there are far more uses of SVD: identification of the null space of $\boldsymbol{A}$, $\text{null}(\boldsymbol{A}) = \text{span}\{\boldsymbol{V}_{k+1}, \ldots, \boldsymbol{V}_n\}$; computation of the matrix pseudo-inverse, $\boldsymbol{A}^- = \boldsymbol{V}\boldsymbol{D}^-\boldsymbol{U}^\top$; low-rank approximations and so on. See Bjorck (1996) and Trefethen and Bau (1997) for details.

Let us now present an overview of algorithms for computing the SVD decomposition, which are not described in details due to their extent. The first stable algorithm for computing the SVD was suggested by Golub and Kahan (1965). It involved reduction of a matrix $\boldsymbol{A}$ to its bidiagonal form by HRs, with singular values and vectors being computed as eigenvalues and eigenvectors of a specific tridiagonal matrix using a method based on Sturm sequences. The final form of the QR algorithm for computing SVD, which has been the preferred SVD method for dense matrices up to now, is due to Golub and Reinsch (1970); see Anderson et al. (1999), Bjorck (1996) or Gentle (1998) for the description of the algorithm and some modifications. An alternative approach based on Jacobi algorithm was given by Hari and Veselic (1987). Latest contributions to the pool of computational methods for SVD, including von Matt (1995), Demmel et al. (1999) and Higham (2000), aim to improve the accuracy of singular values and computational speed using recent advances in the QR decomposition.

5.1.5 Matrix Inversion

In previous sections, we described how matrix decompositions can be used for solving systems of linear equations. Let us now discuss the use of matrix decompositions for inverting a nonsingular squared matrix $\boldsymbol{A} \in \mathbb{R}^{n \times n}$, although matrix inversion is not needed very often. All discussed matrix decomposition construct two or more matrices $\boldsymbol{A_1}, \ldots, \boldsymbol{A_d}$ such that $\boldsymbol{A} = \boldsymbol{A_1} \cdot \ldots \cdot \boldsymbol{A_d}$, where matrices $\boldsymbol{A_l}, l = 1, \ldots, d$, are orthonormal, triangular, or diagonal. Because $\boldsymbol{A}^{-1} = \boldsymbol{A_d}^{-1} \cdot \ldots \cdot \boldsymbol{A_1}^{-1}$, we just need to be able to invert orthonormal and triangular matrices (a diagonal matrix is a special case of a triangular matrix).

First, an orthonormal matrix $\boldsymbol{Q}$ satisfies by definition $\boldsymbol{Q}^\top\boldsymbol{Q} = \boldsymbol{Q}\boldsymbol{Q}^\top = \boldsymbol{I_n}$. Thus, inversion is in this case equivalent to the transposition of a matrix: $\boldsymbol{Q}^{-1} = \boldsymbol{Q}^\top$.

Second, inverting an upper triangular matrix $\boldsymbol{U}$ can be done by solving directly $\boldsymbol{X}\boldsymbol{U} = \boldsymbol{I_n}$, which leads to the backward substitution method. Let $\boldsymbol{X} = \{X_{ij}\}_{i=1,j=1}^{n,n}$ denote the searched for inverse matrix $\boldsymbol{U}^{-1}$.

The inversion of a lower triangular matrix $\boldsymbol{L}$ can be done analogously: the algorithm is applied to $\boldsymbol{L}^\top$, that is, U_{ij} is replaced by L_{ji} for $i, j = 1, \ldots, n$.

There are several other algorithms available such as forward substitution or blockwise inversion. Designed for a faster and more (time) efficient computation, their numerical behavior does not significantly differ from the presented algorithm. See Croz and Higham (1992) for an overview and numerical study.

Algorithm 9

$X = \mathbf{0}_n$
for i = n to 1
    $X_{ii} = 1/U_{ii}$
    for j = i + 1 to n
        $X_{ij} = -\left(\sum_{k=i+1}^{j} X_{kj} U_{ik}\right) / U_{jj}$
    end
end

5.2 Direct Methods for Solving Linear Systems

A system of linear equations can be written in the matrix notation as

$$\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}\ , \tag{5.6}$$

where $\boldsymbol{A}$ denotes the coefficient matrix, $\boldsymbol{b}$ is the right-hand side, and $\boldsymbol{x}$ represents the solution vector we search for. The system (5.6) has a solution if and only if $\boldsymbol{b}$ belongs to the vector space spanned by the columns of $\boldsymbol{A}$.

- If $m < n$, that is, the number of equations is smaller than the number of unknown variables, or if $m \geq n$ but $\boldsymbol{A}$ does not have a full rank (which means that some equations are linear combinations of the other ones), the system is underdetermined and there are either no solution at all or infinitely many of them. In the latter case, any solution can be written as a sum of a particular solution and a vector from the nullspace of $\boldsymbol{A}$. Finding the solution space can involve the SVD decomposition (Sect. 5.1.4).
- If $m > n$ and the matrix $\boldsymbol{A}$ has a full rank, that is, if the number of equations is greater than the number of unknown variables, there is generally no solution and the system is overdetermined. One can search some $\boldsymbol{x}$ such that the distance between $\boldsymbol{A}\,\boldsymbol{x}$ and $\boldsymbol{b}$ is minimized, which leads to the linear least-squares problem if distance is measured by L_2 norm; see Chap. III.8.
- If $m = n$ and the matrix $\boldsymbol{A}$ is nonsingular, the system (5.6) has a unique solution. Methods suitable for this case will be discussed in the rest of this section as well as in Sect. 5.3.

From here on, we concentrate on systems of equations with unique solutions.

There are two basic classes of methods for solving system (5.6). The first class is represented by direct methods. They theoretically give an exact solution in a (predictable) finite number of steps. Unfortunately, this does not have to be true in computational praxis due to rounding errors: an error made in one step spreads in all following steps. Classical direct methods are discussed in this section. Moreover, solving an equation system by means of matrix decompositions, as discussed in Sect. 5.1, can be classified as a direct method as well. The second class is called iterative methods, which construct a series of solution approximations that (under

some assumptions) converges to the solution of the system. Iterative methods are discussed in Sect. 5.3. Finally, note that some methods are on the borderline between the two classes; for example, gradient methods (Sect. 5.3.5) and iterative refinement (Sect. 5.2.2).

Further, the direct methods discussed in this section are not necessarily optimal for an arbitrary system (5.6). Let us deal with the main exceptions. First, even if a unique solution exist, numerical methods can fail to find the solution: if the number of unknown variables n is large, rounding errors can accumulate and result in a wrong solution. The same applies very much to systems with a nearly singular coefficient matrix. One alternative is to use iterative methods (Sect. 5.3), which are less sensitive to these problems. Another approach is to use the QR or SVD decompositions (Sect. 5.1), which can transform some nearly singular problems to nonsingular ones. Second, very large problems including hundreds or thousands of equations and unknown variables may be very time demanding to solve by standard direct methods. On the other hand, their coefficient matrices are often sparse, that is, most of their elements are zeros. Special strategies to store and solve such problems are discussed in Sect. 5.5.

To conclude these remarks, let us mention a close relation between solving the system (5.6) and computing the inverse matrix $\boldsymbol{A}^{-1}$:

- Having an algorithm that for a matrix $\boldsymbol{A}$ computes $\boldsymbol{A}^{-1}$, we can find the solution to (5.6) as $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$;
- An algorithm solving the system (5.6) can be used to compute $\boldsymbol{A}^{-1}$ as follows. Solve n linear systems $\boldsymbol{A}\boldsymbol{x}_i = \boldsymbol{e}_i$, $i = 1, \ldots, n$ (or the corresponding system with multiple right-hand sides), where $\boldsymbol{e}_i$ denotes the ith unit vector. Then $\boldsymbol{A}^{-1} = (\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n)$.

In the rest of this section, we concentrate on the Gauss–Jordan elimination (Sect. 5.2.1) and its modifications and extensions, such as iterative refinement (Sect. 5.2.2). A wealth of information on direct methods can be found in monographs Axelsson (1994), Gentle (1998) and Golub and van Loan (1996).

5.2.1 Gauss–Jordan Elimination

In this subsection, we will simultaneously solve the linear systems

$$\boldsymbol{A}\boldsymbol{x}_1 = \boldsymbol{b}_1\,, \quad \boldsymbol{A}\boldsymbol{x}_2 = \boldsymbol{b}_2\,, \quad \ldots\,, \quad \boldsymbol{A}\boldsymbol{x}_k = \boldsymbol{b}_k$$

and a matrix equation $\boldsymbol{A}\boldsymbol{X} = \boldsymbol{B}$, where $\boldsymbol{X}, \boldsymbol{B} \in \mathbb{R}^{n \times l}$ (its solution is $\boldsymbol{X} = \boldsymbol{A}^{-1}\boldsymbol{B}$, yielding the inverse $\boldsymbol{A}^{-1}$ for a special choice $\boldsymbol{B} = \boldsymbol{I}_n$). They can be written as a linear matrix equation

$$\boldsymbol{A}[\boldsymbol{x}_1|\boldsymbol{x}_2|\ldots|\boldsymbol{x}_k|\boldsymbol{X}] = [\boldsymbol{b}_1|\boldsymbol{b}_2|\ldots|\boldsymbol{b}_k|\boldsymbol{B}]\,, \tag{5.7}$$

where the operator | stands for column augmentation.

The Gauss–Jordan elimination (GJ) is based on elementary operations that do not affect the solution of an equation system. The solution of (5.7) will not change if we perform any of the following operations:

- Interchanging any two rows of $\boldsymbol{A}$ and the corresponding rows of $\boldsymbol{b}_i$'s and $\boldsymbol{B}$, $i = 1, \ldots, k$;
- Multiplying a row of $\boldsymbol{A}$ and the same row of $\boldsymbol{b}_i$'s and $\boldsymbol{B}$ by a nonzero number, $i = 1, \ldots, k$;
- Adding to a chosen row of $\boldsymbol{A}$ and the same row of $\boldsymbol{b}_i$'s and $\boldsymbol{B}$ a linear combination of other rows, $i = 1, \ldots, k$.

Interchanging any two columns of $\boldsymbol{A}$ is possible too, but it has to be followed by interchanging the corresponding rows of all solutions $\boldsymbol{x}_i$ and $\boldsymbol{X}$ as well as of right sides $\boldsymbol{b}_i$ and $\boldsymbol{B}$, $i = 1, \ldots, k$. Each row or column operation described above is equivalent to the pre- or postmultiplication of the system by a certain elementary matrix $\boldsymbol{R}$ or $\boldsymbol{C}$, respectively, that are results of the same operation applied to the identity matrix $\boldsymbol{I_n}$.

GJ is a technique that applies one or more of these elementary operations to (5.7) so that $\boldsymbol{A}$ becomes the identity matrix $\boldsymbol{I_n}$. Simultaneously, the right-hand side becomes the set of solutions. Denoting $\boldsymbol{R_i}, i = 1, \ldots, O$, the matrices corresponding to the ith row operation, the combination of all operations has to constitute inverse $\boldsymbol{A}^{-1} = \boldsymbol{R_O} \cdot \ldots \cdot \boldsymbol{R_3 R_2 R_1}$ and hence $\boldsymbol{x} = \boldsymbol{R_O} \cdot \ldots \cdot \boldsymbol{R_3 R_2 R_1 b}$. The exact choice of these elementary operation is described in the following paragraph.

Pivoting in Gauss–Jordan Elimination

Let us now discuss several well-known variants of the Gauss–Jordan elimination. GJ without pivoting does not interchange any rows or columns; only multiplication and addition of rows are permitted. First, nonzero nondiagonal elements in the first column $\boldsymbol{A}_1$ are eliminated: the first row of (5.7) is divided by its diagonal element A_{11} and the A_{i1}-multiple of the modified first row is subtracted from the ith row, $i = 2, \ldots, n$. We can proceed the same way for all n columns of $\boldsymbol{A}$, and thus, transform $\boldsymbol{A}$ to the identity matrix $\boldsymbol{I_n}$. It is easy to see that the method fails if the diagonal element in a column to be eliminated, the so-called pivot, is zero in some step. Even if this is not the case, one should be aware that GJ without pivoting is numerically unstable.

On the other hand, the GJ method becomes stable when using pivoting. This means that one can interchange rows (partial pivoting) or rows and columns (full pivoting) to put a suitable matrix element to the position of the current pivot. Since it is desirable to keep the already constructed part of the identify matrix, only rows below and columns right to the current pivot are considered. GJ with full pivoting is numerically stable. From the application point of view, GJ with partial pivoting is numerically stable too, although there are artificial examples where it fails. Additionally, the advantage of partial pivoting (compared to full pivoting) is that it does not change the order of solution components.

There are various strategies to choose a pivot. A very good choice is the largest available element (in absolute value). This procedure depends however on the original scaling of the equations. Implicit pivoting takes scaling into account and chooses a pivot as if the original system were rescaled so that the largest element of each row would be equal to one.

Finally, let us add several concluding remarks on efficiency of GJ and its relationship to matrix decompositions. As shown, GJ can efficiently solve problems with multiple right-hand sides known in advance and compute $\boldsymbol{A}^{-1}$ at the same time. On the other hand, if it is necessary to solve later a new system with the same coefficient matrix $\boldsymbol{A}$ but a new right-hand side $\boldsymbol{b}$, one has to start the whole elimination process again, which is time demanding, or compute $\boldsymbol{A}^{-1}\boldsymbol{b}$ using the previously computed inverse matrix $\boldsymbol{A}^{-1}$, which leads to further error accumulation. In praxis, one should prefer matrix decompositions, which do not have this drawback. Specifically, the LU decomposition (Sect. 5.1.2) is equivalent to GJ (with the same kind of pivoting applied in both cases) and allows us to repeatedly solve systems with the same coefficient matrix in an efficient way.

5.2.2 *Iterative Refinement*

In the introduction to Sect. 5.2, we noted that direct methods are rather sensitive to rounding errors. Iterative refinement offers a way to improve the solution obtained by any direct method, unless the system matrix $\boldsymbol{A}$ is too ill-conditioned or even singular.

Let $\boldsymbol{x}_1$ denote an initially computed (approximate) solution of (5.6). Iterative refinement is a process constructing a series $\boldsymbol{x}_i, i = 1, 2, \ldots$, as described in Algorithm 10. First, given a solution $\boldsymbol{x}_i$, the residuum $\boldsymbol{r}_i = \boldsymbol{A}\boldsymbol{x}_i - \boldsymbol{b}$ is computed. Then, one obtains the correction $\boldsymbol{\Delta x}_i$ by solving the original system with residuum $\boldsymbol{r}_i$ on the right-hand side. It is reasonable to carry out the computation of residuals $\boldsymbol{r}_i$ in

Algorithm 10

```
Repeat for i = 1, 2, ...
    compute r_i = b − A x_i
    solve   A Δx_i = r_i  for  Δx_i
    set     x_{i+1} = x_i + Δx_i
until the desired precision is achieved.
```

a higher precision because a lot of cancellation occurs if $\boldsymbol{x}_i$ is a good approximation. Nevertheless, provided that the coefficient matrix $\boldsymbol{A}$ is not too ill-conditioned, Skeel (1980) proved that GJ with partial pivoting and only one step of iterative refinement computed in a fixed precision is stable (it has a relative backward error proportional to the used precision). In spite of this result, one can recommend to use iterative refinement repeatedly until the desired precision is reached.

Additionally, an important feature of iterative refinement is its low computational costs. Provided that a system is solved by means of decompositions (e.g., GJ is implemented as the LU decomposition), a factorization of $\boldsymbol{A}$ is available already after computing the initial solution $\boldsymbol{x}_1$. Subsequently, solving any system with the same coefficient matrix $\boldsymbol{A}$, such as $\boldsymbol{A}\,\boldsymbol{\Delta x}_i = \boldsymbol{r}_i$, can be done fast and efficiently and the computational costs of iterative refinement are small.

5.3 Iterative Methods for Solving Linear Systems

Direct methods for solving linear systems theoretically give the exact solution in a finite number of steps, see Sect. 5.2. Unfortunately, this is rarely true in applications because of rounding errors: an error made in one step spreads further in all following steps! Contrary to direct methods, iterative methods construct a series of solution approximations such that it converges to the exact solution of a system. Their main advantage is that they are self-correcting, see Sect. 5.3.1.

In this section, we first discuss general principles of iterative methods that solve linear system (5.6), $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$, whereby we assume that $\boldsymbol{A} \in \mathbb{R}^{n\times n}$ and the system has exactly one solution $\boldsymbol{x}_e$ (see Sect. 5.2 for more details on other cases). Later, we describe most common iterative methods: the Jacobi, Gauss–Seidel, successive overrelaxation, and gradient methods (Sects. 5.3.2–5.3.5). Monographs containing detailed discussion of these methods include Bjorck (1996), Golub and van Loan (1996) and Hackbusch (1994). Although we treat these methods separately from the direct methods, let us mention here that iterative methods can usually benefit from a combination with the Gauss elimination, see Milaszewicz (1987) and Alanelli and Hadjidimos (2004), for instance.

To unify the presentation of all methods, let $\boldsymbol{D}$, $\boldsymbol{L}$, and $\boldsymbol{U}$ denote the diagonal, lower triangular and upper triangular parts of a matrix $\boldsymbol{A}$ throughout this section:

$$D_{ij} = \begin{cases} A_{ij} & \text{for} \quad i = j\ , \\ 0 & \text{otherwise}\ ; \end{cases} \qquad L_{ij} = \begin{cases} A_{ij} & \text{for} \quad i > j\ , \\ 0 & \text{otherwise}\ ; \end{cases}$$

$$U_{ij} = \begin{cases} A_{ij} & \text{for} \quad i < j\ , \\ 0 & \text{otherwise}\ . \end{cases}$$

5.3.1 General Principle of Iterative Methods for Linear Systems

An iterative method for solving a linear system $\boldsymbol{A}\,\boldsymbol{x} = \boldsymbol{b}$ constructs an iteration series $\boldsymbol{x}_i$, $i = 0, 1, 2, \ldots$, that under some conditions converges to the exact solution $\boldsymbol{x}_e$ of the system ($\boldsymbol{A}\,\boldsymbol{x}_e = \boldsymbol{b}$). Thus, it is necessary to choose a starting point $\boldsymbol{x}_0$ and iteratively apply a rule that computes $\boldsymbol{x}_{i+1}$ from an already known $\boldsymbol{x}_i$.

A starting vector $\boldsymbol{x}_0$ is usually chosen as some approximation of $\boldsymbol{x}$. (Luckily, its choice cannot cause divergence of a convergent method.) Next, given $\boldsymbol{x}_i, i \in \mathbb{N}$, the subsequent element of the series is computed using a rule of the form

$$\boldsymbol{x}_{i+1} = \boldsymbol{B}_i \boldsymbol{x}_i + \boldsymbol{C}_i \boldsymbol{b} \ , \quad i = 0, 1, 2, \dots \ , \tag{5.8}$$

where $\boldsymbol{B}_i, \boldsymbol{C}_i \in \mathbb{R}^{n \times n}, i \in \mathbb{N}$, are matrix series. Different choices of $\boldsymbol{B}_i$ and $\boldsymbol{C}_i$ define different iterative methods.

Let us discuss now a minimal set of conditions on $\boldsymbol{B}_i$ and $\boldsymbol{C}_i$ in (5.8) that guarantee the convergence of an iterative method. First of all, it has to hold that $\boldsymbol{B}_i + \boldsymbol{C}_i \boldsymbol{A} = \boldsymbol{I}_n$ for all $i \in \mathbb{N}$, or equivalently,

$$\boldsymbol{x}_e = \boldsymbol{B}_i \boldsymbol{x}_e + \boldsymbol{C}_i \boldsymbol{b} = (\boldsymbol{B}_i + \boldsymbol{C}_i \boldsymbol{A}) \boldsymbol{x}_e \ , \qquad i \in \mathbb{N} \ . \tag{5.9}$$

In other words, once the iterative process reaches the exact solution $\boldsymbol{x}_e$, all consecutive iterations should stay equal to $\boldsymbol{x}_e$ and the method cannot depart from this solution. Second, starting from a point $\boldsymbol{x}_0 \neq \boldsymbol{x}_e$, we have to ensure that approximations $\boldsymbol{x}_i$ will converge to $\boldsymbol{x}_e$ as i increases.

Theorem 6. *An iteration series $\boldsymbol{x}_i$ given by (5.8) converges to the solution of system (5.6) for any chosen $\boldsymbol{x}_0$ iff*

$$\lim_{i \to \infty} \boldsymbol{B}_i \boldsymbol{B}_{i-1} \dots \boldsymbol{B}_0 = \boldsymbol{0} \ .$$

In praxis, stationary iterative methods are used, that is, methods with constant $\boldsymbol{B}_i = \boldsymbol{B}$ and $\boldsymbol{C}_i = \boldsymbol{C}, i \in \mathbb{N}$. Consequently, an iteration series is then constructed using

$$\boldsymbol{x}_{i+1} = \boldsymbol{B} \boldsymbol{x}_i + \boldsymbol{C} \boldsymbol{b} \ , \quad i = 0, 1, 2, \dots \tag{5.10}$$

and the convergence condition in Theorem 6 has a simpler form.

Theorem 7. *An iteration series $\boldsymbol{x}_i$ given by (5.10) converges to the solution of system (5.6) for any chosen $\boldsymbol{x}_0$ iff the spectral radius $\rho(\boldsymbol{B}) < 1$, where $\rho(\boldsymbol{B}) = \max_{i=1,\dots,n} |\lambda_i|$ and $\lambda_1, \dots, \lambda_n$ represent the eigenvalues of $\boldsymbol{B}$.*

Note that the convergence condition $\rho(\boldsymbol{B}) < 1$ holds, for example, if $\|\boldsymbol{B}\| < 1$ in any matrix norm. Moreover, Theorem 7 guarantees the self-correcting property of iterative methods since convergence takes place independent of the starting value $\boldsymbol{x}_0$. Thus, if computational errors adversely affect $\boldsymbol{x}_i$ during the ith iteration, $\boldsymbol{x}_i$ can be considered as a new starting vector and the iterative method will further converge. Consequently, the iterative methods are in general more robust than the direct ones.

Apparently, such an iterative process can continue arbitrarily long unless $\boldsymbol{x}_i = \boldsymbol{x}_e$ at some point. This is impractical and usually unnecessary. Therefore, one uses stopping (or convergence) criteria that stop the iterative process when a pre-specified condition is met. Commonly used stopping criteria are based on the change of the solution or residual vector achieved during one iteration. Specifically, given a small $\varepsilon > 0$, the iterative process is stopped after the ith iteration when $\|\boldsymbol{x}_i - \boldsymbol{x}_{i-1}\| \leq \varepsilon$, $\|\boldsymbol{r}_i - \boldsymbol{r}_{i-1}\| \leq \varepsilon$, or $\|\boldsymbol{r}_i\| \leq \varepsilon$, where $\boldsymbol{r}_i = \boldsymbol{A}\boldsymbol{x}_i - \boldsymbol{b}$ is a residual vector. Additionally, a maximum acceptable number of iterations is usually specified.

5.3.2 Jacobi Method

The Jacobi method is motivated by the following observation. Let $\boldsymbol{A}$ have nonzero diagonal elements (the rows of any nonsingular matrix can be reorganized to achieve this). Then the diagonal part $\boldsymbol{D}$ of $\boldsymbol{A}$ is nonsingular and the system (5.6) can be rewritten as $\boldsymbol{D}\boldsymbol{x} + (\boldsymbol{L} + \boldsymbol{U})\boldsymbol{x} = \boldsymbol{b}$. Consequently,

$$\boldsymbol{x} = \boldsymbol{D}^{-1}[(-\boldsymbol{L} - \boldsymbol{U})\boldsymbol{x} + \boldsymbol{b}] \,.$$

Replacing $\boldsymbol{x}$ on the left-hand side by $\boldsymbol{x}_{i+1}$ and $\boldsymbol{x}$ on the right-hand side by $\boldsymbol{x}_i$ leads to the iteration formula of the Jacobi method:

$$\boldsymbol{x}_{i+1} = -\boldsymbol{D}^{-1}(\boldsymbol{L} + \boldsymbol{U})\boldsymbol{x}_i + \boldsymbol{D}^{-1}\boldsymbol{b} \,.$$

The intuition of the Jacobi method is very simple: given an approximation $\boldsymbol{x}^{\text{old}}$ of the solution, let us express the kth component x_k of $\boldsymbol{x}$ as a function of the other components from the kth equation and compute x_k given $\boldsymbol{x}^{\text{old}}$:

$$x_k^{\text{new}} = \frac{1}{A_{kk}} \left(b_k - \sum_{\substack{j=1 \\ j \neq k}}^{n} A_{kj} x_j^{\text{old}} \right) \,, \tag{5.11}$$

$k = 1, \ldots, n$ (see Fig. 5.4).

The Jacobi method converges for any starting vector $\boldsymbol{x}_0$ as long as $\rho(\boldsymbol{D}^{-1}(\boldsymbol{L} + \boldsymbol{U})) < 1$, see Theorem 7. This condition is satisfied for a relatively big class of matrices including diagonally dominant matrices (matrices $\boldsymbol{A}$ such that $\sum_{j=1, j\neq i}^{n} |A_{ij}| \leq |A_{ii}|$ for $i = 1, \ldots, n$), and symmetric matrices $\boldsymbol{A}$ such that $\boldsymbol{D}$, $\boldsymbol{A} = \boldsymbol{L} + \boldsymbol{D} + \boldsymbol{U}$, and $-\boldsymbol{L} + \boldsymbol{D} - \boldsymbol{U}$ are all positive definite. Although there are many improvements to the basic principle of the Jacobi method in terms of convergence to $\boldsymbol{x}_e$, see Sects. 5.3.3 and 5.3.4, its advantage is an easy and fast implementation (elements of a new iteration $\boldsymbol{x}_i$ can be computed independently of each other).

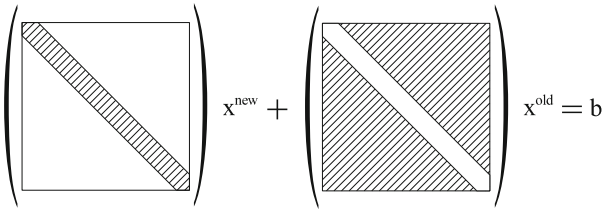


Fig. 5.4 Scheme of the Jacobi method

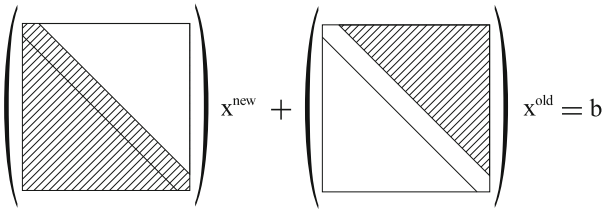


Fig. 5.5 Scheme of the Gauss–Seidel method

5.3.3 Gauss–Seidel Method

Analogously to the Jacobi method, we can rewrite system (5.6) as $(\boldsymbol{L}+\boldsymbol{D})\boldsymbol{x}+\boldsymbol{U}\boldsymbol{x} = \boldsymbol{b}$, which further implies $\boldsymbol{x} = (\boldsymbol{L}+\boldsymbol{D})^{-1}[-\boldsymbol{U}\boldsymbol{x}+\boldsymbol{b}]$. This leads to the iteration formula of the Gauss–Seidel method:

$$\boldsymbol{x}_{i+1} = -(\boldsymbol{L}+\boldsymbol{D})^{-1}\boldsymbol{U}\boldsymbol{x}_i + (\boldsymbol{L}+\boldsymbol{D})^{-1}\boldsymbol{b}\ . \tag{5.12}$$

The main difference to the Jacobi methods lies in a more efficient use of (5.11). When computing the kth element x_k^{new}, the first $k-1$ elements $x_1^{\text{new}}, \ldots, x_{k-1}^{\text{new}}$ are already known (and presumably more precise than $x_1^{\text{old}}, \ldots, x_{k-1}^{\text{old}}$). Thus, it is possible to use these new values instead of the old ones and speed up the convergence (see Fig. 5.5 for a scheme). Moreover, using this strategy, the newly computed elements of $\boldsymbol{x}_{i+1}$ can directly overwrite the respective elements of $\boldsymbol{x}_i$ and save memory this way.

Following the Theorem 7, the Gauss–Seidel method converges for any starting vector $\boldsymbol{x}_0$ if $\rho((\boldsymbol{L}+\boldsymbol{D})^{-1}\boldsymbol{U}) < 1$. This condition holds, for example, for diagonally dominant matrices as well as for positive definite ones.

5.3.4 Successive Overrelaxation Method

The successive overrelaxation (SOR) method aims to further refine the Gauss–Seidel method. The Gauss–Seidel formula (5.12) can be rewritten as

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \boldsymbol{D}^{-1}[\{\boldsymbol{L}\boldsymbol{x}_{i+1} + (\boldsymbol{D} + \boldsymbol{U})\boldsymbol{x}_i\} - \boldsymbol{b}] = \boldsymbol{x}_i - \boldsymbol{\Delta_i} \ ,$$

which describes the difference $\boldsymbol{\Delta_i}$ between $\boldsymbol{x}_{i+1}$ and $\boldsymbol{x}_i$ expressed for the kth element of $\boldsymbol{x}_{i(+1)}$ from the kth equation, $k = 1, \ldots, n$. The question SOR poses is whether the method can converge faster if we "overly" correct $\boldsymbol{x}_{i+1}$ in each step; that is, if $\boldsymbol{x}_i$ is corrected by a multiple ω of $\boldsymbol{\Delta_i}$ in each iteration. This idea leads to the SOR formula:

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i - \omega\boldsymbol{D}^{-1}[\{\boldsymbol{L}\boldsymbol{x}_{i+1} + (\boldsymbol{D} + \boldsymbol{U})\boldsymbol{x}_i\} - \boldsymbol{b}] \ ,$$

or in the form (5.10),

$$\boldsymbol{x}_{i+1} = (\boldsymbol{D} + \omega\boldsymbol{L})^{-1}\{(1-\omega)\boldsymbol{D} - \omega\boldsymbol{U}\}\boldsymbol{x}_i + \omega(\boldsymbol{D} + \omega\boldsymbol{L})^{-1}\boldsymbol{b} \ . \tag{5.13}$$

The parameter ω is called the (over)relaxation parameter and it can be shown that SOR converges only for $\omega \in (0, 2)$, a result derived by Kahan (1958).

A good choice of parameter ω can speed up convergence, as measured by the spectral radius of the corresponding iteration matrix $\boldsymbol{B}$ (see Theorem 7; a lower spectral radius $\rho(\boldsymbol{B})$ means faster convergence). There is a choice of literature devoted to the optimal setting of relaxation parameter: see Hadjidimos (2000) for a recent overview of the main results concerning SOR. We just present one important result, which is due to Young (1954).

Definition 1. A matrix $\boldsymbol{A}$ is said to be two-cyclic consistently ordered if the eigenvalues of the matrix $\boldsymbol{M}(\alpha) = \alpha\boldsymbol{D}^{-1}\boldsymbol{L} + \alpha^{-1}\boldsymbol{D}^{-1}\boldsymbol{U}, \alpha \neq 0$, are independent of α.

Theorem 8. *Let the matrix $\boldsymbol{A}$ be two-cyclic consistently ordered. Let the respective Gauss–Seidel iteration matrix $\boldsymbol{B} = -(\boldsymbol{L} + \boldsymbol{D})^{-1}\boldsymbol{U}$ have the spectral radius $\rho(\boldsymbol{B}) < 1$. Then the optimal relaxation parameter ω in SOR is given by*

$$\omega_{opt} = \frac{2}{1 + \sqrt{1 - \rho(\boldsymbol{B})}}$$

and for this optimal value it holds $\rho(\boldsymbol{B}; \omega_{opt}) = \omega_{opt} - 1$.

Using SOR with the optimal relaxation parameter significantly increases the rate of convergence. Note however that the convergence acceleration is obtained only for ω very close to ω_{opt}. If ω_{opt} cannot be computed exactly, it is better to take ω slightly larger rather than smaller. Golub and van Loan (1996) describe an approximation algorithm for $\rho(\boldsymbol{B})$.

On the other hand, if the assumptions of Theorem 8 are not satisfied, one can employ the symmetric SOR (SSOR), which performs the SOR iteration twice: once as usual, see (5.13), and once with interchanged $\boldsymbol{L}$ and $\boldsymbol{U}$. SSOR requires more computations per iteration and usually converges slower, but it works for any positive definite matrix and can be combined with various acceleration techniques. See Bjorck (1996) and Hadjidimos (2000) for details.

5.3.5 Gradient Methods

Gradient iterative methods are based on the assumption that $\boldsymbol{A}$ is a symmetric positive definite matrix $\boldsymbol{A}$. They use this assumption to reformulate (5.6) as a minimization problem: $\boldsymbol{x}_e$ is the only minimum of the quadratic form

$$Q(\boldsymbol{x}) = \frac{1}{2}\boldsymbol{x}^\top \boldsymbol{A}\boldsymbol{x} - \boldsymbol{x}^\top \boldsymbol{b} \ .$$

Given this minimization problem, gradient methods construct an iteration series of vectors converging to $\boldsymbol{x}_e$ using the following principle. Having the ith approximation $\boldsymbol{x}_i$, choose a direction $\boldsymbol{v}_i$ and find a number α_i such that the new vector

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i \boldsymbol{v}_i$$

is a minimum of $Q(\boldsymbol{x})$ on the line $\boldsymbol{x}_i + \alpha \boldsymbol{v}_i$, $\alpha \in \mathbb{R}$. Various choices of directions $\boldsymbol{v}_i$ then render different gradient methods, which are in general nonstationary ($\boldsymbol{v}_i$ changes in each iteration). We discuss here three methods: the Gauss–Seidel (as a gradient method), steepest descent and conjugate gradients methods.

Gauss–Seidel Method as a Gradient Method

Interestingly, the Gauss–Seidel method can be seen as a gradient method for the choice

$$\boldsymbol{v}_{kn+i} = \boldsymbol{e}_i \ , \quad k = 0, 1, 2, \dots \ , \quad i = 1, \dots, n \ ,$$

where $\boldsymbol{e}_i$ denotes the ith unit vector. The kth Gauss–Seidel iteration corresponds to n subiterations with $\boldsymbol{v}_{kn+i}$ for $i = 1, \dots, n$.

Steepest Descent Method

The steepest descent method is based on the direction $\boldsymbol{v}_i$ given by the gradient of $Q(\boldsymbol{x})$ at $\boldsymbol{x}_i$. Denoting the residuum of the ith approximation $\boldsymbol{r}_i = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_i$, the iteration formula is

$$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \frac{\boldsymbol{r}_i^\top \boldsymbol{r}_i}{\boldsymbol{r}_i^\top \boldsymbol{A}\boldsymbol{r}_i}\boldsymbol{r}_i \ ,$$

where $\boldsymbol{r}_i$ represents the direction $\boldsymbol{v}_i$ and its coefficient is the $Q(\boldsymbol{x})$-minimizing choice of α_i. By definition, this method reduces $Q(\boldsymbol{x}_i)$ at each step, but it is not very effective. The conjugate gradient method discussed in the next subsection will usually perform better.

Conjugate Gradient Method

In the conjugate gradient (CG) method proposed by Hestenes and Stiefel (1952), the directions $\boldsymbol{v}_i$ are generated by the $\boldsymbol{A}$-orthogonalization of residuum vectors. Given a symmetric positive definite matrix $\boldsymbol{A}$, $\boldsymbol{A}$-orthogonalization is a procedure that constructs a series of linearly independent vectors $\boldsymbol{v}_i$ such that $\boldsymbol{v}_i^\top \boldsymbol{A}\boldsymbol{v}_j = 0$ for $i \neq j$ (conjugacy or $\boldsymbol{A}$-orthogonality condition). It can be used to solve the system (5.6) as follows ($\boldsymbol{r}_i = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_i$ represents residuals).

Algorithm 11

$\boldsymbol{v}_0 = \boldsymbol{r}_0 = \boldsymbol{b} - \boldsymbol{A}\boldsymbol{x}_0$
`do`
$\alpha_i = (\boldsymbol{v}_i^\top \boldsymbol{r}_i)/(\boldsymbol{v}_i^\top \boldsymbol{A}\boldsymbol{v}_i)$
$\boldsymbol{x}_{i+1} = \boldsymbol{x}_i + \alpha_i \boldsymbol{v}_i$
$\boldsymbol{r}_{i+1} = \boldsymbol{r}_i - \alpha_i \boldsymbol{A}\boldsymbol{v}_i$
$\beta_i = -(\boldsymbol{v}_i^\top \boldsymbol{A}\boldsymbol{r}_{i+1})/(\boldsymbol{v}_i^\top \boldsymbol{A}\boldsymbol{v}_i)$
$\boldsymbol{v}_{i+1} = \boldsymbol{r}_{i+1} + \beta_i \boldsymbol{v}_i$
`until a stop criterion holds`

An interesting theoretic property of CG is that it reaches the exact solution in at most n steps because there are not more than n ($\boldsymbol{A}$-)orthogonal vectors. Thus, CG is not a truly iterative method. (This does not have to be the case if $\boldsymbol{A}$ is a singular or non-square matrix, see Kammerer and Nashed 1972.) On the other hand, it is usually used as an iterative method, because it can give a solution within the given accuracy much earlier than after n iterations. Moreover, if the approximate solution $\boldsymbol{x}_n$ after n iterations is not accurate enough (due to computational errors), the algorithm can be restarted with $\boldsymbol{x}_0$ set to $\boldsymbol{x}_n$. Finally, let us note that CG is attractive for use with large sparse matrices because it addresses $\boldsymbol{A}$ only by its multiplication by a vector. This operation can be done very efficiently for a properly stored sparse matrix, see Sect. 5.5.

The principle of CG has many extensions that are applicable also for nonsymmetric nonsingular matrices: for example, generalized minimal residual, Saad and Schultz (1986); (stabilized) biconjugate gradients, Vorst (1992); or quasi-minimal residual, Freund and Nachtigal (1991).

5.4 Eigenvalues and Eigenvectors

In this section, we deal with methods for computing eigenvalues and eigenvectors of a matrix $\boldsymbol{A} \in \mathbb{R}^{n\times n}$. First, we discuss a simple power method for computing one or few eigenvalues (Sect. 5.4.1). Next, we concentrate on methods performing the complete eigenanalysis, that is, finding all eigenvalues (the Jacobi, QR, and LR methods in Sects. 5.4.2–5.4.5). Finally, we briefly describe a way to improve already

computed eigenvalues and to find the corresponding eigenvector. Additionally, note that eigenanalysis can be also done by means of SVD, see Sect. 5.1.4. For more details on the described as well as some other methods, one can consult monographs by Gentle (1998), Golub and van Loan (1996), Press et al. (1992) and Stoer and Bulirsch (2002).

Before discussing specific methods, let us describe the principle common to most of them. We assume that $\boldsymbol{A} \in \mathbb{R}^{n \times n}$ has eigenvalues $|\lambda_1| \geq |\lambda_2| \geq \ldots \geq |\lambda_n|$. To find all eigenvalues, we transform the original matrix $\boldsymbol{A}$ to a simpler matrix $\boldsymbol{B}$ such that it is similar to $\boldsymbol{A}$ (recall that matrices $\boldsymbol{A}$ and $\boldsymbol{B}$ are similar if there is a matrix $\boldsymbol{T}$ such that $\boldsymbol{B} = \boldsymbol{T}^{-1}\boldsymbol{A}\boldsymbol{T}$). The similarity of $\boldsymbol{A}$ and $\boldsymbol{B}$ is crucial since it guarantees that both matrices have the same eigenvalues and their eigenvectors follow simple relation: if $\boldsymbol{g}$ is an eigenvector of $\boldsymbol{B}$ corresponding to its eigenvalue λ, then $\boldsymbol{T}\boldsymbol{g}$ is an eigenvector of $\boldsymbol{A}$ corresponding to the same eigenvalue λ.

There are two basic strategies to construct a similarity transformation $\boldsymbol{B}$ of the original matrix $\boldsymbol{A}$. First, one can use a series of simple transformations, such as GRs, and eliminate elements of $\boldsymbol{A}$ one by one (see the Jacobi method, Sect. 5.4.2). This approach is often used to transform $\boldsymbol{A}$ to its tridiagonal or upper Hessenberg forms. (Matrix $\boldsymbol{B}$ has the upper Hessenberg form if it is an upper triangular except for the first subdiagonal; that is, $A_{ij} = 0$ for $i > j+1$, where $i, j = 1, \ldots, n$). Second, one can also factorize $\boldsymbol{A}$ into $\boldsymbol{A} = \boldsymbol{F}_L\boldsymbol{F}_R$ and switch the order of factors, $\boldsymbol{B} = \boldsymbol{F}_R\boldsymbol{F}_L$ (similarity of $\boldsymbol{A}$ and $\boldsymbol{B}$ follows from $\boldsymbol{B} = \boldsymbol{F}_R\boldsymbol{F}_L = \boldsymbol{F}_L^{-1}\boldsymbol{A}\boldsymbol{F}_L$). This is used for example by the LR method (Sect. 5.4.5). Finally, there are methods combining both approaches.

5.4.1 Power Method

In its basic form, the power method aims at finding only the largest eigenvalue λ_1 of a matrix $\boldsymbol{A}$ and the corresponding eigenvector. Let us assume that the matrix $\boldsymbol{A}$ has a dominant eigenvalue ($|\lambda_1| > |\lambda_2|$) and n linearly independent eigenvectors.

The power method constructs two series c_i and $\boldsymbol{x}_i$, $i \in \mathbb{N}$, that converge to λ_1 and to the corresponding eigenvector $\boldsymbol{g}_1$, respectively. Starting from a vector $\boldsymbol{x}_0$ that is not orthogonal to $\boldsymbol{g}_1$, one only has to iteratively compute $\boldsymbol{A}\boldsymbol{x}_i$ and split it to its norm c_{i+1} and the normalized vector $\boldsymbol{x}_{i+1}$, see Algorithm 5.4.1. Usually, the Euclidian ($c_{i+1} = \|\boldsymbol{A}\boldsymbol{x}_i\|_2$) and maximum ($c_{i+1} = \max_{j=1,\ldots,n} |(\boldsymbol{A}\boldsymbol{x}_i)_j|$) norms are used.

Algorithm 12

$i = 0$
do
  $i = i + 1$
  $\boldsymbol{x}_{i+1} = \boldsymbol{A}\boldsymbol{x}_i$
  $c_{i+1} = \|\boldsymbol{A}\boldsymbol{x}_{i+1}\|$
  $\boldsymbol{x}_{i+1} = \boldsymbol{x}_{i+1}/c_{i+1}$
until a stop criterion holds

Although assessing the validity of assumptions is far from trivial, one can usually easily recognize whether the method converges from the behaviour of the two constructed series.

Furthermore, the power method can be extended to search also for other eigenvalues; for example, the smallest one and the second largest one. First, if $\boldsymbol{A}$ is nonsingular, we can apply the power method to $\boldsymbol{A}^{-1}$ to find the smallest eigenvalue λ_n because $1/\lambda_n$ is the largest eigenvalue of $\boldsymbol{A}^{-1}$. Second, if we need more eigenvalues and λ_1 is already known, we can use a reduction method to construct a matrix $\boldsymbol{B}$ that has the same eigenvalues and eigenvectors as $\boldsymbol{A}$ except for λ_1, which is replaced by zero eigenvalue. To do so, we need to find a (normalized) eigenvector $\boldsymbol{h}_1$ of $\boldsymbol{A}^\top$ corresponding to λ_1 ($\boldsymbol{A}$ and $\boldsymbol{A}^\top$ have the same eigenvalues) and to set $\boldsymbol{B} = \boldsymbol{A} - \lambda_1 \boldsymbol{h}_1 \boldsymbol{h}_1^\top$. Naturally, this process can be repeated to find the third and further eigenvalues.

Finally, let us mention that the power method can be used also for some matrices without dominant eigenvalue (e.g., matrices with $\lambda_1 = \ldots = \lambda_p$ for some $1 < p \leq n$). For further extensions of the power method see Sidi (1989), for instance.

5.4.2 Jacobi Method

For a symmetric matrix $\boldsymbol{A}$, the Jacobi method constructs a series of orthogonal matrices $\boldsymbol{R_i}$, $i \in \mathbb{N}$, such that the matrix $\boldsymbol{T_i} = \boldsymbol{R_i}^\top \ldots \boldsymbol{R_1}^\top \boldsymbol{A} \boldsymbol{R_1} \ldots \boldsymbol{R_i}$ converges to a diagonal matrix $\boldsymbol{D}$. Each matrix $\boldsymbol{R_i}$ is a GR matrix defined in (5.5), whereby the angle α is chosen so that one nonzero element $(\boldsymbol{T_i})_{jk}$ becomes zero in $\boldsymbol{T_{i+1}}$. Formulas for computing $\boldsymbol{R_i}$ given the element (j, k) to be zeroed are described in Gentle (1998), for instance. Once the matrix $\boldsymbol{A}$ is diagonalized this way, the diagonal of $\boldsymbol{D}$ contains the eigenvalues of $\boldsymbol{A}$ and the columns of matrix $\boldsymbol{R} = \boldsymbol{R_1} \cdot \ldots \cdot \boldsymbol{R_i}$ represent the associated eigenvectors.

There are various strategies to choose an element (j, k) which will be zeroed in the next step. The classical Jacobi method chooses the largest off-diagonal element in absolute value and it is known to converge. (Since searching the maximal element is time consuming, various systematic schemes were developed, but their convergence cannot be often guaranteed.) Because the Jacobi method is relatively slow, other methods are usually preferred (e.g., the QR method). On the other hand, it has recently become interesting again because of its accuracy and easy parallelization (Higham 1997; Zhou and Brent 2003).

5.4.3 Givens and Householder Reductions

The Givens and Householder methods use a similar principle as the Jacobi method. A series of GRs or HRs, designed such that they form similarity transformations, is applied to a symmetric matrix $\boldsymbol{A}$ in order to transformed it to a tridiagonal

matrix. (A tridiagonal matrix is the Hessenberg form for symmetric matrices.) This tridiagonal matrix is then subject to one of the iterative methods, such as the QR or LR methods discussed in the following paragraphs. Formulas for Givens and Householder similarity transformations are given in Press et al. (1992), for instance.

5.4.4 QR Method

The QR method is one of the most frequently used methods for the complete eigenanalysis of a nonsymmetric matrix, despite the fact that its convergence is not ensured. A typical algorithm proceeds as follows. In the first step, the matrix $\boldsymbol{A}$ is transformed into a Hessenberg matrix using Givens or Householder similarity transformations (see Sects. 5.1.3 and 5.4.3). In the second step, this Hessenberg matrix is subject to the iterative process called chasing. In each iteration, similarity transformations, such as GRs, are first used to create nonzero entries in positions $(i+2, i)$, $(i+3, i)$ and $(i+3, i+1)$ for $i = 1$. Next, similarity transformations are repeatedly used to zero elements $(i+2, i)$ and $(i+3, i)$ and to move these "nonzeros" towards the lower right corner of the matrix (i.e., to elements $(i+2, i)$, $(i+3, i)$ and $(i+3, i+1)$ for $i = i+1$). As a result of chasing, one or two eigenvalues can be extracted. If $A_{n,n-1}$ becomes zero (or negligible) after chasing, element $A_{n,n}$ is an eigenvalue. Consequently, we can delete the nth row and column of the matrix and apply chasing to this smaller matrix to find another eigenvalue. Similarly, if $A_{n-1,n-2}$ becomes zero (or negligible), the two eigenvalues of the 2×2 submatrix in the lower right corner are eigenvalues of $\boldsymbol{A}$. Subsequently, we can delete last two rows and columns and continue with the next iteration.

Since a more detailed description of the whole iterative process goes beyond the extent of this contribution, we refer a reader to Gentle (1998) for a shorter discussion and to Golub and van Loan (1996) and Press et al. (1992) for a more detailed discussion of the QR method.

5.4.5 LR Method

The LR method is based on a simple observation that decomposing a matrix $\boldsymbol{A}$ into $\boldsymbol{A} = \boldsymbol{F}_L \boldsymbol{F}_R$ and multiplying the factors in the inverse order results in a matrix $\boldsymbol{B} = \boldsymbol{F}_R \boldsymbol{F}_L$ similar to $\boldsymbol{A}$. Using the LU decomposing (Sect. 5.1.2), the LR method constructs a matrix series $\boldsymbol{A}_i$ for $i \in \mathbb{N}$, where $\boldsymbol{A}_1 = \boldsymbol{A}$ and

$$\boldsymbol{A}_i = \boldsymbol{L}_i \boldsymbol{U}_i \quad \Longrightarrow \quad \boldsymbol{A}_{i+1} = \boldsymbol{U}_i \boldsymbol{L}_i \ ,$$

where $\boldsymbol{L}_i$ is a lower triangular matrix and $\boldsymbol{U}_i$ is an upper triangular matrix with ones on its diagonal. For a wide class of matrices, including symmetric positive

definite matrices, $\boldsymbol{A_i}$ and $\boldsymbol{L_i}$ are proved to converge to the same lower triangular matrix $\boldsymbol{L}$, whereby the eigenvalues of $\boldsymbol{A}$ form the diagonal of $\boldsymbol{L}$ and are ordered by the decreasing absolute value.

5.4.6 *Inverse Iterations*

The method of inverse iterations can be used to improve an approximation λ^* of an eigenvalue λ of a matrix $\boldsymbol{A}$. The method is based on the fact that the eigenvector $\boldsymbol{g}$ associated with λ is also an eigenvector of $\tilde{\boldsymbol{A}} = (\boldsymbol{A} - \lambda^* \boldsymbol{I})^{-1}$ associated with the eigenvalue $\tilde{\lambda} = (\lambda - \lambda^*)^{-1}$. For an initial approximation λ^* close to λ, $\tilde{\lambda}$ is the dominant eigenvalue of $\tilde{\boldsymbol{A}}$. Thus, it can be computed by the power method described in Sect. 5.4.1, whereby λ^* could be modified in each iteration in order to improve the approximation of λ.

This method is not very efficient without a good starting approximation, and therefore, it is not suitable for the complete eigenanalysis. On the other hand, the use of the power method makes it suitable for searching of the eigenvector $\boldsymbol{g}$ associated with λ. Thus, the method of inverse iterations often complements methods for complete eigenanalysis and serves then as a tool for eigenvector analysis. For this purpose, one does not have to perform the iterative improvement of initial λ^*: applying the power method on $\tilde{\boldsymbol{A}} = (\boldsymbol{A} - \lambda^* \boldsymbol{I})^{-1}$ suffices. See Ipsen (1997), Press et al. (1992) and Stoer and Bulirsch (2002) for more details.

5.5 Sparse Matrices

Numerical problems arising in some applications, such as seemingly unrelated regressions, spatial statistics, or support vector machines (Chap. III.15), are sparse: they often involve large matrices, which have only a small number of nonzero elements. (It is difficult to specify what exactly "small number" is.) From the practical point of view, a matrix is sparse if it has so many zero elements that it is worth to inspect their structure and use appropriate methods to save storage and the number of operations. Some sparse matrices show a regular pattern of nonzero elements (e.g., band matrices), while some exhibit a rather irregular pattern. In both cases, solving the respective problem efficiently means to store and operate on only nonzero elements and to keep the "fill," the number of newly generated nonzero elements, as small as possible.

In this section, we first discuss some of storage schemes for sparse matrices, which could indicate what types of problems can be effectively treated as sparse ones (Sect. 5.5.1). Later, we give examples of classical algorithms adopted for sparse matrices (Sect. 5.5.2). Monographs introducing a range of methods for sparse matrices include Duff et al. (1989), Hackbusch (1994) and Saad (2003).

5.5.1 Storage Schemes for Sparse Matrices

To save storage, only nonzero elements of a sparse vector or matrix should be stored. There are various storage schemes, which require approximately from two to five times the number of nonzero elements to store a vector or a matrix. Unfortunately, there is no standard scheme. We discuss here the widely used and sufficiently general compressed (row) storage for vectors and for general and banded matrices.

The compressed form of a vector $\boldsymbol{x}$ consists of a triplet $(\boldsymbol{c}, \boldsymbol{i}, n_0)$, where $\boldsymbol{c}$ is a vector containing nonzero elements of $\boldsymbol{x}$, $\boldsymbol{i}$ is an integer vector containing the indices of elements stored in $\boldsymbol{c}$ and n_0 specifies the number of nonzero elements. The stored elements are related to the original vector by formula $x_{\{i_j\}} = c_j$ for $j = 1, \ldots, n_0$. To give an example, the vector $\boldsymbol{x} = (0, 0, 3, 0, -8, 1.5, 0, 0, 0, 16, 0)$ could be stored as

$$\boldsymbol{c} = (3, 1.5, -8, 16)\ , \quad \boldsymbol{i} = (3, 6, 5, 10)\ , \quad n_0 = 4\ .$$

Obviously, there is no need to store the elements in the original order. Therefore, adding new nonzero elements is easy. Operations involving more sparse vectors are simpler if we can directly access elements of one vector, that is, if one of the vectors is "uncompressed." For example, computing the inner product $a = \boldsymbol{x}^\top \boldsymbol{y}$ of a sparse vector $\boldsymbol{x}$ stored in the compressed form with a sparse uncompressed vector $\boldsymbol{y}$ follows the algorithm

$$a = 0\ ; \quad \text{for} \quad j = 1\ , \ldots, n_0\ : \quad a = a + y_{\{i_j\}} \cdot c_j\ .$$

The compressed row storage for matrices is a generalization of the vector concept. We store the nonzero elements of $\boldsymbol{A}$ as a set of sparse row (or column) vectors in the compressed form. The main difference is that, instead of a single number n_0, we need to store a whole vector $\boldsymbol{n_0}$ specifying the positions of the first row elements of $\boldsymbol{A}$ in $\boldsymbol{c}$. For example, the matrix

$$\boldsymbol{A} = \begin{pmatrix} A_{11} & A_{12} & 0 & 0 & 0 & 0 \\ A_{21} & 0 & 0 & A_{24} & 0 & 0 \\ 0 & 0 & 0 & A_{34} & 0 & 0 \\ 0 & 0 & A_{43} & 0 & A_{45} & 0 \\ 0 & A_{52} & 0 & 0 & 0 & A_{56} \end{pmatrix}$$

would be represented rowwise as

$$\begin{aligned} \boldsymbol{c} &= (A_{11}, A_{12} | A_{21}, A_{24} | A_{34} | A_{43}, A_{45} | A_{52}, A_{56})\ , \\ \boldsymbol{i} &= (1, 2 | 1, 4 | 4 | 3, 5 | 2, 6)\ , \\ \boldsymbol{n_0} &= (1, 3, 5, 6, 8, 10)\ . \end{aligned}$$

(The sign "|" just emphasizes the end of a row and has no consequence for the storage itself.) As in the case of vectors, the elements in each row do not have to be ordered. Consequently, there is no direct access to a particular element A_{ij} stored in $\boldsymbol{c}$. Nevertheless, retrieving a row is easy: it suffices to examine the part of $\boldsymbol{i}$ corresponding to the ith row, which is given by $\boldsymbol{n_0}$. On the contrary, retrieving a column involves a search through the whole storage scheme. Therefore, if a fast access to columns is necessary, it is preferable to simultaneously store $\boldsymbol{A}$ rowwise and columnwise.

A special type of sparse matrices are matrices with a banded structure.

Definition 2. The row bandwidth of a matrix $\boldsymbol{A} \in \mathbb{R}^{m\times n}$ is defined as

$$w(\boldsymbol{A}) = \max_{1\le i\le m} \left(l_i(\boldsymbol{A}) - f_i(\boldsymbol{A}) + 1\right) ,$$

where $f_i(\boldsymbol{A}) = \min\{j \,|\, \boldsymbol{A}_{ij} \neq 0\}$ and $l_i(\boldsymbol{A}) = \max\{j \,|\, \boldsymbol{A}_{ij} \neq 0\}$ are column indices of the first and last nonzero elements in the ith row of $\boldsymbol{A}$.

A banded matrix $\boldsymbol{A}$ is considered to be sparse if $w(\boldsymbol{A}) \ll n$. Contrary to the general case, vector $\boldsymbol{c}$ of a banded matrix typically contains for each row all elements between the first and last nonzero ones. Thus, the storage scheme does not have to include in $\boldsymbol{i}$ all column indices, only one index for the first nonzero element in a row. On the other hand, zeros within the band have to be stored as well. For example, the matrix

$$\boldsymbol{A} = \begin{pmatrix} A_{11} & A_{12} & 0 & 0 & 0 \\ 0 & A_{22} & 0 & A_{24} & 0 \\ 0 & 0 & 0 & A_{34} & 0 \\ 0 & 0 & A_{43} & 0 & A_{45} \end{pmatrix}$$

would be represented as

$$\begin{aligned} \boldsymbol{c} &= (A_{11}, A_{12}|A_{22}, 0, A_{24}|A_{34}|A_{43}, 0, A_{45}) , \\ \boldsymbol{i} &= (1, 2, 4, 3) , \\ \boldsymbol{n}_0 &= (1, 3, 6, 7, 10) . \end{aligned}$$

An interesting observation is that the row bandwidth $w(\boldsymbol{A})$ can be influenced by column permutations. The fill-minimizing column orderings are discussed by Bjorck (1996) and George and Ng (1983), for instance.

Details on some other storage schemes can be found in Duff et al. (1989) and Press et al. (1992).

5.5.2 Methods for Sparse Matrices

Methods for sparse matrices are still subject to intensive research. Moreover, the choice of a suitable method for a given problem (and even the choice of an

algorithm for elementary operations such as matrix-vector multiplication) depends on many factors, including dimension, matrix type storage scheme, and computational environment (e.g., storage in virtual memory vs. auxiliary storage; vector vs. parallel computing, etc.). Therefore, we provide only a general overview and references to most general results. More details can be found in Bjorck (1996), Dongarra and Eijkhout (2000), Duff et al. (1989), Hackbusch (1994) and Saad (2003).

First, many discussed algorithms can be relatively easily adopted for banded matrices. For example, having a row-based storage scheme, one just needs to modify the summation limits in the row version of Cholesky decomposition. Moreover, the positions of nonzero elements can be determined in advance (Ng and Peyton 1993).

Second, the algorithms for general sparse matrices are more complicated. A graph representation may be used to capture the nonzero pattern of a matrix as well as to predict the pattern of the result (e.g., the nonzero pattern of $\boldsymbol{A}^\top \boldsymbol{A}$, the Cholesky factor $\boldsymbol{U}$, etc.). To give an overview, methods adopted for sparse matrices include, but are not limited to, usually used decompositions (e.g., Cholesky, Ng and Peyton 1993; LU and LDU, Mittal and Al-Kurdi 2002; QR, George and Liu 1987, and Heath 1984), solving systems of equations by direct (Gupta 2002; Tran et al. 1996) and iterative methods (Makinson and Shah 1986; Zlatev and Nielsen 1988) and searching eigenvalues (Bergamaschi and Putti 2002; Golub et al. 2000).

References

Alanelli, M., Hadjidimos, A.: Block Gauss elimination followed by a classical iterative method for the solution of linear systems. J. Comput. Appl. Math. **163**(2), 381–400 (2004)

Anderson, E., Bai, Z., Bischof, C., Blackford, S., Demmel, J., Dongarra, J., Croz, J.D., Greenbaum, A., Hammarling, S., McKenney, A., Sorensen, D.: LAPACK Users' Guide, (3rd edn.), SIAM Press, Philadelphia, USA (1999)

Axelsson, O.: Iterative Solution Methods. Cambridge University Press, Cambridge, UK (1994)

Bergamaschi, L., Putti, M.: Numerical comparison of iterative eigensolvers for large sparse symmetric positive definite matrices. Comput. Meth. Appl. Mech. Eng. **191**, 5233–5247 (2002)

Benoit, C.: Note sur une mèthode de rèsolution des èquations normales provenant de l'application de la mèthode des moindres carrès á un systï£¡me d'èquations linèaires en nombre infèrieur á celui des inconnues. Application de la mèthode á la rèsolution d'un systï£¡me dèfini d'èquations linèaires (Procèdè du Commandant Cholesky). Bull. gèodèsique **2**, 5–77 (1924)

Björck, A.: Numerics of Gram–Schmidt Orthogonalization. Lin. Algebra Appl. **198**, 297–316 (1994)

Björck, A.: Numerical Methods for Least Squares Problems. SIAM Press, Philadelphia, USA (1996)

Croz, J.D., Higham, N.J.: Stability of methods for matrix inversion. IMA J. Numer. Anal. **12**, 1–19 (1992)

Dax, A.: A modified Gram–Schmidt algorithm with iterative orthogonalization and column pivoting. Lin. Algebra Appl. **310**, 25–42 (2000)

Demmel, J.W., Gu, M., Eisenstat, S., Slapničar, I., Veselić, K., Drmač, Z.: Computing the singular value decomposition with high relative accuracy. Lin. Algebra Appl. **299**, 21–80 (1999)

Dongarra, J.J., Eijkhout, V.: Numerical linear algebra algorithms and software. J. Comput. Appl. Math. **123**, 489–514 (2000)
Duff, I.S., Erisman, A.M., Reid, J.K.: Direct Methods for Sparse Matrices. Oxford University Press, USA (1989)
Freund, R., Nachtigal, N.: QMR: a quasi-minimal residual method for non-Hermitian linear systems. Numer. Math. **60**, 315–339 (1991)
Gallivan, K.A., Plemmons, R.J., Sameh, A.H.: Parallel algorithms for dense linear algebra computations. SIAM Rev. **32**, 54–135 (1990)
Gentle, J.E.: Numerical Linear Algebra for Applications in Statistics. Springer, New York, USA (1998)
Gentleman, W.M.: Least squares computations by Givens transformations without square roots. J. Inst. Math. Appl. **12**, 329–336 (1973)
Gentleman, W.M.: Error analysis of QR decomposition by Givens transformations. Lin. Algebra Appl. **10**, 189–197 (1975)
George, A., Liu, J.W.H.: Householder reflections versus givens rotations in sparse orthogonal decomposition. Lin. Algebra Appl. **88**, 223–238 (1987)
George, J.A., Ng, E.G.: On row and column orderings for sparse least squares problems. SIAM J. Numer. Anal. **20**, 326–344 (1983)
Givens, W.: Computation of Plane Unitary Rotations Transforming a General Matrix to Triangular Form. J. SIAM **6**(1), 26–50 (1958)
Golub, G.H.: Numerical methods for solving least squares problems. Numer. Math. **7**, 206–216 (1965)
Golub, G.H., Kahan, W.: Calculating the singular values and pseudo-inverse of a matrix. SIAM J. Numer. Anal. B **2**, 205–224 (1965)
Golub, G.H., Reinsch, C.: Singular value decomposition and least squares solution. Numer. Math. **14**, 403–420 (1970)
Golub, G.H., van Loan, C.F.: Matrix Computations. Johns Hopkins University Press, Baltimore, Maryland (1996)
Golub, G.H., Zhang, Z., Zha, H.: Large sparse symmetric eigenvalue problems with homogeneous linear constraints: the Lanczos process with inner-outer iterations. Lin. Algebra Appl. **309**, 289–306 (2000)
Gupta, A.: Recent Advances in Direct Methods for Solving Unsymmetric Sparse Systems of Linear Equations. ACM Trans. Math. Software **28**, 301–324 (2002)
Hackbusch, W.: Iterative Solution of Large Sparse Systems of Equations. Springer, New York, USA (1994)
Hadjidimos, A.: Successive Overrelaxation (SOR) and related methods. J. Comput. Appl. Math. **123**, 177–199 (2000)
Hammarling, S.: A note on modifications to the Givens plane rotation. J. Inst. Math. Appl. **13**, 215–218 (1974)
Hari, V., Veselić, K.: On Jacobi methods for singular value decompositions. SIAM J. Sci. Stat. Comput. **8**, 741–754 (1987)
Harville, D.A.: Matrix Algebra from a Statistician's Perspective. Springer, New York, USA (1997)
Heath, M.T.: Numerical methods for large sparse linear least squares problems. SIAM J. Sci. Stat. Comput. **26**, 497–513 (1984)
Hestenes, M.R., Stiefel, E.: Method of conjugate gradients for solving linear systems. J. Res. Nat. Bur. Standards B **49**, 409–436 (1952)
Higham, N.J.: The accuracy of solutions to triangular systems. SIAM J. Numer. Anal. **26**, 1252–1265 (1989)
Higham, N.J.: Recent Developments in Dense Numerical Linear Algebra. In: Duff, I.S., Watson, G.A. (eds.) State of the Art in Numerical Analysis. Oxford University Press, Oxford (1997)
Higham, N.J.: QR factorization with complete pivoting and accurate computation of the SVD. Lin. Algebra Appl. **309**, 153–174 (2000)
Higham, N.J.: Accuracy and Stability of Numerical Algorithms. (2nd edn.), SIAM Press, Philadelphia, USA (2002)

Hong, Y.P., Tan, C.T.: Rank-revealing QR fatorizations and the singular value decomposition. Math. Comput. **58**, 213–232 (1992)
Householder, A.S.: Unitary triangularization of a nonsymmetric matrix. J. Assoc. Comput. Machinery **5**, 339–342 (1958)
Ipsen, I.C.F.: Computing an Eigenvector with Inverse Iteration. SIAM Rev. **39**, 254–291 (1997)
Kahan, W.: Gauss–Seidel Methods of Solving Large Systems of Linear Equations. Doctoral thesis, University of Toronto, Toronto, Canada (1958)
Kammerer, W.J., Nashed, M.Z.: On the convergence of the conjugate gradient method for singular linear operator equations. SIAM J. Numer. Anal. **9**, 165–181 (1972)
Makinson, G.J., Shah, A.A.: An iterative solution method for solving sparse nonsymmetric linear systems. J. Comput. Appl. Math. **15**, 339–352 (1986)
Martin, R.S., Peters, G., Wilkinson, J.H.: Symmetric decomposition of a positive definite matrix. In: Wilkinson, J.H., Reinsch, C. (eds.) Linear Algebra (Handbook for Automation Computation. vol. 2, Springer, Heidelberg, Germany (1965)
Meinguet, J.: Refined error analysis of cholesky factorization. SIAM J. Numer. Anal. **20**, 1243–1250 (1983)
Milaszewicz, J.P.: Improving Jacobi and Gauss–Seidel Iterations. Lin. Algebra Appl. **93**, 161–170 (1987)
Miranian, L., Gu, M.: Strong rank revealing LU factorizations. Lin. Algebra Appl. **367**, 1–16 (2003)
Mittal, R.C., Al-Kurdi, A.: LU-decomposition and numerical structure for solving large sparse nonsymmetric linear systems. Comput. Math. Appl. **43**, 131–155 (2002)
Ng, E.G., Peyton, B.W.: Block Sparse Cholesky Algorithm on Advanced Uniprocessor Computers. SIAM J. Sci. Comput. **14**, 1034–1056 (1993)
Nool, M.: Explicit parallel block Cholesky algorithms on the CRAY APP. Appl. Numer. Math. **19**, 91–114 (1995)
Pan, C.T.: On the existence and computation of rank revealing LU factorizations. Lin. Algebra Appl. **316**, 199–222 (2000)
Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C: the Art of Scientific Computing. (2nd edn). Cambridge University Press, Cambridge, UK (1992)
Rice, J.R.: Experiments on Gram–Schmidt orthogonalization. Math. Comput. **20**, 325–328 (1966)
Saad, Y.: Iterative Methods for Sparse Linear Systems. (2nd edn.). SIAM Press, USA (2003)
Saad, Y., Schultz, M.H.: GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **7**, 856–869 (1986)
Sidi, A.: On extensions of the power method for normal operators. Lin. Algebra Appl. **120**, 207–224 (1989)
Skeel, R.D.: Iterative refinement implies numerical stability for Gaussian elimination. Math. Comput. **35**, 817–832 (1980)
Stewart, G.W.: The economical storage of plane rotations. Numer. Math. **25**, 137–138 (1976)
Stewart, G.W.: Matrix Algorithms, Volume I: Basic Decompositions. SIAM Press, Philadelphia, USA (1998)
Stoer, J., Bulirsch, R.: Introduction to Numerical Analysis. (3rd edn.). Springer, New York, USA (2002)
Tran, T.M., Gruber, R., Appert, K., Wuthrich, S.: A direct parallel sparse matrix solver. Comput. Phys. Comm. **96**, 118–128 (1996)
Trefethen, L.N., Bau, D.: Numerical Linear Algebra. SIAM Press, Philadelphia, USA (1997)
von Matt, U.: The Orthogonal QD-Algorithm. In: Moonen, M., De Moor, B. (eds.) SVD and Signal Processing, III: Algorithms, Architectures and Applications. Elsevier, Amsterdam (1995)
Vorst, V.D.: Bi-CGSTAB: A fast and smoothly converging variant of Bi-CG for the solution of nonsymmetric linear systems. SIAM J. Sci. Stat. Comput. **13**, 631–644 (1992)
Wampler, R.H.: A report on the accuracy of some widely used least squares computer programs. J. Am. Stat. Assoc. **65**, 549–565 (1970)

Young, D.M.: Iterative methods for solving partial differential equations of elliptic type. Trans. Am. Math. Soc. **76**, 92–111 (1954)
Zhou, B.B., Brent, R.P.: An efficient method for computing eigenvalues of a real normal matrix. J. Parallel Distr. Comput. **63**, 638–648 (2003)
Zlatev, Z., Nielsen, H.B.: Solving large and sparse linear least-squares problems by conjugate gradient algorithms. Comput. Math. Appl. **15**, 185–202 (1988)
Zou, Q.: An observation on Gauss elimination. Comput. Math. Appl. **22**, 69–70 (1991)

Chapter 6
The EM Algorithm

Shu Kay Ng, Thriyambakam Krishnan, and Geoffrey J. McLachlan

6.1 Introduction

The Expectation-Maximization (EM) algorithm is a broadly applicable approach to the iterative computation of maximum likelihood (ML) estimates, useful in a variety of incomplete-data problems. It is based on the idea of solving a succession of simpler problems that are obtained by augmenting the original observed variables (the incomplete data) with a set of additional variables that are unobservable or unavailable to the user. These additional data are referred to as the missing data in the EM framework. The EM algorithm is closely related to the *ad hoc* approach to estimation with missing data, where the parameters are estimated after filling in initial values for the missing data. The latter are then updated by their predicted values using these initial parameter estimates. The parameters are then re-estimated, and so on, proceeding iteratively until convergence. On each iteration of the EM algorithm, there are two steps called the Expectation step (or the E-step) and the Maximization step (or the M-step). The name "EM algorithm" was given by Dempster et al. (1977) in their fundamental paper.

The EM algorithm has a number of desirable properties, such as its numerical stability, reliable global convergence, and simplicity of implementation. However, the EM algorithm is not without its limitations. In its basic form, the EM algorithm

S.K. Ng (✉)
School of Medicine, Griffith University, Meadowbrook, QLD 4131, Australia
e-mail: s.ng@griffith.edu.au

T. Krishnan
Mu-Sigma Business Solutions Pvt. Ltd, Kalyani Platina, K.R. Puram Hobli, Bangalore, India
e-mail: krishnant001@gmail.com

G.J. McLachlan
Department of Mathematics, University of Queensland, Brisbane, QLD, Australia
e-mail: g.mclachlan@uq.edu.au

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_6,

lacks of an in-built procedure to compute the covariance matrix of the parameter estimates and it is sometimes very slow to converge. Moreover, certain complex incomplete-data problems lead to intractable E-steps and M-steps. The first edition of the book chapter published in 2004 covered the basic theoretical framework of the EM algorithm and discussed further extensions of the EM algorithm to handle complex problems. The second edition attempts to capture advanced developments in EM methodology in recent years. In particular, there are many connections between the EM algorithm and Markov chain Monte Carlo algorithms. Furthermore, the key idea of the EM algorithm where a function of the log likelihood is maximized in a iterative procedure occurs in other optimization procedures as well, leading to a more general way of treating EM algorithm as an optimization procedure. Capturing the above developments in the second edition has led to the addition of new examples in the applications of the EM algorithm or its variants to complex problems, especially in the related fields of biomedical and health sciences.

The remaining of Sect. 6.1 focusses on a brief description of ML estimation and the incomplete-data structure of the EM algorithm. The basic theoretical framework of the EM algorithm is presented in Sect. 6.2. In particular, the monotonicity of the algorithm, convergence, and rate of convergence properties are systematically examined. In Sect. 6.3, the EM methodology presented in this chapter is illustrated in some commonly occurring situations such as the fitting of normal mixtures and missing observations in terms of censored failure times. Another example is provided in which the EM algorithm is used to train a mixture-of-experts model. Consideration is given also to clarify some misconceptions about the implementation of the E-step, and the important issue associated with the use of the EM algorithm, namely the provision of standard errors. We discuss further modifications and extensions to the EM algorithm in Sect. 6.4. In particular, the extensions of the EM algorithm known as the Monte Carlo EM, ECM, ECME, AECM, and PX–EM algorithms are considered. With the considerable attention being given to the analysis of large data sets, as in typical data mining applications, recent work on speeding up the implementation of the EM algorithm is discussed. These include the IEM, SPIEM, and the use of multiresolution kd-trees. In Sect. 6.5, the relationship of the EM algorithm to other data augmentation techniques, such as the Gibbs sampler and MCMC methods is presented briefly. The Bayesian perspective is also included by showing how the EM algorithm and its variants can be adapted to compute the maximum *a posteriori* (MAP) estimate. We conclude the chapter with a brief account of the applications of the EM algorithm in such topical and interesting areas as bioinformatics and health sciences.

6.1.1 Maximum Likelihood Estimation

Maximum likelihood estimation and likelihood-based inference are of central importance in statistical theory and data analysis. Maximum likelihood estimation is a general-purpose method with attractive properties. It is the most-often used

estimation technique in the frequentist framework, and it can be equally applied to find the mode of the posterior distribution in a Bayesian framework (Chap. III.26). Often Bayesian solutions are justified with the help of likelihoods and maximum likelihood estimates (MLE), and Bayesian solutions are similar to penalized likelihood estimates. Maximum likelihood estimation is an ubiquitous technique and is used extensively in every area where statistical techniques are used.

We assume that the observed data $\boldsymbol{y}$ has probability density function (p.d.f.) $g(\boldsymbol{y}; \boldsymbol{\Psi})$, where $\boldsymbol{\Psi}$ is the vector containing the unknown parameters in the postulated form for the p.d.f. of $\boldsymbol{Y}$. Our objective is to maximize the likelihood $L(\boldsymbol{\Psi}) = g(\boldsymbol{y}; \boldsymbol{\Psi})$ as a function of $\boldsymbol{\Psi}$, over the parameter space $\boldsymbol{\Omega}$. That is,

$$\partial L(\boldsymbol{\Psi})/\partial \boldsymbol{\Psi} = \mathbf{0},$$

or equivalently, on the log likelihood,

$$\partial \log L(\boldsymbol{\Psi})/\partial \boldsymbol{\Psi} = \mathbf{0}. \tag{6.1}$$

The aim of ML estimation is to determine an estimate $\hat{\boldsymbol{\Psi}}$, so that it defines a sequence of roots of (6.1) that is consistent and asymptotically efficient. Such a sequence is known to exist under suitable regularity conditions (Cramér 1946). With probability tending to one, these roots correspond to local maxima in the interior of $\boldsymbol{\Omega}$. For estimation models in general, the likelihood usually has a global maximum in the interior of $\boldsymbol{\Omega}$. Then typically a sequence of roots of (6.1) with the desired asymptotic properties is provided by taking $\hat{\boldsymbol{\Psi}}$ to be the root that globally maximizes $L(\boldsymbol{\Psi})$; in this case, $\hat{\boldsymbol{\Psi}}$ is the MLE. We shall henceforth refer to $\hat{\boldsymbol{\Psi}}$ as the MLE, even in situations where it may not globally maximize the likelihood. Indeed, in some of the examples on mixture models (McLachlan and Peel 2000, Chap. 3), the likelihood is unbounded. However, for these models there may still exist under the usual regularity conditions a sequence of roots of (6.1) with the properties of consistency, efficiency, and asymptotic normality (McLachlan and Basford 1988, Chap. 12).

When the likelihood or log likelihood is quadratic in the parameters as in the case of independent normally distributed observations, its maximum can be obtained by solving a system of linear equations in parameters. However, often in practice the likelihood function is not quadratic giving rise to nonlinearity problems in ML estimation. Examples of such situations are: (a) models leading to means which are nonlinear in parameters; (b) despite a possible linear structure, the likelihood is not quadratic in parameters due to, for instance, non-normal errors, missing data, or dependence.

Traditionally ML estimation in these situations has been carried out using numerical iterative methods of solution of equations such as the Newton–Raphson (NR) method and its variants like Fisher's method of scoring. Under reasonable assumptions on $L(\boldsymbol{\Psi})$ and a sufficiently accurate starting value, the sequence of iterates $\{\boldsymbol{\Psi}^{(k)}\}$ produced by the NR method enjoys local quadratic convergence to a solution $\boldsymbol{\Psi}^*$ of (6.1). Quadratic convergence is regarded as the major strength of

the NR method. But in applications, these methods could be tedious analytically and computationally even in fairly simple cases; see McLachlan and Krishnan (2008, Sect. 1.3) and Meng and van Dyk (1997). The EM algorithm offers an attractive alternative in a variety of settings. It is now a popular tool for iterative ML estimation in a variety of problems involving missing data or incomplete information.

6.1.2 Idea Behind the EM Algorithm: Incomplete-Data Structure

In the application of statistical methods, one is often faced with the problem of estimation of parameters when the likelihood function is complicated in structure resulting in difficult-to-compute maximization problems. This difficulty could be analytical or computational or both. Some examples are grouped, censored or truncated data, multivariate data with some missing observations, multiway frequency data with a complex cell probability structure, and data from mixtures of distributions. In many of these problems, it is often possible to formulate an associated statistical problem with the same parameters with "augmented data" from which it is possible to work out the MLE in an analytically and computationally simpler manner. The augmented data could be called the "complete data" and the available data could be called the "incomplete data", and the corresponding likelihoods, the "complete-data likelihood" and the "incomplete-data likelihood", respectively. The EM Algorithm is a generic method for computing the MLE of an incomplete-data problem by formulating an associated complete-data problem, and exploiting the simplicity of the MLE of the latter to compute the MLE of the former. The augmented part of the data could also be called "missing data", with respect to the actual incomplete-data problem on hand. The missing data need not necessarily be missing in the practical sense of the word. It may just be a conceptually convenient technical device. Thus the phrase "incomplete data" is used quite broadly to represent a variety of statistical data models, including mixtures, convolutions, random effects, grouping, censoring, truncated and missing observations.

A brief history of the EM algorithm can be found in McLachlan and Krishnan (2008, Sect. 1.8). In their fundamental paper, Dempster et al. (1977) synthesized earlier formulations of this algorithm in many particular cases and presented a general formulation of this method of finding MLE in a variety of problems. Since then the EM algorithm has been applied in a staggering variety of general statistical problems such as resolution of mixtures, multiway contingency tables, variance components estimation, factor analysis, as well as in specialized applications in such areas as genetics, medical imaging, and neural networks.

6.2 Basic Theoretical Framework of the EM Algorithm

6.2.1 The E- and M-Steps

Within the incomplete-data framework of the EM algorithm, we let $\boldsymbol{x}$ denote the vector containing the complete data and we let $\boldsymbol{z}$ denote the vector containing the missing data. Even when a problem does not at first appear to be an incomplete-data one, computation of the MLE is often greatly facilitated by artificially formulating it to be as such. This is because the EM algorithm exploits the reduced complexity of ML estimation given the complete data. For many statistical problems the complete-data likelihood has a nice form.

We let $g_c(\boldsymbol{x}; \boldsymbol{\Psi})$ denote the p.d.f. of the random vector $\boldsymbol{X}$ corresponding to the complete-data vector $\boldsymbol{x}$. Then the complete-data log likelihood function that could be formed for $\boldsymbol{\Psi}$ if $\boldsymbol{x}$ were fully observable is given by

$$\log L_c(\boldsymbol{\Psi}) = \log g_c(\boldsymbol{x}; \boldsymbol{\Psi}).$$

The EM algorithm approaches the problem of solving the incomplete-data likelihood equation (6.1) indirectly by proceeding iteratively in terms of $\log L_c(\boldsymbol{\Psi})$. As it is unobservable, it is replaced by its conditional expectation given $\boldsymbol{y}$, using the current fit for $\boldsymbol{\Psi}$. On the $(k+1)$th iteration of the EM algorithm,
E-Step: Compute $Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)})$, where

$$Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)}) = E_{\boldsymbol{\Psi}^{(k)}}\{\log L_c(\boldsymbol{\Psi})|\boldsymbol{y}\}. \tag{6.2}$$

M-Step: Choose $\boldsymbol{\Psi}^{(k+1)}$ to be any value of $\boldsymbol{\Psi} \in \boldsymbol{\Omega}$ that maximizes $Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)})$:

$$Q(\boldsymbol{\Psi}^{(k+1)}; \boldsymbol{\Psi}^{(k)}) \geq Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)}) \quad \forall \boldsymbol{\Psi} \in \boldsymbol{\Omega}. \tag{6.3}$$

The E- and M-steps are alternated repeatedly until convergence, which may be determined, for instance, by using a suitable stopping rule like $\|\boldsymbol{\Psi}^{(k+1)} - \boldsymbol{\Psi}^{(k)}\| < \varepsilon$ for some $\varepsilon > 0$ with some appropriate norm $\|\cdot\|$ or the difference $L(\boldsymbol{\Psi}^{(k+1)}) - L(\boldsymbol{\Psi}^{(k)})$ changes by an arbitrarily small amount in the case of convergence of the sequence of likelihood values $\{L(\boldsymbol{\Psi}^{(k)})\}$.

It can be shown that both the E- and M-steps will have particularly simple forms when $g_c(\boldsymbol{x}; \boldsymbol{\Psi})$ is from an exponential family:

$$g_c(\boldsymbol{x}; \boldsymbol{\Psi}) = b(\boldsymbol{x}) \exp\{\boldsymbol{c}^{\top}(\boldsymbol{\Psi})\boldsymbol{t}(\boldsymbol{x})\}/a(\boldsymbol{\Psi}), \tag{6.4}$$

where $\boldsymbol{t}(\boldsymbol{x})$ is a $k \times 1$ $(k \geq d)$ vector of complete-data sufficient statistics and $\boldsymbol{c}(\boldsymbol{\Psi})$ is a $k \times 1$ vector function of the parameter vector $\boldsymbol{\Psi}$, and $a(\boldsymbol{\Psi})$ and $b(\boldsymbol{x})$ are scalar functions. Here d is the number of unknown parameters in $\boldsymbol{\Psi}$. Members of the exponential family include most common distributions, such as the multivariate

normal, Poisson, multinomial and others. For exponential families, the E-step can be written as

$$Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)}) = E_{\boldsymbol{\Psi}^{(k)}}(\log b(\boldsymbol{x})|\boldsymbol{y}) + \boldsymbol{c}^{\top}(\boldsymbol{\Psi})\boldsymbol{t}^{(k)} - \log a(\boldsymbol{\Psi}),$$

where $\boldsymbol{t}^{(k)} = E_{\boldsymbol{\Psi}^{(k)}}\{\boldsymbol{t}(\boldsymbol{X})|\boldsymbol{y}\}$ is an estimator of the sufficient statistic. The M-step maximizes the Q-function with respect to $\boldsymbol{\Psi}$; but $E_{\boldsymbol{\Psi}^{(k)}}(\log b(\boldsymbol{x})|\boldsymbol{y})$ does not depend on $\boldsymbol{\Psi}$. Hence it is sufficient to write:
E–Step: Compute

$$\boldsymbol{t}^{(k)} = E_{\boldsymbol{\Psi}^{(k)}}\{\boldsymbol{t}(\boldsymbol{X})|\boldsymbol{y}\}.$$

M–Step: Compute

$$\boldsymbol{\Psi}^{(k+1)} = \arg\max_{\boldsymbol{\Psi}}[\boldsymbol{c}^{\top}(\boldsymbol{\Psi})\boldsymbol{t}^{(k)} - \log a(\boldsymbol{\Psi})].$$

In Example 2 of Sect. 6.3.2, the complete-data p.d.f. has an exponential family representation. We shall show how the implementation of the EM algorithm can be simplified.

6.2.2 Generalized EM Algorithm

Often in practice, the solution to the M-step exists in closed form. In those instances where it does not, it may not be feasible to attempt to find the value of $\boldsymbol{\Psi}$ that globally maximizes the function $Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)})$. For such situations, Dempster et al. (1977) defined a generalized EM (GEM) algorithm for which the M-Step requires $\boldsymbol{\Psi}^{(k+1)}$ to be chosen such that

$$Q(\boldsymbol{\Psi}^{(k+1)}; \boldsymbol{\Psi}^{(k)}) \geq Q(\boldsymbol{\Psi}^{(k)}; \boldsymbol{\Psi}^{(k)}) \tag{6.5}$$

holds. That is, one chooses $\boldsymbol{\Psi}^{(k+1)}$ to increase the Q-function, $Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)})$, over its value at $\boldsymbol{\Psi} = \boldsymbol{\Psi}^{(k)}$, rather than to maximize it over all $\boldsymbol{\Psi} \in \boldsymbol{\Omega}$ in (6.3).

It is of interest to note that the EM (GEM) algorithm as described above implicitly defines a mapping $\boldsymbol{\Psi} \to \boldsymbol{M}(\boldsymbol{\Psi})$, from the parameter space $\boldsymbol{\Omega}$ to itself such that

$$\boldsymbol{\Psi}^{(k+1)} = \boldsymbol{M}(\boldsymbol{\Psi}^{(k)}) \qquad (k = 0, 1, 2, \ldots).$$

The function $\boldsymbol{M}$ is called the EM mapping. We shall use this function in our subsequent discussion on the convergence property of the EM algorithm.

6.2.3 Convergence of the EM Algorithm

Let $k(\boldsymbol{x}|\boldsymbol{y}; \boldsymbol{\Psi}) = g_c(\boldsymbol{x}; \boldsymbol{\Psi})/g(\boldsymbol{y}; \boldsymbol{\Psi})$ be the conditional density of $\boldsymbol{X}$ given $\boldsymbol{Y} = \boldsymbol{y}$. Then the complete-data log likelihood can be expressed by

$$\log L_c(\boldsymbol{\Psi}) = \log g_c(\boldsymbol{x}; \boldsymbol{\Psi}) = \log L(\boldsymbol{\Psi}) + \log k(\boldsymbol{x}|\boldsymbol{y}; \boldsymbol{\Psi}). \tag{6.6}$$

Taking expectations on both sides of (6.6) with respect to the conditional distribution $\boldsymbol{x}|\boldsymbol{y}$ using the fit $\boldsymbol{\Psi}^{(k)}$ for $\boldsymbol{\Psi}$, we have

$$Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)}) = \log L(\boldsymbol{\Psi}) + H(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)}), \tag{6.7}$$

where $H(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)}) = E_{\boldsymbol{\Psi}^{(k)}}\{\log k(\boldsymbol{X}|\boldsymbol{y}; \boldsymbol{\Psi})|\boldsymbol{y}\}$. It follows from (6.7) that

$$\begin{aligned}\log L(\boldsymbol{\Psi}^{(k+1)}) - \log L(\boldsymbol{\Psi}^{(k)}) = \{&Q(\boldsymbol{\Psi}^{(k+1)}; \boldsymbol{\Psi}^{(k)}) - Q(\boldsymbol{\Psi}^{(k)}; \boldsymbol{\Psi}^{(k)})\} \\ &-\{H(\boldsymbol{\Psi}^{(k+1)}; \boldsymbol{\Psi}^{(k)}) - H(\boldsymbol{\Psi}^{(k)}; \boldsymbol{\Psi}^{(k)})\}. \end{aligned} \tag{6.8}$$

By Jensen's inequality and the concavity of the logarithmic function, we have $H(\boldsymbol{\Psi}^{(k+1)}; \boldsymbol{\Psi}^{(k)}) \leq H(\boldsymbol{\Psi}^{(k)}; \boldsymbol{\Psi}^{(k)})$. From (6.3) or (6.5), the first difference on the right-hand side of (6.8) is nonnegative. Hence, the likelihood function is not decreased after an EM or GEM iteration:

$$L(\boldsymbol{\Psi}^{(k+1)}) \geq L(\boldsymbol{\Psi}^{(k)}) \qquad (k = 0, 1, 2, \ldots). \tag{6.9}$$

A consequence of (6.9) is the self-consistency of the EM algorithm. Thus for a bounded sequence of likelihood values $\{L(\boldsymbol{\Psi}^{(k)})\}$, $L(\boldsymbol{\Psi}^{(k)})$ converges monotonically to some L^*. Now questions naturally arise as to the conditions under which L^* corresponds to a stationary value and when this stationary value is at least a local maximum if not a global maximum. Examples are known where the EM algorithm converges to a local *minimum* and to a saddle point of the likelihood (McLachlan and Krishnan 2008, Sect. 3.6). There are also questions of convergence of the sequence of EM iterates, that is, of the sequence of parameter values $\{\boldsymbol{\Psi}^{(k)}\}$ to the MLE.

Wu (1983) investigates in detail several convergence issues of the EM algorithm in its generality, and their relationship to other optimization methods. He shows that when the complete data are from a curved exponential family with compact parameter space, and when the Q-function satisfies a certain mild differentiability condition, then any EM sequence converges to a stationary point (not necessarily a maximum) of the likelihood function. If $L(\boldsymbol{\Psi})$ has multiple stationary points, convergence of the EM sequence to either type (local or global maximizers, saddle points) depends upon the starting value $\boldsymbol{\Psi}^{(0)}$ for $\boldsymbol{\Psi}$. If $L(\boldsymbol{\Psi})$ is unimodal in $\boldsymbol{\Omega}$ and satisfies the same differentiability condition, then any sequence $\{\boldsymbol{\Psi}^{(k)}\}$ will converge to the unique MLE of $\boldsymbol{\Psi}$, irrespective of its starting value.

To be more specific, one of the basic convergence results of the EM algorithm is the following:

$$\log L(\boldsymbol{M}(\boldsymbol{\Psi})) \geq \log L(\boldsymbol{\Psi})$$

with equality if and only if

$$Q(\boldsymbol{M}(\boldsymbol{\Psi}); \boldsymbol{\Psi}) = Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}) \quad \text{and} \quad k(\boldsymbol{x}|\boldsymbol{y}; \boldsymbol{M}(\boldsymbol{\Psi})) = k(\boldsymbol{x}|\boldsymbol{y}; \boldsymbol{\Psi}).$$

This means that the likelihood function increases at each iteration of the EM algorithm, until the condition for equality is satisfied and a fixed point of the iteration is reached. If $\hat{\boldsymbol{\Psi}}$ is an MLE, so that $\log L(\hat{\boldsymbol{\Psi}}) \geq \log L(\boldsymbol{\Psi}),\ \forall\ \boldsymbol{\Psi} \in \boldsymbol{\Omega}$, then $\log L(\boldsymbol{M}(\hat{\boldsymbol{\Psi}})) = \log L(\hat{\boldsymbol{\Psi}})$. Thus MLE are fixed points of the EM algorithm. If we have the likelihood function bounded (as might happen in many cases of interest), the EM sequence $\{\boldsymbol{\Psi}^{(k)}\}$ yields a bounded nondecreasing sequence $\{\log L(\boldsymbol{\Psi}^{(k)})\}$ which must converge as $k \to \infty$.

The theorem does not quite imply that fixed points of the EM algorithm are in fact MLEs. This is however true under fairly general conditions. For proofs and other details, see McLachlan and Krishnan (2008, Sect. 3.5) and Wu (1983). Furthermore, if a sequence of EM iterates $\{\boldsymbol{\Psi}^{(k)}\}$ satisfy the conditions

1. $[\partial Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)})/\partial \boldsymbol{\Psi}]_{\boldsymbol{\Psi}=\boldsymbol{\Psi}^{(k+1)}} = \mathbf{0}$, and
2. The sequence $\{\boldsymbol{\Psi}^{(k)}\}$ converges to some value $\boldsymbol{\Psi}^*$ and $\log k(\boldsymbol{x}|\boldsymbol{y}; \boldsymbol{\Psi})$ is sufficiently smooth,

then we have $[\partial \log L(\boldsymbol{\Psi})/\partial \boldsymbol{\Psi}]_{\boldsymbol{\Psi}=\boldsymbol{\Psi}^*} = \mathbf{0}$; see Little and Rubin (2002) and Wu (1983). Thus, despite the earlier convergence results, there is no guarantee that the convergence will be to a global maximum. For likelihood functions with multiple maxima, convergence will be to a local maximum which depends on the starting value $\boldsymbol{\Psi}^{(0)}$.

In some estimation problems with constrained parameter spaces, the parameter value maximizing the log likelihood is on the boundary of the parameter space. Here some elements of the EM sequence may lie on the boundary, thus not fulfilling Wu's conditions for convergence. Nettleton (1999) extends Wu's convergence results to the case of constrained parameter spaces and establishes some stricter conditions to guarantee convergence of the EM likelihood sequence to some local maximum and the EM parameter iterates to converge to the MLE.

6.2.4 *Rate of Convergence of the EM Algorithm*

The rate of convergence of the EM algorithm is usually slower than the quadratic convergence typically available with Newton-type methods. Dempster et al. (1977) show that the rate of convergence of the EM algorithm is linear and the rate depends on the proportion of information in the observed data. Thus in comparison to the formulated complete-data problem, if a large portion of data is missing, convergence can be quite slow.

Recall the EM mapping $\boldsymbol{M}$ defined in Sect. 6.2.2. If $\boldsymbol{\Psi}^{(k)}$ converges to some point $\boldsymbol{\Psi}^*$ and $\boldsymbol{M}(\boldsymbol{\Psi})$ is continuous, then $\boldsymbol{\Psi}^*$ is a fixed point of the algorithm; that is, $\boldsymbol{\Psi}^*$ must satisfy $\boldsymbol{\Psi}^* = \boldsymbol{M}(\boldsymbol{\Psi}^*)$. By a Taylor series expansion of $\boldsymbol{\Psi}^{(k+1)} = \boldsymbol{M}(\boldsymbol{\Psi}^{(k)})$ about the point $\boldsymbol{\Psi}^{(k)} = \boldsymbol{\Psi}^*$, we have in a neighborhood of $\boldsymbol{\Psi}^*$ that

$$\boldsymbol{\Psi}^{(k+1)} - \boldsymbol{\Psi}^* \approx \boldsymbol{J}(\boldsymbol{\Psi}^*)(\boldsymbol{\Psi}^{(k)} - \boldsymbol{\Psi}^*),$$

where $\boldsymbol{J}(\boldsymbol{\Psi})$ is the $d \times d$ Jacobian matrix for $\boldsymbol{M}(\boldsymbol{\Psi}) = (M_1(\boldsymbol{\Psi}), \ldots, M_d(\boldsymbol{\Psi}))^\top$, having (i, j)th element $r_{ij}(\boldsymbol{\Psi})$ equal to

$$r_{ij}(\boldsymbol{\Psi}) = \partial M_i(\boldsymbol{\Psi}) / \partial \Psi_j,$$

where $\Psi_j = (\boldsymbol{\Psi})_j$ and d is the dimension of $\boldsymbol{\Psi}$. Thus, in a neighborhood of $\boldsymbol{\Psi}^*$, the EM algorithm is essentially a linear iteration with rate matrix $\boldsymbol{J}(\boldsymbol{\Psi}^*)$, since $J(\boldsymbol{\Psi}^*)$ is typically nonzero. For this reason, $\boldsymbol{J}(\boldsymbol{\Psi}^*)$ is often referred to as the matrix rate of convergence. For vector $\boldsymbol{\Psi}$, a measure of the actual observed convergence rate is the global rate of convergence, which is defined as

$$r = \lim_{k \to \infty} \| \boldsymbol{\Psi}^{(k+1)} - \boldsymbol{\Psi}^* \| / \| \boldsymbol{\Psi}^{(k)} - \boldsymbol{\Psi}^* \|,$$

where $\| \cdot \|$ is any norm on d-dimensional Euclidean space $\Re^d$. It is noted that the observed rate of convergence equals the largest eigenvalue of $\boldsymbol{J}(\boldsymbol{\Psi}^*)$ under certain regularity conditions (Meng and van Dyk 1997). As a large value of r implies slow convergence, the global speed of convergence is defined to be $s = 1 - r$ (Meng 1994); see also McLachlan and Krishnan (2008, Sect. 3.9).

6.2.5 *Initialization of the EM Algorithm*

The EM algorithm will converge very slowly if a poor choice of initial value $\boldsymbol{\Psi}^{(0)}$ were used. Indeed, in some cases where the likelihood is unbounded on the edge of the parameter space, the sequence of estimates $\{\boldsymbol{\Psi}^{(k)}\}$ generated by the EM algorithm may diverge if $\boldsymbol{\Psi}^{(0)}$ is chosen too close to the boundary. Also, with applications where the likelihood equation has multiple roots corresponding to local maxima, the EM algorithm should be applied from a wide choice of starting values in any search for all local maxima. A variation of the EM algorithm (Wright and Kennedy 2000) uses interval analysis methods to locate multiple stationary points of a log likelihood within any designated region of the parameter space; see also McLachlan and Krishnan (2008, Sect. 7.9).

Different ways of specification of initial value have been considered specifically within the mixture models framework. With the EMMIX program (McLachlan and Peel 2000, pp. 343–344), an initial parameter value can be obtained automatically using either random partitions of the data, k-means clustering algorithm, or hierarchical clustering methods. With random starts, the effect of the central limit theorem tends to have the component parameters initially being similar at least in large samples. With the EMMIX program, there is an additional option for random starts to reduce this effect by first selecting a random subsample from the data, which is then randomly assigned to the g components. As described in McLachlan and Peel (2000, Sect. 2.12), the subsample has to be sufficiently large to ensure that the first M-step is able to produce a nondegenerate estimate of the parameter vector $\boldsymbol{\Psi}$.

Ueda and Nakano (1998) considered a deterministic annealing EM (DAEM) algorithm in order for the EM iterative process to be able to recover from a poor choice of starting value. They proposed using the principle of maximum entropy and the statistical mechanics analogy, whereby a parameter, say θ, is introduced with $1/\theta$ corresponding to the "temperature" in an annealing sense. With their DAEM algorithm, the E-step is effected by averaging $\log L_c(\boldsymbol{\Psi})$ over the distribution taken to be proportional to that of the current estimate of the conditional density of the complete data (given the observed data) raised to the power of θ; see for example McLachlan and Peel (2000, pp. 58–60). Recently, Pernkopf and Bouchaffra (2005) combined genetic algorithms (GA) and the EM algorithm for fitting normal mixtures, where the proposed algorithm is less sensitive to its initialization and enables escaping from local optimal solutions.

6.3 Examples of the EM Algorithm

6.3.1 Example 1: Normal Mixtures

One of the classical formulation of the statistical pattern recognition involves a mixture of p-dimensional normal distributions with a finite number, say g, of components in some unknown proportions $\pi_1, \ldots, \pi_g$ that sum to one. Here, we have n independent observations $\boldsymbol{y}_1, \boldsymbol{y}_2, \ldots, \boldsymbol{y}_n$ from the mixture density

$$f(\boldsymbol{y}; \boldsymbol{\Psi}) = \sum_{i=1}^{g} \pi_i \phi(\boldsymbol{y}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i),$$

where $\phi(\boldsymbol{y}; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ denotes the p-dimensional normal density function with mean vector $\boldsymbol{\mu}_i$ and covariance matrix $\boldsymbol{\Sigma}_i$ $(i = 1, \ldots, g)$. The vector $\boldsymbol{\Psi}$ of unknown parameters consists of the mixing proportions $\pi_1, \ldots, \pi_{g-1}$, the elements of the component means $\boldsymbol{\mu}_i$, and the distinct elements of the component-covariance matrices $\boldsymbol{\Sigma}_i$. The problem of estimating $\boldsymbol{\Psi}$ is an instance of the problem of resolution of mixtures or in pattern recognition parlance an "unsupervised learning problem".

Consider the corresponding "supervised learning problem", where observations on the random vector $\boldsymbol{X} = (\boldsymbol{Z}, \boldsymbol{Y})$ are $\boldsymbol{x}_1 = (\boldsymbol{z}_1, \boldsymbol{y}_1)$, $\boldsymbol{x}_2 = (\boldsymbol{z}_2, \boldsymbol{y}_2), \ldots, \boldsymbol{x}_n = (\boldsymbol{z}_n, \boldsymbol{y}_n)$. Here $\boldsymbol{z}_j$ is the unobservable component-indicator vector, where the ith element z_{ij} of $\boldsymbol{z}_j$ is taken to be one or zero according as the jth observation does or does not come from the ith component $(j = 1, \ldots, n)$. The MLE problem is far simpler here with easy closed-form MLE. The classificatory vectors $\boldsymbol{z} = (\boldsymbol{z}_1^\top, \ldots, \boldsymbol{z}_n^\top)^\top$ could be called the missing data. The unsupervised learning problem could be called the incomplete-data problem and the supervised learning problem the complete-data problem. A relatively simple iterative method for computing the

MLE for the unsupervised problem could be given exploiting the simplicity of the MLE for the supervised problem. This is the essence of the EM algorithm.

The complete-data log likelihood function for $\boldsymbol{\Psi}$ is given by

$$\log L_c(\boldsymbol{\Psi}) = \sum_{i=1}^{g}\sum_{j=1}^{n} z_{ij}\{\log \pi_i + \log \phi(\boldsymbol{y}_j; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\}. \tag{6.10}$$

Now the EM algorithm for this problem starts with some initial value $\boldsymbol{\Psi}^{(0)}$ for the parameters. As $\log L_c(\boldsymbol{\Psi})$ in (6.10) is a linear function of the unobservable data $\boldsymbol{z}$ for this problem, the calculation of $Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)})$ on the E-step is effected simply by replacing z_{ij} by its current conditional expectation given the observed data $\boldsymbol{y}$, which is the usual posterior probability of the jth observation arising from the ith component

$$\tau_{ij}^{(k)} = E_{\boldsymbol{\Psi}^{(k)}}(Z_{ij}|\boldsymbol{y}) = \frac{\pi_i^{(k)}\phi(\boldsymbol{y}_j; \boldsymbol{\mu}_i^{(k)}, \boldsymbol{\Sigma}_i^{(k)})}{\sum_{l=1}^{g}\pi_l^{(k)}\phi(\boldsymbol{y}_j; \boldsymbol{\mu}_l^{(k)}, \boldsymbol{\Sigma}_l^{(k)})}.$$

From (6.10), it follows that

$$Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)}) = \sum_{i=1}^{g}\sum_{j=1}^{n} \tau_{ij}^{(k)}\{\log \pi_i + \log \phi(\boldsymbol{y}_j; \boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)\}. \tag{6.11}$$

For mixtures with normal component densities, it is computationally advantageous to work in terms of the sufficient statistics (Ng and McLachlan 2003) given by

$$\begin{aligned} T_{i1}^{(k)} &= \sum_{j=1}^{n} \tau_{ij}^{(k)} \\ \boldsymbol{T}_{i2}^{(k)} &= \sum_{j=1}^{n} \tau_{ij}^{(k)} \boldsymbol{y}_j \\ \boldsymbol{T}_{i3}^{(k)} &= \sum_{j=1}^{n} \tau_{ij}^{(k)} \boldsymbol{y}_j \boldsymbol{y}_j^T. \end{aligned} \tag{6.12}$$

By differentiating (6.11) with respect to $\boldsymbol{\Psi}$ on the basis of the sufficient statistics in (6.12), the M-step exists in closed form as

$$\begin{aligned} \pi_i^{(k+1)} &= T_{i1}^{(k)}/n \\ \boldsymbol{\mu}_i^{(k+1)} &= \boldsymbol{T}_{i2}^{(k)}/T_{i1}^{(k)} \\ \boldsymbol{\Sigma}_i^{(k+1)} &= \{\boldsymbol{T}_{i3}^{(k)} - T_{i1}^{(k)^{-1}}\boldsymbol{T}_{i2}^{(k)}\boldsymbol{T}_{i2}^{(k)^T}\}/T_{i1}^{(k)}. \end{aligned} \tag{6.13}$$

The E- and M-steps are then iterated until convergence. Unlike in the MLE for the supervised problem, in the M-step of the unsupervised problem, the posterior probabilities τ_{ij}, which are between 0 and 1, are used. The mean vectors $\boldsymbol{\mu}_i$ and the covariance matrix $\boldsymbol{\Sigma}_i$ $(i = 1, \ldots, g)$ are computed using the $\tau_{ij}^{(k)}$ as weights in weighted averages.

In the case of unrestricted component-covariance matrices $\boldsymbol{\Sigma}_i$, $L(\boldsymbol{\Psi})$ is unbounded, as each data point gives rise to a singularity on the edge of the parameter space (McLachlan and Peel 2000, Sect. 3.8). In practice, the component-covariance matrices $\boldsymbol{\Sigma}_i$ can be restricted to being the same, $\boldsymbol{\Sigma}_i = \boldsymbol{\Sigma}$ $(i = 1, \ldots, g)$, where $\boldsymbol{\Sigma}$ is unspecified. In this case of homoscedastic normal components, the updated estimate of the common component-covariance matrix $\boldsymbol{\Sigma}$ is given by

$$\boldsymbol{\Sigma}^{(k+1)} = \sum_{i=1}^{g} T_{i1}^{(k)} \boldsymbol{\Sigma}_i^{(k+1)} / n,$$

where $\boldsymbol{\Sigma}_i^{(k+1)}$ is given by (6.13), and the updates of π_i and $\boldsymbol{\mu}_i$ are as above in the heteroscedastic case.

6.3.2 Example 2: Censored Failure-Time Data

In survival or reliability analyses, the focus is the distribution of time T to the occurrence of some event that represents failure (for computational methods in survival analysis see also Chap. III.27). In many situations, there will be individuals who do not fail at the end of the study, or individuals who withdraw from the study before it ends. Such observations are censored, as we know only that their failure times are greater than particular values. We let $\boldsymbol{y} = (c_1, \delta_1, \ldots, c_n, \delta_n)^\top$ denote the observed failure-time data, where $\delta_j = 0$ or 1 according as the jth observation T_j is censored or uncensored at c_j $(j = 1, \ldots, n)$. That is, if T_j is uncensored, $t_j = c_j$, whereas if $t_j > c_j$, it is censored at c_j.

In the particular case where the p.d.f. for T is exponential with mean μ, we have

$$f(t; \mu) = \mu^{-1} \exp(-t/\mu) I_{(0,\infty)}(t) \qquad (\mu > 0), \tag{6.14}$$

where the indicator function $I_{(0,\infty)}(t) = 1$ for $t > 0$ and is zero elsewhere. The unknown parameter vector $\boldsymbol{\Psi}$ is now a scalar, being equal to μ. Denote by s the number of uncensored observations. By re-ordering the data so that the uncensored observations precede censored observations. It can be shown that the log likelihood function for μ is given by

$$\log L(\mu) = -s \log \mu - \sum_{j=1}^{n} c_j / \mu. \tag{6.15}$$

By equating the derivative of (6.15) to zero, the MLE of μ is

$$\hat{\mu} = \sum_{j=1}^{n} c_j / s. \tag{6.16}$$

Thus there is no need for the iterative computation of $\hat{\mu}$. But in this simple case, it is instructive to demonstrate how the EM algorithm would work and how its implementation could be simplified as the complete-data log likelihood belongs to the regular exponential family (see Sect. 6.2.1).

The complete-data vector $\boldsymbol{x}$ can be declared to be $\boldsymbol{x} = (t_1, \ldots, t_s, \boldsymbol{z}^\top)^\top$, where $\boldsymbol{z} = (t_{s+1}, \ldots, t_n)^\top$ contains the unobservable realizations of the $n - s$ censored random variables. The complete-data log likelihood is given by

$$\log L_c(\mu) = -n \log \mu - \sum_{j=1}^{n} t_j / \mu. \tag{6.17}$$

As $\log L_c(\mu)$ is a linear function of the unobservable data $\boldsymbol{z}$, the E-step is effected simply by replacing $\boldsymbol{z}$ by its current conditional expectation given $\boldsymbol{y}$. By the lack of memory of the exponential distribution, the conditional distribution of $T_j - c_j$ given that $T_j > c_j$ is still exponential with mean μ. So, we have

$$E_{\mu^{(k)}}(T_j | \boldsymbol{y}) = E_{\mu^{(k)}}(T_j | T_j > c_j) = c_j + \mu^{(k)} \tag{6.18}$$

for $j = s + 1, \ldots, n$. Accordingly, the Q-function is given by

$$Q(\mu; \mu^{(k)}) = -n \log \mu - \mu^{-1} \left\{ \sum_{j=1}^{n} c_j + (n - s)\mu^{(k)} \right\}.$$

In the M-step, we have

$$\mu^{(k+1)} = \left\{ \sum_{j=1}^{n} c_j + (n - s)\mu^{(k)} \right\} \Big/ n. \tag{6.19}$$

On putting $\mu^{(k+1)} = \mu^{(k)} = \mu^*$ in (6.19) and solving for μ^*, we have for $s < n$ that $\mu^* = \hat{\mu}$. That is, the EM sequence $\{\mu^{(k)}\}$ has the MLE $\hat{\mu}$ as its unique limit point, as $k \to \infty$; see McLachlan and Krishnan (2008, Sect. 1.5.2).

From (6.17), it can be seen that $\log L_c(\mu)$ has the exponential family form (6.4) with canonical parameter μ^{-1} and sufficient statistic $\boldsymbol{t}(\boldsymbol{X}) = \sum_{j=1}^{n} T_j$. Hence, from (6.18), the E-step requires the calculation of $\boldsymbol{t}^{(k)} = \sum_{j=1}^{n} c_j + (n - s)\mu^{(k)}$. The M-step then yields $\mu^{(k+1)}$ as the value of μ that satisfies the equation

$$\boldsymbol{t}^{(k)} = E_{\mu}\{\boldsymbol{t}(\boldsymbol{X})\} = n\mu.$$

This latter equation can be seen to be equivalent to (6.19), as derived by direct differentiation of the Q-function.

6.3.3 Example 3: Mixture-of-Experts Models

Among the various kinds of modular networks, mixtures-of-experts (Jacobs et al. 1991) and hierarchical mixtures-of-experts (Jordan and Jacobs 1994) are of much interest due to their wide applicability and the advantage of fast learning via the EM algorithm (Jordan and Xu 1995; Ng and McLachlan 2004a). In mixture-of-experts (ME) networks, there are a finite number, say m, of modules, referred to as expert networks. These expert networks approximate the distribution of the output $\boldsymbol{y}_j$ within each region of the input space. The expert network maps its input $\boldsymbol{x}_j$ to an output, the density $f_h(\boldsymbol{y}_j|\boldsymbol{x}_j;\boldsymbol{\theta}_h)$, where $\boldsymbol{\theta}_h$ is a vector of unknown parameters for the hth expert network. It is assumed that different experts are appropriate in different regions of the input space. The gating network provides a set of scalar coefficients $\pi_h(\boldsymbol{x}_j;\boldsymbol{\alpha})$ that weight the contributions of the various experts, where $\boldsymbol{\alpha}$ is a vector of unknown parameters in the gating network. Therefore, the final output of the ME neural network is a weighted sum of all the output vectors produced by expert networks:

$$f(\boldsymbol{y}_j|\boldsymbol{x}_j;\boldsymbol{\Psi}) = \sum_{h=1}^{m} \pi_h(\boldsymbol{x}_j;\boldsymbol{\alpha}) f_h(\boldsymbol{y}_j|\boldsymbol{x}_j;\boldsymbol{\theta}_h), \tag{6.20}$$

where $\boldsymbol{\Psi} = (\boldsymbol{\alpha}^{\top}, \boldsymbol{\theta}_1^{\top}, \ldots, \boldsymbol{\theta}_m^{\top})^{\top}$ is the vector of all the unknown parameters. The output of the gating network is modeled by the softmax function as

$$\pi_h(\boldsymbol{x};\boldsymbol{\alpha}) = \frac{\exp(\boldsymbol{v}_h^{\top}\boldsymbol{x})}{\sum_{l=1}^{m}\exp(\boldsymbol{v}_l^{\top}\boldsymbol{x})} \qquad (h = 1,\ldots,m), \tag{6.21}$$

where $\boldsymbol{v}_h$ is the weight vector of the hth expert in the gating network and $\boldsymbol{v}_m = \boldsymbol{0}$. It is implicitly assumed that the first element of $\boldsymbol{x}$ is one, to account for an intercept term. It follows from (6.21) that $\boldsymbol{\alpha}$ contains the elements in $\boldsymbol{v}_h$ $(h = 1,\ldots,m-1)$.

To apply the EM algorithm to the ME networks, we introduce the indicator variables z_{hj}, where z_{hj} is one or zero according to whether $\boldsymbol{y}_j$ belongs or does not belong to the hth expert (Ng and McLachlan 2004a). The complete-data log likelihood for $\boldsymbol{\Psi}$ is given by

$$\log L_c(\boldsymbol{\Psi}) = \sum_{j=1}^{n}\sum_{h=1}^{m} z_{hj}\{\log \pi_h(\boldsymbol{x}_j;\boldsymbol{\alpha}) + \log f_h(\boldsymbol{y}_j|\boldsymbol{x}_j;\boldsymbol{\theta}_h)\}. \tag{6.22}$$

On the $(k+1)$th iteration, the E-step calculates the Q-function as

$$\begin{aligned} Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)}) &= E_{\boldsymbol{\Psi}^{(k)}}\{\log L_c(\boldsymbol{\Psi})|\boldsymbol{y}, \boldsymbol{x}\} \\ &= \sum_{j=1}^{n}\sum_{h=1}^{m} E_{\boldsymbol{\Psi}^{(k)}}(Z_{hj}|\boldsymbol{y}, \boldsymbol{x})\{\log \pi_h(\boldsymbol{x}_j; \boldsymbol{\alpha}) + \log f_h(\boldsymbol{y}_j|\boldsymbol{x}_j; \boldsymbol{\theta}_h)\} \\ &= Q_\alpha + Q_\theta, \end{aligned} \tag{6.23}$$

where the Q-function can be decomposed into two terms with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\theta}_h$ $(h = 1, \ldots, m)$, respectively, as

$$Q_\alpha = \sum_{j=1}^{n}\sum_{h=1}^{m} \tau_{hj}^{(k)} \log \pi_h(\boldsymbol{x}_j; \boldsymbol{\alpha}), \tag{6.24}$$

and

$$Q_\theta = \sum_{j=1}^{n}\sum_{h=1}^{m} \tau_{hj}^{(k)} \log f_h(\boldsymbol{y}_j|\boldsymbol{x}_j; \boldsymbol{\theta}_h), \tag{6.25}$$

where

$$\begin{aligned} \tau_{hj}^{(k)} &= E_{\boldsymbol{\Psi}^{(k)}}(Z_{hj}|\boldsymbol{y}, \boldsymbol{x}) \\ &= \pi_h(\boldsymbol{x}_j; \boldsymbol{\alpha}^{(k)}) f_h(\boldsymbol{y}_j|\boldsymbol{x}_j; \boldsymbol{\theta}_h^{(k)}) / \sum_{r=1}^{m} \pi_r(\boldsymbol{x}_j; \boldsymbol{\alpha}^{(k)}) f_r(\boldsymbol{y}_j|\boldsymbol{x}_j; \boldsymbol{\theta}_r^{(k)}) \end{aligned}$$

is the current estimated posterior probability that $\boldsymbol{y}_j$ belongs to the hth expert $(h = 1, \ldots, m)$.

Hence, the M-step consists of two separate maximization problems. With the gating network (6.21), the updated estimate of $\boldsymbol{\alpha}^{(k+1)}$ is obtained by solving

$$\sum_{j=1}^{n}\left(\tau_{hj}^{(k)} - \frac{\exp(\boldsymbol{v}_h^\top \boldsymbol{x}_j)}{1 + \sum_{l=1}^{m-1}\exp(\boldsymbol{v}_l^\top \boldsymbol{x}_j)}\right)\boldsymbol{x}_j = 0 \qquad (h = 1, \ldots, m-1), \tag{6.26}$$

which is a set of non-linear equations with $(m-1)p$ unknown parameters, where p is the dimension of $\boldsymbol{x}_j$ $(j = 1, \ldots, n)$. It can be seen from (6.26) that the non-linear equation for the hth expert depends not only on the parameter vector $\boldsymbol{v}_h$, but also on other parameter vectors $\boldsymbol{v}_l$ $(l = 1, \ldots, m-1)$. In other words, each parameter vector $\boldsymbol{v}_h$ cannot be updated independently. With the iterative reweighted least squares (IRLS) algorithm presented in Jordan and Jacobs (1994), the independence assumption on these parameter vectors was used implicitly and each parameter vector was updated independently and in parallel as

$$\boldsymbol{v}_h^{(s+1)} = \boldsymbol{v}_h^{(s)} - \gamma_\alpha \left(\frac{\partial^2 Q_\alpha}{\partial \boldsymbol{v}_h \boldsymbol{v}_h^\top} \right)^{-1} \frac{\partial Q_\alpha}{\partial \boldsymbol{v}_h} \qquad (h = 1, \ldots, m-1), \tag{6.27}$$

where $\gamma_\alpha \leq 1$ is the learning rate (Jordan and Xu 1995). That is, there are $m-1$ sets of non-linear equations each with p variables instead of a set of non-linear equations with $(m-1)p$ variables. In Jordan and Jacobs (1994), the iteration (6.27) is referred to as the inner loop of the EM algorithm. This inner loop is terminated when the algorithm has converged or the algorithm has still not converged after some pre-specified number of iterations. The above independence assumption on the parameter vectors is equivalent to the adoption of an incomplete Hessian matrix of the Q-function (Ng and McLachlan 2004a).

The densities $f_h(\boldsymbol{y}_j|\boldsymbol{x}_j;\boldsymbol{\theta}_h)$ $(h = 1, \ldots, m)$ can be assumed to belong to the exponential family (Jordan and Jacobs 1994). In this case, the ME model (6.20) will have the form of a mixture of generalized linear models (McLachlan and Peel 2000, Sect. 5.13). The updated estimate of $\boldsymbol{\theta}_h^{(k+1)}$ is obtained by solving

$$\sum_{j=1}^{n} \tau_{hj}^{(k)} \partial \log f_h(\boldsymbol{y}_j|\boldsymbol{x}_j;\boldsymbol{\theta}_h)/\partial \boldsymbol{\theta}_h = \boldsymbol{0} \qquad (h = 1, \ldots, m). \tag{6.28}$$

Equation (6.28) can be solved separately for each expert $(h = 1, \ldots, m)$ when the density $f_h(\boldsymbol{y}_j|\boldsymbol{x}_j;\boldsymbol{\theta}_h)$ is assumed to be normally distributed. With some other members of the exponential family such as multinomial distribution, (6.28) requires iterative methods to solve; see Example 5 in Sect. 6.4.2.

6.3.4 Misconceptions on the E-Step

Examples 1 to 3 may have given an impression that the E-step consists in simply replacing the missing data by their conditional expectations given the observed data at current parameter values. However, this will be valid only if the complete-data log likelihood $\log L_c(\boldsymbol{\Psi})$ were a linear function of the missing data $\boldsymbol{z}$. Unfortunately, it is not always true in general. Rather, as should be clear from the general theory described in Sect. 6.2.1, the E-step consists in replacing $\log L_c(\boldsymbol{\Psi})$ by its conditional expectation given the observed data at current parameter values. Flury and Zoppé (2000) give an example to demonstrate the point that the E-step does not always consist in plugging in "estimates" for missing data. Similar misconceptions exist in the applications of the EM algorithm to train neural networks. Let

$$(\boldsymbol{x}_1^\top, \boldsymbol{y}_1^\top)^\top, \ldots, (\boldsymbol{x}_n^\top, \boldsymbol{y}_n^\top)^\top \tag{6.29}$$

denote the n examples available for training a neural network, where $\boldsymbol{x}_j$ is an input feature vector and $\boldsymbol{y}_j$ is an output vector $(j = 1, \ldots, n)$. In the training process, the

unknown parameters in the neural network, denoted by a vector $\boldsymbol{\Psi}$, are inferred from the observed training data given by (6.29). We let $\boldsymbol{x} = (\boldsymbol{x}_1^\top, \ldots, \boldsymbol{x}_n^\top)^\top$ and $\boldsymbol{y} = (\boldsymbol{y}_1^\top, \ldots, \boldsymbol{y}_n^\top)^\top$. In order to estimate $\boldsymbol{\Psi}$ by the statistical technique of maximum likelihood, we have to impose a statistical distribution for the observed data (6.29), which will allow us to form a log likelihood function, $\log L(\boldsymbol{\Psi}; \boldsymbol{y}, \boldsymbol{x})$, for $\boldsymbol{\Psi}$. In general, we proceed conditionally on the values for the input variable $\boldsymbol{x}$; that is, we shall consider the specification of the conditional distribution of the random variable $\boldsymbol{Y}$ corresponding to the observed output $\boldsymbol{y}$ given the input $\boldsymbol{x}$; see, for example, (6.20) in Sect. 6.3.3.

Within the EM framework, the unknown vector $\boldsymbol{\Psi}$ is estimated by consideration of the complete-data log likelihood formed on the basis of both the observed and the missing data $\boldsymbol{z}$, $\log L_c(\boldsymbol{\Psi}; \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{x})$. On the $(k+1)$th iteration of the EM algorithm, the E-step computes the Q-function, which is given by

$$Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)}) = E_{\boldsymbol{\Psi}^{(k)}}\{\log L_c(\boldsymbol{\Psi}; \boldsymbol{y}, \boldsymbol{z}, \boldsymbol{x}) | \boldsymbol{y}, \boldsymbol{x}\}. \tag{6.30}$$

In some instances, a modified form of the EM algorithm is being used unwittingly in that on the E-step, the Q-function is effected simply by replacing the random vector $\boldsymbol{z}$ by its conditional expectation. That is, (6.30) is computed by the approximation

$$Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)}) \approx \log L_c(\boldsymbol{\Psi}; \boldsymbol{y}, \tilde{\boldsymbol{z}}, \boldsymbol{x}), \tag{6.31}$$

where

$$\tilde{\boldsymbol{z}} = E_{\boldsymbol{\Psi}^{(k)}}\{\boldsymbol{Z} | \boldsymbol{y}, \boldsymbol{x}\}.$$

As described above, the approximation (6.31) will be invalid when the complete-data log likelihood is non-linear in $\boldsymbol{z}$, for example, in the multilayer perceptron networks or the radial basis function networks with regression weights; see Ng and McLachlan (2004a).

6.3.5 *Provision of Standard Errors*

Several methods have been suggested in the EM literature for augmenting the EM computation with some computation for obtaining an estimate of the covariance matrix of the computed MLE. Many such methods attempt to exploit the computations in the EM steps. These methods are based on the observed information matrix $\boldsymbol{I}(\hat{\boldsymbol{\Psi}}; \boldsymbol{y})$, the expected information matrix $\mathcal{I}(\boldsymbol{\Psi})$ or on resampling methods. Baker (1992) reviews such methods and also develops a method for computing the observed information matrix in the case of categorical data. Jamshidian and Jennrich (2000) review more recent methods including the Supplemented EM (SEM) algorithm of Meng and Rubin (1991) and suggest some newer methods based on numerical differentiation.

Theoretically one may compute the asymptotic covariance matrix by inverting the observed or expected information matrix at the MLE. In practice, however, this

may be tedious analytically or computationally, defeating one of the advantages of the EM approach. Louis (1982) extracts the observed information matrix in terms of the conditional moments of the gradient and curvature of the complete-data log likelihood function introduced within the EM framework. These conditional moments are generally easier to work out than the corresponding derivatives of the incomplete-data log likelihood function. An alternative approach is to numerically differentiate the likelihood function to obtain the Hessian. In an EM-aided differentiation approach, Meilijson (1989) suggests perturbation of the incomplete-data score vector to compute the observed information matrix. In the SEM algorithm (Meng and Rubin 1991), numerical techniques are used to compute the derivative of the EM operator $\boldsymbol{M}$ to obtain the observed information matrix. The basic idea is to use the fact that the rate of convergence is governed by the fraction of the missing information to find the increased variability due to missing information to add to the assessed complete-data covariance matrix. More specifically, let $\boldsymbol{V}$ denote the asymptotic covariance matrix of the MLE $\hat{\boldsymbol{\Psi}}$. Meng and Rubin (1991) show that

$$\boldsymbol{I}^{-1}(\hat{\boldsymbol{\Psi}};\, \boldsymbol{y}) = \mathcal{I}_c^{-1}(\hat{\boldsymbol{\Psi}};\, \boldsymbol{y}) + \Delta \boldsymbol{V}, \tag{6.32}$$

where $\Delta \boldsymbol{V} = \{\boldsymbol{I}_d - \boldsymbol{J}(\hat{\boldsymbol{\Psi}})\}^{-1}\boldsymbol{J}(\hat{\boldsymbol{\Psi}})\mathcal{I}_c^{-1}(\hat{\boldsymbol{\Psi}};\, \boldsymbol{y})$ and $\mathcal{I}_c(\hat{\boldsymbol{\Psi}};\, \boldsymbol{y})$ is the conditional expected complete-data information matrix, and where $\boldsymbol{I}_d$ denotes the $d \times d$ identity matrix. Thus the diagonal elements of $\Delta \boldsymbol{V}$ give the increases in the asymptotic variances of the components of $\hat{\boldsymbol{\Psi}}$ due to missing data. For a wide class of problems where the complete-data density is from the regular exponential family, the evaluation of $\mathcal{I}_c(\hat{\boldsymbol{\Psi}};\, \boldsymbol{y})$ is readily facilitated by standard complete-data computations (McLachlan and Krishnan 2008, Sect. 4.5). The calculation of $\boldsymbol{J}(\hat{\boldsymbol{\Psi}})$ can be readily obtained by using only EM code via numerically differentiation of $\boldsymbol{M}(\boldsymbol{\Psi})$. Let $\hat{\boldsymbol{\Psi}} = \boldsymbol{\Psi}^{(k+1)}$ where the sequence of EM iterates has been stopped according to a suitable stopping rule. Let M_i be the ith component of $\boldsymbol{M}(\boldsymbol{\Psi})$. Let $\boldsymbol{u}^{(j)}$ be a column d-vector with the jth coordinate 1 and others 0. With a possibly different EM sequence $\boldsymbol{\Psi}^{(k)}$, let r_{ij} be the (i, j)th element of $\boldsymbol{J}(\hat{\boldsymbol{\Psi}})$, we have

$$r_{ij}^{(k)} = \frac{M_i[\hat{\boldsymbol{\Psi}} + (\boldsymbol{\Psi}_j^{(k)} - \hat{\boldsymbol{\Psi}}_j \boldsymbol{u}^{(j)})] - \hat{\boldsymbol{\Psi}}_i}{\boldsymbol{\Psi}_j^{(k)} - \hat{\boldsymbol{\Psi}}_j}.$$

Use a suitable stopping rule like $|r_{ij}^{(k+1)} - r_{ij}^{(k)}| < \sqrt{\epsilon}$ to stop each of the sequences r_{ij} $(i, j = 1, 2, \ldots, d)$ and take $r_{ij}^{*} = r_{ij}^{(k+1)}$; see McLachlan and Krishnan (2008, Sect. 4.5).

It is important to emphasize that estimates of the covariance matrix of the MLE based on the expected or observed information matrices are guaranteed to be valid inferentially only asymptotically. In particular for mixture models, it is well known that the sample size n has to be very large before the asymptotic theory of maximum likelihood applies. A resampling approach, the bootstrap (Efron 1979; Efron and Tibshirani 1993), has been considered to tackle this problem; see also Chernick

(2008) for recent developments of the bootstrap in statistics. Basford et al. (1997) compared the bootstrap and information-based approaches for some normal mixture models and found that unless the sample size was very large, the standard errors obtained by an information-based approach were too unstable to be recommended.

The bootstrap is a powerful technique that permits the variability in a random quantity to be assessed using just the data at hand. Standard error estimation of $\hat{\boldsymbol{\Psi}}$ may be implemented according to the bootstrap as follows. Further discussion on bootstrap and resampling methods can be found in Chaps. III.17 and III.18 of this handbook.

1. A new set of data, $\boldsymbol{y}^*$, called the bootstrap sample, is generated according to $\hat{F}$, an estimate of the distribution function of $\boldsymbol{Y}$ formed from the original observed data $\boldsymbol{y}$. That is, in the case where $\boldsymbol{y}$ contains the observed values of a random sample of size n, $\boldsymbol{y}^*$ consists of the observed values of the random sample

$$\boldsymbol{Y}_1^*, \ldots, \boldsymbol{Y}_n^* \overset{\text{i.i.d.}}{\sim} \hat{F},$$

   where the estimate $\hat{F}$ (now denoting the distribution function of a single observation $\boldsymbol{Y}_j$) is held fixed at its observed value.
2. The EM algorithm is applied to the bootstrap observed data $\boldsymbol{y}^*$ to compute the MLE for this data set, $\hat{\boldsymbol{\Psi}}^*$.
3. The bootstrap covariance matrix of $\hat{\boldsymbol{\Psi}}^*$ is given by

$$\text{Cov}^*(\hat{\boldsymbol{\Psi}}^*) = E^*[\{\hat{\boldsymbol{\Psi}}^* - E^*(\hat{\boldsymbol{\Psi}}^*)\}\{\hat{\boldsymbol{\Psi}}^* - E^*(\hat{\boldsymbol{\Psi}}^*)\}^\top], \tag{6.33}$$

   where E^* denotes expectation over the bootstrap distribution specified by $\hat{F}$.

The bootstrap covariance matrix can be approximated by Monte Carlo methods. Steps 1 and 2 are repeated independently a number of times (say, B) to give B independent realizations of $\hat{\boldsymbol{\Psi}}^*$, denoted by $\hat{\boldsymbol{\Psi}}^*_1, \ldots, \hat{\boldsymbol{\Psi}}^*_B$. Then (6.33) can be approximated by the sample covariance matrix of these B bootstrap replications to give

$$\text{Cov}^*(\hat{\boldsymbol{\Psi}}^*) \approx \sum_{b=1}^{B} (\hat{\boldsymbol{\Psi}}^*_b - \overline{\hat{\boldsymbol{\Psi}}}^*)(\hat{\boldsymbol{\Psi}}^*_b - \overline{\hat{\boldsymbol{\Psi}}}^*)^\top / (B-1), \tag{6.34}$$

where $\overline{\hat{\boldsymbol{\Psi}}}^* = \sum_{b=1}^{B} \hat{\boldsymbol{\Psi}}^*_b / B$. The standard error of the ith element of $\hat{\boldsymbol{\Psi}}$ can be estimated by the positive square root of the ith diagonal element of (6.34). It has been shown that 50 to 100 bootstrap replications are generally sufficient for standard error estimation (Efron and Tibshirani 1993).

In Step 1 above, the nonparametric version of the bootstrap would take $\hat{F}$ to be the empirical distribution function formed from the observed data $\boldsymbol{y}$. Situations where we may wish to use the latter include problems where the observed data are censored or are missing in the conventional sense.

6.4 Variations on the EM Algorithm

In this section, further modifications and extensions to the EM algorithm are considered. In general, there are extensions of the EM algorithm:

1. To produce standard errors of the MLE using the EM.
2. To surmount problems of difficult E-step and/or M-step computations.
3. To tackle problems of slow convergence.
4. In the direction of Bayesian or regularized or penalized ML estimations.

We have already discussed methods like the SEM algorithm for producing standard errors of EM-computed MLE in Sect. 6.3.5. The modification of the EM algorithm for Bayesian inference will be discussed in Sect. 6.5.1. In this section, we shall focus on the problems of complicated E- or M-steps and of slow convergence of the EM algorithm.

6.4.1 Complicated E-Step

In some applications of the EM algorithm, the E-step is complex and does not admit a close-form solution to the Q-function. In this case, the E-step at the $(k+1)$th iteration may be executed by a Monte Carlo (MC) process:

1. Make M independent draws of the missing values $\boldsymbol{Z}, \boldsymbol{z}^{(1_k)}, \ldots, \boldsymbol{z}^{(M_k)}$, from the conditional distribution $k(\boldsymbol{z}|\boldsymbol{y}; \boldsymbol{\Psi}^{(k)})$.
2. Approximate the Q-function as

$$Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)}) \approx Q_M(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)}) = \frac{1}{M} \sum_{m=1}^{M} \log k(\boldsymbol{\Psi}|\boldsymbol{z}^{(m_k)}; \boldsymbol{y}).$$

In the M-step, the Q-function is maximized over $\boldsymbol{\Psi}$ to obtain $\boldsymbol{\Psi}^{(k+1)}$. The variant is known as the Monte Carlo EM (MCEM) algorithm (Wei and Tanner 1990). As MC error is introduced at the E-step, the monotonicity property is lost. But in certain cases, the algorithm gets close to a maximizer with a high probability (Booth and Hobert 1999). The problems of specifying M and monitoring convergence are of central importance in the routine use of the algorithm (Levine and Fan 2004). Wei and Tanner (1990) recommend small values of M be used in initial stages and be increased as the algorithm moves closer to convergence. As to monitoring convergence, they recommend that the values of $\boldsymbol{\Psi}^{(k)}$ be plotted against k and when convergence is indicated by the stabilization of the process with random fluctuations about $\hat{\boldsymbol{\Psi}}$, the process may be terminated or continued with a larger value of M. Alternative schemes for specifying M and stopping rule are considered by Booth and Hobert (1999) and McCulloch (1997). The computation of standard errors with MCEM algorithm is discussed in Robert and Casella (2004, Sect. 5.3).

Example 4: Generalized Linear Mixed Models

Generalized linear mixed models (GLMM) are extensions of generalized linear models (GLM) (McCullagh and Nelder 1989) that incorporate random effects in the linear predictor of the GLM (more material on the GLM can be found in Chap. III.24). We let $\boldsymbol{y} = (y_1, \ldots, y_n)^\top$ denote the observed data vector. Conditional on the unobservable random effects vector, $\boldsymbol{u} = (u_1, \ldots, u_q)^\top$, we assume that $\boldsymbol{y}$ arise from a GLM. The conditional mean $\mu_j = E(y_j|\boldsymbol{u})$ is related to the linear predictor $\eta_j = \boldsymbol{x}_j^\top \boldsymbol{\beta} + \boldsymbol{z}_j^\top \boldsymbol{u}$ by the link function $g(\mu_j) = \eta_j$ ($j = 1, \ldots, n$), where $\boldsymbol{\beta}$ is a p-vector of fixed effects and $\boldsymbol{x}_j$ and $\boldsymbol{z}_j$ are, respectively, p-vector and q-vector of explanatory variables associated with the fixed and random effects. This formulation encompasses the modeling of data involving multiple sources of random error, such as repeated measures within subjects and clustered data collected from some experimental units (Breslow and Clayton 1993; Ng et al. 2004).

We let the distribution for $\boldsymbol{u}$ be $g(\boldsymbol{u}; \boldsymbol{D})$ that depends on parameters $\boldsymbol{D}$. The observed data $\boldsymbol{y}$ are conditionally independent with density functions of the form

$$f(y_j|\boldsymbol{u};\, \boldsymbol{\beta}, \kappa) = \exp[m_j \kappa^{-1}\{\theta_j y_j - b(\theta_j)\} + c(y_j;\, \kappa)], \tag{6.35}$$

where θ_j is the canonical parameter, κ is the dispersion parameter, and m_j is the known prior weight. The conditional mean and canonical parameters are related through the equation $\mu_j = b'(\theta_j)$, where the prime denotes differentiation with respect to θ_j. Let $\boldsymbol{\Psi}$ denotes the vector of unknown parameters within $\boldsymbol{\beta}$, κ, and $\boldsymbol{D}$. The likelihood function for $\boldsymbol{\Psi}$ is given by

$$L(\boldsymbol{\Psi}) = \int \prod_{j=1}^{n} f(y_j|\boldsymbol{u};\, \boldsymbol{\beta}, \kappa) g(\boldsymbol{u}; \boldsymbol{D}) d\boldsymbol{u}, \tag{6.36}$$

which cannot usually be evaluated in closed form and has an intractable integral whose dimension depends on the structure of the random effects.

Within the EM framework, the random effects are considered as missing data. The complete data is then $\boldsymbol{x} = (\boldsymbol{y}^\top, \boldsymbol{u}^\top)^\top$ and the complete-data log likelihood is given by

$$\log L_c(\boldsymbol{\Psi}) = \sum_{j=1}^{n} \log f(y_j|\boldsymbol{u};\, \boldsymbol{\beta}, \kappa) + \log g(\boldsymbol{u}; \boldsymbol{D}). \tag{6.37}$$

On the $(k+1)$th iteration of the EM algorithm, the E-step involves the computation of the Q-function, $Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)}) = E_{\boldsymbol{\Psi}^{(k)}}\{\log L_c(\boldsymbol{\Psi})|\boldsymbol{y}\}$, where the expectation is with respect to the conditional distribution of $\boldsymbol{u}|\boldsymbol{y}$ with current parameter value $\boldsymbol{\Psi}^{(k)}$. As this conditional distribution involves the (marginal) likelihood function $L(\boldsymbol{\Psi})$ given in (6.36), an analytical evaluation of the Q-function for the model (6.35) will be impossible outside the normal theory mixed model (Booth and Hobert 1999). The MCEM algorithm can be adopted to tackle this problem by replacing the expectation

in the E-step with a MC approximation. Let $\boldsymbol{u}^{(1_k)}, \ldots, \boldsymbol{u}^{(M_k)}$ denote a random sample from $k(\boldsymbol{u}|\boldsymbol{y}; \boldsymbol{\Psi}^{(k)})$ at the $(k+1)$th iteration. A MC approximation of the Q-function is given by

$$Q_M(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)}) = \frac{1}{M} \sum_{m=1}^{M} \{\log f(\boldsymbol{y}|\boldsymbol{u}^{(m_k)}; \boldsymbol{\beta}, \kappa) + \log g(\boldsymbol{u}^{(m_k)}; \boldsymbol{D})\}. \quad (6.38)$$

From (6.38), it can be seen that the first term of the approximated Q-function involves only parameters $\boldsymbol{\beta}$ and κ, while the second term involves only $\boldsymbol{D}$. Thus, the maximization in the MC M-step is usually relatively simple within the GLMM context (McCulloch 1997).

Alternative simulation schemes for $\boldsymbol{u}$ can be used for (6.38). For example, Booth and Hobert (1999) proposed the rejection sampling and a multivariate t importance sampling approximations. McCulloch (1997) considered dependent MC samples using MC Newton-Raphson (MCNR) algorithm. A two-slice EM algorithm has developed by Vaida and Meng (2005) to handle GLMM with binary response, where the MC E-step is implemented via a slice sampler.

6.4.2 Complicated M-Step

One of major reasons for the popularity of the EM algorithm is that the M-step involves only complete-data ML estimation, which is often computationally simple. But if the complete-data ML estimation is rather complicated, then the EM algorithm is less attractive. In many cases, however, complete-data ML estimation is relatively simple if maximization process on the M-step is undertaken conditional on some functions of the parameters under estimation. To this end, Meng and Rubin (1993) introduce a class of GEM algorithms, which they call the Expectation–Conditional Maximization (ECM) algorithm.

ECM and Multicycle ECM Algorithms

The ECM algorithm takes advantage of the simplicity of complete-data conditional maximization by replacing a complicated M-step of the EM algorithm with several computationally simpler conditional maximization (CM) steps. Each of these CM-steps maximizes the Q-function found in the preceding E-step subject to constraints on $\boldsymbol{\Psi}$, where the collection of all constraints is such that the maximization is over the full parameter space of $\boldsymbol{\Psi}$.

A CM-step might be in closed form or it might itself require iteration, but because the CM maximizations are over smaller dimensional spaces, often they are simpler, faster, and more stable than the corresponding full maximizations called for on the M-step of the EM algorithm, especially when iteration is required. The ECM algorithm typically converges more slowly than the EM in terms of number of

iterations, but can be faster in total computer time. More importantly, the ECM algorithm preserves the appealing convergence properties of the EM algorithm, such as its monotone convergence.

We suppose that the M-step is replaced by $S > 1$ steps and let $\boldsymbol{\Psi}^{(k+s/S)}$ denote the value of $\boldsymbol{\Psi}$ on the sth CM-step of the $(k+1)$th iteration. In many applications of the ECM algorithm, the S CM-steps correspond to the situation where the parameter vector $\boldsymbol{\Psi}$ is partitioned into S subvectors,

$$\boldsymbol{\Psi} = (\boldsymbol{\Psi}_1^\top, \ldots, \boldsymbol{\Psi}_S^\top)^\top.$$

The sth CM-step then requires the maximization of the Q-function with respect to the sth subvector $\boldsymbol{\Psi}_s$ with the other $(S-1)$ subvectors held fixed at their current values. The convergence properties and the rate of convergence of the ECM algorithm have been discussed in Meng (1994), Meng and Rubin (1993), and Sexton and Swensen (2000); see also the discussion in McLachlan and Krishnan (2008, Sect. 5.2.3), where the link to the monotone convergence of Iterative Proportional Fitting with complete data (Bishop et al. 2007, Chap. 3) is described.

It can be shown that

$$Q(\boldsymbol{\Psi}^{(k+1)}; \boldsymbol{\Psi}^{(k)}) \geq Q(\boldsymbol{\Psi}^{(k+(S-1)/S)}; \boldsymbol{\Psi}^{(k)}) \geq \cdots \geq Q(\boldsymbol{\Psi}^{(k)}; \Psi^{(k)}), \qquad (6.39)$$

which implies that the ECM algorithm is a GEM algorithm and so possesses its desirable convergence properties. As noted in Sect. 6.2.3, the inequality (6.39) is a sufficient condition for

$$L(\boldsymbol{\Psi}^{(k+1)}) \geq L(\boldsymbol{\Psi}^{(k)})$$

to hold. In many cases, the computation of an E-step may be much cheaper than the computation of the CM-steps. Hence one might wish to perform one E-step before each CM-step. A cycle is defined to be one E-step followed by one CM-step. The corresponding algorithm is called the multicycle ECM (Meng and Rubin 1993). A multicycle ECM may not necessarily be a GEM algorithm; that is, the inequality (6.39) may not be hold. However, it is not difficult to show that the multicycle ECM algorithm monotonically increases the likelihood function $L(\boldsymbol{\Psi})$ after each cycle, and hence, after each iteration. The convergence results of the ECM algorithm apply to a multicycle version of it. An obvious disadvantage of using a multicycle ECM algorithm is the extra computation at each iteration. Intuitively, as a tradeoff, one might expect it to result in larger increases in the log likelihood function per iteration since the Q-function is being updated more often (Meng 1994; Meng and Rubin 1993).

Example 5: Mixture-of-Experts Models for Multiclass Classification

It is reported in the literature that ME networks trained by the EM algorithm using the IRLS algorithm in the inner loop of the M-step often performed poorly in multiclass classification because of the incorrect independence assumption (Chen

et al. 1999); see also the discussion in Sect. 6.3.3. In this section, we present an ECM algorithm to train ME networks for multiclass classification such that the parameters in the gating and expert networks are separable. It follows that the independence assumption is not required and the parameters in both (6.26) and (6.28) can be updated separately; see, for example, Ng and McLachlan (2004a) and Ng et al. (2006a).

For multiclass classification, the densities $f_h(\boldsymbol{y}_j|\boldsymbol{x}_j;\boldsymbol{\theta}_h)$ $(h = 1,\ldots,m)$ are modelled by a multinomial distribution consisting of one draw on multiple (say, g) categories. That is, we have

$$f_h(\boldsymbol{y}_j|\boldsymbol{x}_j,\boldsymbol{\theta}_h) = \prod_{i=1}^{g-1}\left(\frac{\exp(\boldsymbol{w}_{hi}^\top \boldsymbol{x}_j)}{1+\sum_{r=1}^{g-1}\exp(\boldsymbol{w}_{hr}^\top \boldsymbol{x}_j)}\right)^{y_{ij}}\left(\frac{1}{1+\sum_{r=1}^{g-1}\exp(\boldsymbol{w}_{hr}^\top \boldsymbol{x}_j)}\right)^{y_{gj}}, \tag{6.40}$$

where $\boldsymbol{\theta}_h$ contains the elements in $\boldsymbol{w}_{hi}$ $(i = 1,\ldots,g-1)$. Equation (6.28) in Sect. 6.3.3 thus becomes

$$\sum_{j=1}^{n}\tau_{hj}^{(k)}\left(y_{ij} - \frac{\exp(\boldsymbol{w}_{hi}^\top \boldsymbol{x}_j)}{1+\sum_{r=1}^{g-1}\exp(\boldsymbol{w}_{hr}^\top \boldsymbol{x}_j)}\right)\boldsymbol{x}_j = \boldsymbol{0} \qquad (i = 1,\ldots,g-1) \tag{6.41}$$

for $h = 1,\ldots,m$, which are m sets of non-linear equations each with $(g-1)p$ unknown parameters.

With the ECM algorithm, the M-step is replaced by several computationally simpler CM-steps. For example, the parameter vector $\boldsymbol{\alpha}$ is partitioned as $(\boldsymbol{v}_1^\top,\ldots,\boldsymbol{v}_{m-1}^\top)^\top$. On the $(k+1)$th iteration of the ECM algorithm, the E-step is the same as given in Equations (6.23)–(6.25) for the EM algorithm, but the M-step of the latter is replaced by $(m-1)$ CM-steps, as follows:

- *CM-step 1:* Calculate $\boldsymbol{v}_1^{(k+1)}$ by maximizing Q_α with $\boldsymbol{v}_l$ $(l = 2,\ldots,m-1)$ fixed at $\boldsymbol{v}_l^{(k)}$.
- *CM-step 2:* Calculate $\boldsymbol{v}_2^{(k+1)}$ by maximizing Q_α with $\boldsymbol{v}_1$ fixed at $\boldsymbol{v}_1^{(k+1)}$ and $\boldsymbol{v}_l$ $(l = 3,\ldots,m-1)$ fixed at $\boldsymbol{v}_l^{(k)}$.
- $\vdots$
- *CM-step* $(\boldsymbol{m-1})$ **:** Calculate $\boldsymbol{v}_{(m-1)}^{(k+1)}$ by maximizing Q_α with $\boldsymbol{v}_l$ $(l = 1,\ldots,m-2)$ fixed at $\boldsymbol{v}_l^{(k+1)}$.

As each CM-step above corresponds to a separable set of the parameters in $\boldsymbol{v}_h$ for $h = 1,\ldots,m-1$, it can be obtained using the IRLS approach; see Ng and McLachlan (2004a).

6.4.3 *Speeding Up Convergence*

Several suggestions are available in the literature for speeding up convergence, some of a general kind and some problem-specific; see for example McLachlan

and Krishnan (2008, Chap. 4). Most of them are based on standard numerical analytic methods and suggest a hybrid of EM with methods based on Aitken acceleration, over-relaxation, line searches, Newton methods, conjugate gradients, etc. Unfortunately, the general behaviour of these hybrids is not always clear and they may not yield monotonic increases in the log likelihood over iterations. There are also methods that approach the problem of speeding up convergence in terms of "efficient" data augmentation scheme (Meng and van Dyk 1997). Since the convergence rate of the EM algorithm increases with the proportion of observed information in the prescribed EM framework (Sect. 6.2.4), the basic idea of the scheme is to search for an efficient way of augmenting the observed data. By efficient, they mean less augmentation of the observed data (greater speed of convergence) while maintaining the simplicity and stability of the EM algorithm. A common trade-off is that the resulting E- and/or M-steps may be made appreciably more difficult to implement. To this end, Meng and van Dyk (1997) introduce a working parameter in their specification of the complete data to index a class of possible schemes to facilitate the search.

ECME, AECM, and PX–EM Algorithms

Liu and Rubin (1994, 1998) present an extension of the ECM algorithm called the ECME (expectation–conditional maximization either) algorithm. Here the "either" refers to the fact that with this extension, each CM-step either maximizes the Q-function or the actual (incomplete-data) log likelihood function $\log L(\boldsymbol{\Psi})$, subject to the same constraints on $\boldsymbol{\Psi}$. The latter choice should lead to faster convergence as no augmentation is involved. Typically, the ECME algorithm is more tedious to code than the ECM algorithm, but the reward of faster convergence is often worthwhile especially because it allows convergence to be more easily assessed.

A further extension of the EM algorithm, called the Space-Alternating Generalized EM (SAGE), has been proposed by Fessler and Hero (1994), where they update sequentially small subsets of parameters using appropriately smaller complete data spaces. This approach is eminently suitable for situations like image reconstruction where the parameters are large in number. Meng and van Dyk (1997) combined the ECME and SAGE algorithms. The so-called Alternating ECM (AECM) algorithm allows the data augmentation scheme to vary where necessary over the CM-steps, within and between iterations. With this flexible data augmentation and model reduction schemes, the amount of data augmentation decreases and hence efficient computations are achieved.

In contrast to the AECM algorithm where the optimal value of the working parameter is determined before EM iterations, a variant is considered by Liu et al. (1998) which maximizes the complete-data log likelihood as a function of the working parameter within each EM iteration. The so-called parameter-expanded EM (PX–EM) algorithm has been used for fast stable computation of MLE in a wide range of models (Little and Rubin 2002). This variant has been further developed, known as the one-step-late PX–EM algorithm, to compute maximum *a posteriori*

(MAP) or maximum penalized likelihood (MPL) estimates (van Dyk and Tang 2003). Analogous convergence results hold for the ECME, AECM, and PX–EM algorithms as for the EM and ECM algorithms. More importantly, these algorithms preserve the monotone convergence of the EM algorithm.

Incremental Scheme of the EM Algorithm

The EM algorithm can be viewed as alternating minimization of a joint function between a parameter space $\boldsymbol{\Omega}$ and a family of distributions $\boldsymbol{\Phi}$ over the unobserved variables (Csiszár and Tusnády 1984; Hathaway 1986). Let $\boldsymbol{z}$ denote the vector containing the unobservable data and let P be any distribution defined over the support of $\boldsymbol{Z}$. The joint function is defined as

$$D(P, \boldsymbol{\Psi}) = -\log L(\boldsymbol{\Psi}) + KL[P, g(\boldsymbol{z}|\boldsymbol{y}; \boldsymbol{\Psi})], \tag{6.42}$$

where $g(\boldsymbol{z}|\boldsymbol{y}; \boldsymbol{\Psi})$ is the conditional distribution of $\boldsymbol{Z}$ given the observed data and $KL[P, g(\boldsymbol{z}|\boldsymbol{y}; \boldsymbol{\Psi})]$ is the Kullback-Leibler information that measures the divergence of P relative to $g(\boldsymbol{z}|\boldsymbol{y}; \boldsymbol{\Psi})$. Hathaway (1986) shows that, given the current estimates $\boldsymbol{\Psi}^{(k)}$, the E-step on the $(k+1)$th scan corresponds to the minimization of (6.42) with respect to P over $\boldsymbol{\Phi}$. For fixed $P^{(k+1)}$, the M-step then minimizes (6.42) with respect to $\boldsymbol{\Psi}$ over $\boldsymbol{\Omega}$.

From this perspective, Neal and Hinton (1998) justify an incremental variant of the EM algorithm in which only a block of unobserved data is calculated in each E-step at a time before performing a M-step. A scan of the incremental EM (IEM) algorithm thus consists of B "partial" E-steps and B M-steps, where B is the total number of blocks of data. This variant of the EM algorithm has been shown empirically to give faster convergence compared to the EM algorithm in applications where the M-step is computationally simple, for example, in fitting multivariate normal mixtures (Ng and McLachlan 2003, 2004b). With the IEM algorithm, Neal and Hinton (1998) showed that the partial E-step and the M-step both monotonically increase $F(P, \boldsymbol{\Psi}) = -D(P, \boldsymbol{\Psi})$ and if a local maximum (or saddle point) of $F(P, \boldsymbol{\Psi})$ occurs at P^* and $\boldsymbol{\Psi}^*$, then a local maximum (or saddle point) of the log likelihood occurs at $\boldsymbol{\Psi}^*$ as well. Although the IEM algorithm can possess stable convergence to stationary points in the log likelihood under slightly stronger conditions of Wu (1983) for the EM algorithm, the current theoretical results for the IEM algorithm do not quarantine monotonic behaviour of the log likelihood as the EM algorithm does. The same argument for proving that the EM algorithm always increases the log likelihood cannot be adopted here, as the estimate of $\boldsymbol{\Psi}$ in $Q(\boldsymbol{\Psi}; \boldsymbol{\Psi}^{(k)})$ of (6.7) is changing at each iteration within each scan (Ng and McLachlan 2003). However, it is noted that $F(P, \boldsymbol{\Psi})$ can be considered as a lower bound on the log likelihood since the Kullback-Leibler information is non-negative. For given P, as obtained in the partial E-step, the M-step increases $F(P, \boldsymbol{\Psi})$ with respect to $\boldsymbol{\Psi}$. It follows that

$$F(P, \boldsymbol{\Psi}^{(k+(b+1)/B)}) \geq F(P, \boldsymbol{\Psi}^{(k+b/B)}) \qquad (b = 0, \ldots, B\text{-}1).$$

That is, the lower bound of the log likelihood is monotonic increasing after each iteration.

The argument for improved rate of convergence is that the IEM algorithm exploits new information more quickly rather than waiting for a complete scan of the data before parameters are updated by an M-step. Another method suggested by Neal and Hinton (1998) is the sparse EM (SPEM) algorithm. In fitting a mixture model to a data set by ML via the EM, the current estimates of some posterior probabilities $\tau_{ij}^{(k)}$ for a given data point $\boldsymbol{y}_j$ are often close to zero. For example, if $\tau_{ij}^{(k)} < 0.005$ for the first two components of a four-component mixture being fitted, then with the SPEM algorithm we would fix $\tau_{ij}^{(k)}$ (i=1,2) for membership of $\boldsymbol{y}_j$ with respect to the first two components at their current values and only update $\tau_{ij}^{(k)}$ (i=3,4) for the last two components. This sparse E-step will take time proportional to the number of components that needed to be updated. A sparse version of the IEM algorithm (SPIEM) can be formulated by combining the partial E-step and the sparse E-step. With these versions, the likelihood is still found to be increased after each scan. Ng and McLachlan (2003) study the relative performances of these algorithms with various number of blocks B for the fitting of normal mixtures. They propose to choose B to be that factor of n that is the closest to $B^* = \text{round}(n^{2/5})$ for unrestricted component-covariance matrices, where round(r) rounds r to the nearest integer.

Ng and McLachlan (2004b) propose to speed up further the IEM and SPIEM algorithms for the fitting of normal mixtures by imposing a multiresolution kd-tree (mrkd-tree) structure in performing the E-step. Here kd stands for k-dimensional where, in our notation, $k = p$, the dimension of an observation $\boldsymbol{y}_j$. The mrkd-tree is a binary tree that recursively splits the whole set of data points into partition (Moore 1999). The contribution of all the data points in a tree node to the sufficient statistics is simplified by calculating at the mean of these data points to save time. The mrkd-tree approach does not guarantee the desirable reliable convergence properties of the EM algorithm. However, the IEM-based mrkd-tree algorithms have been shown empirically to give a monotonic convergence as reliable as the EM algorithm when the size of leaf nodes are sufficiently small (Ng and McLachlan 2004b). It is noted that the number of leaf nodes will increase dramatically when the dimension of the data points p increases. This implies that mrkd-trees-based algorithms will not be able to speed up the EM algorithm for applications to high dimensional data (Ng and McLachlan 2004b). Recently, a number of techniques have been developed to reduce dimensionality without losing significant information and separability among mixture components; see, for example, the matrix factorization approach of Nikulin and McLachlan (2010) and the references therein.

6.5 Miscellaneous Topics on the EM Algorithm

6.5.1 EM Algorithm for MAP Estimation

Although we have focussed on the application of the EM algorithm for computing MLEs in a frequentist framework, it can be equally applied to find the mode of the posterior distribution in a Bayesian framework. This problem is analogous to MLE and hence the EM algorithm and its variants can be adapted to compute maximum *a posteriori* (MAP) estimates. The computation of the MAP estimate in a Bayesian framework via the EM algorithm corresponds to the consideration of some prior density for $\boldsymbol{\Psi}$. The E-step is effectively the same as for the computation of the MLE of $\boldsymbol{\Psi}$ in a frequentist framework, requiring the calculation of the Q-function. The M-step differs in that the objective function for the maximization process is equal to the Q-function, augmented by the log prior density. The combination of prior and sample information provides a posterior distribution of the parameter on which the estimation is based.

The advent of inexpensive high speed computers and the simultaneous rapid development in posterior simulation techniques such as Markov chain Monte Carlo (MCMC) methods (Gelfand and Smith 1990) enable Bayesian estimation to be undertaken. In particular, posterior quantities of interest can be approximated through the use of MCMC methods such as the Gibbs sampler. Such methods allow the construction of an ergodic Markov chain with stationary distribution equal to the posterior distribution of the parameter of interest. A concise theoretical treatment of MCMC is provided in Gamerman and Lopes (2006) and Robert and Casella (2004); see also McLachlan and Krishnan (2008, Chap. 8) and the references therein. A detailed description of the MCMC technology can also be found in Chap. II.4.

Although the application of MCMC methods is now routine, there are some difficulties that have to be addressed with the Bayesian approach, particularly in the context of mixture models. One main hindrance is that improper priors yield improper posterior distributions. Another hindrance is that when the number of components g is unknown, the parameter space is simultaneously ill-defined and of infinite dimension. This prevents the use of classical testing procedures and priors (McLachlan and Peel 2000, Chap. 4).

6.5.2 Iterative Simulation Algorithms

In computing Bayesian solutions to incomplete-data problems, iterative simulation techniques have been adopted to find the MAP estimates or estimating the entire posterior density. These iterative simulation techniques are conceptually similar to the EM algorithm, simply replacing the E- and M-steps by draws from the current conditional distribution of the missing data and $\boldsymbol{\Psi}$, respectively. However, in some methods such as the MCEM algorithm described in Sect. 6.4.1, only the E-step is

so implemented. Many of these methods can be interpreted as iterative simulation analogs of the various versions of the EM and its extensions. Some examples are Stochastic EM, Data Augmentation algorithm, and MCMC methods such as the Gibbs sampler (McLachlan and Krishnan 2008, Chap. 6). Here, we give a very brief outline of the Gibbs sampler; see also Chap. II.4 of this handbook and the references therein.

The Gibbs sampler is extensively used in many Bayesian problems where the joint distribution is too complicated to handle, but the conditional distributions are often easy enough to draw from; see Casella and George (1992). On the Gibbs sampler, an approximate sample from $p(\boldsymbol{\Psi} \mid \boldsymbol{y})$ is obtained by simulating directly from the (full) conditional distribution of a subvector of $\boldsymbol{\Psi}$ given all the other parameters in $\boldsymbol{\Psi}$ and $\boldsymbol{y}$. We write $\boldsymbol{\Psi} = (\boldsymbol{\Psi}_1, \ldots, \boldsymbol{\Psi}_d)$ in component form, a d-dimensional Gibbs sampler makes a Markov transition from $\boldsymbol{\Psi}^{(k)}$ to $\boldsymbol{\Psi}^{(k+1)}$ via d successive simulations as follows:

(1) Draw $\Psi_1^{(k+1)}$ from $p(\Psi_1 \mid \boldsymbol{y}; \Psi_2^{(k)}, \ldots, \Psi_d^{(k)})$.
(2) Draw $\Psi_2^{(k+1)}$ from $p(\Psi_2 \mid \boldsymbol{y}; \Psi_1^{(k+1)}, \Psi_3^{(k)} \ldots, \Psi_d^{(k)})$.
$\vdots$ $\vdots$ $\vdots$
(d) Draw $\Psi_d^{(k+1)}$ from $p(\Psi_d \mid \boldsymbol{y}; \Psi_1^{(k+1)}, \ldots, \Psi_{d-1}^{(k+1)})$.

The vector sequence $\{\boldsymbol{\Psi}^{(k)}\}$ thus generated is known to be a realization of a homogeneous Markov Chain. Many interesting properties of such a Markov sequence have been established, including geometric convergence, as $k \to \infty$; to a unique stationary distribution that is the posterior density $p(\Psi_1^{(k)}, \ldots, \Psi_d^{(k)} \mid \boldsymbol{y})$ under certain conditions; see Roberts and Polson (1994). Among other sampling methods, there is the Metropolis-Hastings algorithm (Hastings 1970), which, in contrast to the Gibbs sampler, accepts the candidate simulated component in $\boldsymbol{\Psi}$ with some defined probability (McLachlan and Peel 2000, Chap. 4).

The Gibbs sampler and other such iterative simulation techniques being Bayesian in their point of view consider both parameters and missing values as random variables and both are subjected to random draw operations. In the iterative algorithms under a frequentist framework, like the EM-type algorithms, parameters are subjected to a maximization operation and missing values are subjected to an averaging operation. Thus the various versions of the Gibbs sampler can be viewed as stochastic analogs of the EM, ECM, and ECME algorithms (Robert and Casella 2004). Besides these connections, the EM-type algorithms also come in useful as starting points for iterative simulation algorithms where typically regions of high density are not known *a priori* (McLachlan and Krishnan 2008, Sect. 6.10). The relationship between the EM algorithm and the Gibbs sampler and the connection between their convergence properties have been examined in Sahu and Roberts (1999).

6.5.3 *Further Applications of the EM Algorithm*

Since the publication of Dempster et al. (1977), the number, variety, and range of applications of the EM algorithm and its extensions have been tremendous. Applications in many different contexts can be found in monographs Little and Rubin (2002), McLachlan et al. (2004), McLachlan and Krishnan (2008), and McLachlan and Peel (2000). We conclude the chapter with a quick summary of some of the more interesting and topical applications of the EM algorithm.

Bioinformatics: EMMIX-GENE and EMMIX-WIRE Procedures

In bioinformatics, much attention is centered on the cluster analysis of the tissue samples and also the genes. The clustering of tumour tisses can play a useful role in the discovery and understanding of new subtypes of diseases (McLachlan et al. 2002), while the clustering of gene expression profiles contributes significantly to the elucidation of unknown gene function, the validation of gene discoveries and the interpretation of biological processes (Ng et al. 2006b). The EM algorithm and its variants have been applied to tackle some of the problems arisen in such applications. For example, the clustering of tumour tissues on the basis of genes expression is a nonstandard cluster analysis problem since the dimension of each tissue sample is so much greater than the number of tissues. The EMMIX-GENE procedure of McLachlan et al. (2002) handles the problem of a high-dimensional feature vector by using mixtures of factor analyzers whereby the component correlations between the genes are explained by their conditional linear dependence on a small number of latent or unobservable factors specific to each component. The mixtures of factor analyzers model can be fitted by using the AECM algorithm (Meng and van Dyk 1997); see, for example, McLachlan et al. (2004).

The clustering of gene profiles is also not straightforward as the profiles of the genes are not all independently distributed and the expression levels may have been obtained from an experimental design involving replicated arrays (Lee et al. 2000; Pavlidis et al. 2003). Similarly, in time-course studies (Storey et al. 2005), where expression levels are measured under various conditions or at different time points, gene expressions obtained from the same condition (tissue sample) are correlated. Ng et al. (2006b) have developed a random-effects model that provides a unified approach to the clustering of genes with correlated expression levels measured in a wide variety of experimental situations. The EMMIX-WIRE procedure of Ng et al. (2006b) formulates a linear-mixed-effects model (LMM) for the mixture components in which both gene-specific and tissue-specific random effects are incorporated in the modelling of the microarray data. In their model, the gene profiles are not all independently distributed as genes within the same component in the mixture model are allowed to be dependent due to the presence of the tissue-specific random effects. This problem is circumvented by proceeding conditionally on the tissue-specific random effects, as given these terms, the gene

profiles are all conditionally independent. In this way, Ng et al. (2006b) showed that the unknown parameter vector $\boldsymbol{\Psi}$ can be estimated by ML via the EM algorithm under a conditional mode, where both the E- and M-steps are carried out in closed form.

Health Science: On-Line Prediction of Hospital Resource Utilization

The continuing development and innovative use of information technology in health care has played a significant role in contributing and advancing this active and burgeoning field. Inpatient length of stay (LOS) is an important measure of hospital activity and health care utilization. It is also considered to be a measurement of disease severity and patient acuity (Ng et al. 2006a; Pofahl et al. 1998). Length of stay predictions have therefore important implications in various aspects of health care decision support systems. Most prediction tools use a batch-mode training process. That is, the model is trained only after the entire training set is available. Such a training method is unrealistic in the prediction of LOS as the data become available over time and the input-output pattern of data changes dynamically over time.

An intelligent ME network for on-line prediction of LOS via an incremental ECM algorithm has been proposed by Ng et al. (2006a). The strength of an incremental training process is that it enables the network to be updated when an input-output datum becomes known. These on-line and incremental updating features increase the simulation between neural networks and human decision making capability in terms of learning from "every" experience. In addition, an on-line process is capable of providing an output whenever a new datum becomes available. This on-the-spot information is therefore more useful and practical for adaptive training of model parameters and making decisions (Jepson et al. 2003; Lai and Fang 2005), especially when one deals with a tremendous amount of data.

The incremental training process for on-line prediction is formulated based on the incremental scheme of the EM algorithm described in Sect. 6.4.3; see also Ng and McLachlan (2003) and Ng et al. (2006a). In particular, the unknown parameters are updated in the CM-step when a single input-output datum is available. Also, a discount parameter is introduced to gradually "forget" the effect of previous estimated posterior probabilities obtained from earlier less-accurate estimates (Jordan and Jacobs 1994; Sato and Ishii 2000). It implies that the sufficient statistics required in the CM-step are decayed exponentially with a multiplicative discount factor as the training proceeds. When the discount parameter is scheduled to approach one as the iteration tends to infinity, the updating rules so formed can be considered as a stochastic approximation for obtaining the ML estimators (Sato and Ishii 2000; Titterington 1984).

References

Baker, S.G.: A simple method for computing the observed information matrix when using the EM algorithm with categorical data. J. Comput. Graph. Stat. **1**, 63–76 (1992)

Basford, K.E., Greenway, D.R., McLachlan, G.J., Peel, D.: Standard errors of fitted means under normal mixture models. Comput. Stat. **12**, 1–17 (1997)

Bishop, Y.M.M., Fienberg, S.E., Holland, P.W.: Discrete Multivariate Analysis: Theory and Practice. Springer, New York (2007)

Booth, J.G., Hobert, J.P.: Maximizing generalized linear mixed model likelihoods with an automated Monte Carlo EM algorithm. J. Roy. Stat. Soc. B **61**, 265–285 (1999)

Breslow, N.E., Clayton, D.G.: Approximate inference in generalized linear mixed models. J. Am. Stat. Assoc. **88**, 9–25 (1993)

Casella, G., George, E.I.: Explaining the Gibbs sampler. Am. Stat. **46**, 167–174 (1992)

Chen, K., Xu, L., Chi, H.: Improved learning algorithms for mixture of experts in multiclass classification. Neural Netw. **12**, 1229–1252 (1999)

Chernick, M.R.: Bootstrap Methods: A Guide for Practitioners and Researchers. Wiley, Hoboken, New Jersey (2008)

Cramér, H.: Mathematical Methods of Statistics. Princeton University Press, Princeton, New Jersey (1946)

Csiszár, I., Tusnády, G.: Information geometry and alternating minimization procedure. In: Dudewicz, E.J., Plachky, D., Sen, P.K. (eds.) Recent Results in Estimation Theory and Related Topics, pp. 205–237. R. Oldenbourg, Munich (1984)

Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. J. Roy. Stat. Soc. B **39**, 1–38 (1977)

Efron, B.: Bootstrap methods: another look at the jackknife. Ann. Stat. **7**, 1–26 (1979)

Efron, B., Tibshirani, R.: An Introduction to the Bootstrap. Chapman & Hall, London (1993)

Fessler, J.A., Hero, A.O.: Space-alternating generalized expectation-maximization algorithm. IEEE Trans. Signal. Process. **42**, 2664–2677 (1994)

Flury, B., Zoppé, A.: Exercises in EM. Am. Stat. **54**, 207–209 (2000)

Gamerman, D., Lopes, H.F.: Markov Chain Monte Carlo: Stochastic Simulation for Bayesian Inference, 2nd edn. Chapman & Hall/CRC, Boca Raton, FL (2006)

Gelfand, A.E., Smith, A.F.M.: Sampling-based approaches to calculating marginal densities. J. Am. Stat. Assoc. **85**, 398–409 (1990)

Hathaway, R.J.: Another interpretation of the EM algorithm for mixture distributions. Stat. Probab. Lett. **4**, 53–56 (1986)

Hastings, W.K.: Monte Carlo sampling methods using Markov chains and their applications. Biometrika **57**, 97–109 (1970)

Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E.: Adaptive mixtures of local experts. Neural Comput. **3**, 79–87 (1991)

Jamshidian, M., Jennrich, R.I.: Standard errors for EM estimation. J. Roy. Stat. Soc. B **62**, 257–270 (2000)

Jepson, A.D., Fleet, D.J., EI-Maraghi, T.F.: Robust online appearance models for visual tracking. IEEE Trans. Pattern Anal. Mach. Intell. **25**, 1296–1311 (2003)

Jordan, M.I., Jacobs, R.A.: Hierarchical mixtures of experts and the EM algorithm. Neural Comput. **6**, 181–214 (1994)

Jordan, M.I., Xu, L.: Convergence results for the EM approach to mixtures of experts architectures. Neural Netw. **8**, 1409–1431 (1995)

Lai, S.H., Fang, M.: An adaptive window width/center adjustment system with online training capabilities for MR images. Artif. Intell. Med. **33**, 89–101 (2005)

Lee, M.L.T., Kuo, F.C., Whitmore, G.A., Sklar, J.: Importance of replication in microarray gene expression studies: statistical methods and evidence from repetitive cDNA hybridizations. Proc. Natl. Acad. Sci. USA **97**, 9834–9838 (2000)

Levine, R., Fan, J.J.: An automated (Markov chain) Monte Carlo EM algorithm. J. Stat. Comput. Simulat. **74**, 349–359 (2004)
Little, R.J.A., Rubin, D.B.: Statistical Analysis with Missing Data, 2nd edn. Wiley, New York (2002)
Liu, C., Rubin, D.B.: The ECME algorithm: a simple extension of EM and ECM with faster monotone convergence. Biometrika **81**, 633–648 (1994)
Liu, C., Rubin, D.B.: Maximum likelihood estimation of factor analysis using the ECME algorithm with complete and incomplete data. Stat. Sin. **8**, 729–747 (1998)
Liu, C., Rubin, D.B., Wu, Y.N.: Parameter expansion to accelerate EM: the PX–EM algorithm. Biometrika **85**, 755–770 (1998)
Louis, T.A.: Finding the observed information matrix when using the EM algorithm. J. Roy. Stat. Soc. B **44**, 226–233 (1982)
McCullagh, P.A., Nelder, J.: Generalized Linear Models, 2nd edn. Chapman & Hall, London (1989)
McCulloch, C.E.: Maximum likelihood algorithms for generalized linear mixed models. J. Am. Stat. Assoc. **92**, 162–170 (1997)
McLachlan, G.J., Basford, K.E.: Mixture Models: Inference and Applications to Clustering. Marcel Dekker, New York (1988)
McLachlan, G.J., Bean, R.W., Peel, D.: A mixture model-based approach to the clustering of microarray expression data. Bioinformatics **18**, 413–422 (2002)
McLachlan, G.J., Do, K.A., Ambroise, C.: Analyzing Microarray Gene Expression Data. Wiley, New York (2004)
McLachlan, G.J., Krishnan, T.: The EM Algorithm and Extensions, 2nd edn. Wiley, Hoboken, New Jersey (2008)
McLachlan, G.J., Peel, D.: Finite Mixture Models. Wiley, New York (2000)
Meilijson, I.: A fast improvement of the EM algorithm in its own terms. J. Roy. Stat. Soc. B **51**, 127–138 (1989)
Meng, X.L.: On the rate of convergence of the ECM algorithm. Ann. Stat. **22**, 326–339 (1994)
Meng, X.L., Rubin, D.B.: Using EM to obtain asymptotic variance-covariance matrices: the SEM algorithm. J. Am. Stat. Assoc. **86**, 899–909 (1991)
Meng, X.L., Rubin, D.B.: Maximum likelihood estimation via the ECM algorithm: a general framework. Biometrika **80**, 267–278 (1993)
Meng, X.L., van Dyk, D.: The EM algorithm – an old folk song sung to a fast new tune. J. Roy. Stat. Soc. B **59**, 511–567 (1997)
Moore, A.W.: Very fast EM-based mixture model clustering using multiresolution kd-trees. In: Kearns, M.S., Solla, S.A., Cohn, D.A. (eds.) Advances in Neural Information Processing Systems 11, pp. 543–549. MIT Press, MA (1999)
Neal, R.M., Hinton, G.E.: A view of the EM algorithm that justifies incremental, sparse, and other variants. In: Jordan, M.I. (ed.) Learning in Graphical Models, pp. 355–368. Kluwer, Dordrecht (1998)
Nettleton, D.: Convergence properties of the EM algorithm in constrained parameter spaces. Can. J. Stat. **27**, 639–648 (1999)
Ng, S.K., McLachlan, G.J.: On the choice of the number of blocks with the incremental EM algorithm for the fitting of normal mixtures. Stat. Comput. **13**, 45–55 (2003)
Ng, S.K., McLachlan, G.J (2004a). Using the EM algorithm to train neural networks: misconceptions and a new algorithm for multiclass classification. IEEE Trans. Neural Netw. **15**, 738–749.
Ng, S.K., McLachlan, G.J (2004b). Speeding up the EM algorithm for mixture model-based segmentation of magnetic resonance images. Pattern Recogn. **37**, 1573–1589.
Ng, S.K., McLachlan, G.J., Lee, A.H (2006a). An incremental EM-based learning approach for on-line prediction of hospital resource utilization. Artif. Intell. Med. **36**, 257–267.
Ng, S.K., McLachlan, G.J., Wang, K., Ben-Tovim Jones, L., Ng, S.W (2006b). A mixture model with random-effects components for clustering correlated gene-expression profiles. Bioinformatics **22**, 1745–1752.

Ng, S.K., McLachlan, G.J., Yau, K.K.W., Lee, A.H.: Modelling the distribution of ischaemic stroke-specific survival time using an EM-based mixture approach with random effects adjustment. Stat. Med. **23**, 2729–2744 (2004)

Nikulin, V., McLachlan, G.J.: A gradient-based algorithm for matrix factorization applied to dimensionality reduction. In: Fred, A., Filipe, J., Gamboa, H. (eds.) Proceedings of BIOSTEC 2010, the 3rd International Joint Conference on Biomedical Engineering Systems and Technologies, pp. 147–152. Institute for Systems and Technologies of Information, Control and Communication, Portugal (2010)

Pavlidis, P., Li, Q., Noble, W.S.: The effect of replication on gene expression microarray experiments. Bioinformatics **19**, 1620–1627 (2003)

Pernkopf, F., Bouchaffra, D.: Genetic-based EM algorithm for learning Gaussian mixture models. IEEE Trans. Pattern Anal. Mach. Intell. **27**, 1344–1348 (2005)

Pofahl, W.E., Walczak, S.M., Rhone, E., Izenberg, S.D.: Use of an artificial neural network to predict length of stay in acute pancreatitis. Am. Surg. **64**, 868–872 (1998)

Robert, C.P., Casella, G.: Monte Carlo Statistical Methods, 2nd edn. Springer, New York (2004)

Roberts, G.O., Polson, N.G.: On the geometric convergence of the Gibbs sampler. J. Roy. Stat. Soc. B **56**, 377–384 (1994)

Sahu, S.K., Roberts, G.O.: On convergence of the EM algorithm and the Gibbs sampler. Stat. Comput. **9**, 55–64 (1999)

Sato, M., Ishii, S.: On-line EM algorithm for the normalized Gaussian network. Neural Comput. **12**, 407–432 (2000)

Sexton, J., Swensen, A.R.: ECM algorithms that converge at the rate of EM. Biometrika **87**, 651–662 (2000)

Storey, J.D., Xiao, W., Leek, J.T., Tompkins, R.G., Davis, R.W.: Significance analysis of time course microarray experiments. Proc. Natl. Acad. Sci. USA **102**, 12837–12842 (2005)

Titterington, D.M.: Recursive parameter estimation using incomplete data. J. Roy. Stat. Soc. B **46**, 257–267 (1984)

Ueda, N., Nakano, R.: Deterministic annealing EM algorithm. Neural Netw. **11**, 271–282 (1998)

van Dyk, D.A., Tang, R.: The one-step-late PXEM algorithm. Stat. Comput. **13**, 137–152 (2003)

Vaida, F., Meng, X.L.: Two-slice EM algorithms for fitting generalized linear mixed models with binary response. Stat. Modelling **5**, 229–242 (2005)

Wei, G.C.G., Tanner, M.A.: A Monte Carlo implementation of the EM algorithm and the poor man's data augmentation algorithms. J. Am. Stat. Assoc. **85**, 699–704 (1990)

Wright, K., Kennedy, W.J.: An interval analysis approach to the EM algorithm. J. Comput. Graph. Stat. **9**, 303–318 (2000)

Wu, C.F.J.: On the convergence properties of the EM algorithm. Ann. Stat. **11**, 95–103 (1983)

Chapter 7
Stochastic Optimization

James C. Spall

Stochastic optimization algorithms have been growing rapidly in popularity over the last decade or two, with a number of methods now becoming "industry standard" approaches for solving challenging optimization problems. This chapter provides a synopsis of some of the critical issues associated with stochastic optimization and a gives a summary of several popular algorithms. Much more complete discussions are available in the indicated references.

To help constrain the scope of this article, we restrict our attention to methods using only measurements of the criterion (loss function). Hence, we do not cover the many stochastic methods using information such as gradients of the loss function. Section 7.1 discusses some general issues in stochastic optimization. Section 7.2 discusses random search methods, which are simple and surprisingly powerful in many applications. Section 7.3 discusses stochastic approximation, which is a foundational approach in stochastic optimization. Section 7.4 discusses a popular method that is based on connections to natural evolution – genetic algorithms. Finally, Sect. 7.5 offers some concluding remarks.

7.1 Introduction

7.1.1 General Background

Stochastic optimization plays a significant role in the analysis, design, and operation of modern systems. Methods for stochastic optimization provide a means of coping with inherent system noise and coping with models or systems that are highly

J.C. Spall (✉)
The Johns Hopkins University, Applied Physics Laboratory Laurel, MD, USA
e-mail: James.Spall@jhuapl.edu

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_7,

nonlinear, high dimensional, or otherwise inappropriate for classical deterministic methods of optimization. Stochastic optimization algorithms have broad application to problems in statistics (e.g., design of experiments and response surface modeling), science, engineering, and business. Algorithms that employ some form of stochastic optimization have become widely available. For example, many modern data mining packages include methods such as simulated annealing and genetic algorithms as tools for extracting patterns in data.

Specific applications include business (making short- and long-term investment decisions in order to increase profit), aerospace engineering (running computer simulations to refine the design of a missile or aircraft), medicine (designing laboratory experiments to extract the maximum information about the efficacy of a new drug), and traffic engineering (setting the timing for the signals in a traffic network). There are, of course, *many* other applications.

Let us introduce some concepts and notation. Suppose Θ is the domain of allowable values for a vector $\boldsymbol{\theta}$. The fundamental problem of interest is to find the value(s) of a vector $\boldsymbol{\theta} \in \Theta$ that minimize a scalar-valued *loss function* $L(\boldsymbol{\theta})$. Other common names for L are *performance measure*, *objective function*, *measure-of-effectiveness* (MOE), *fitness function* (or *negative fitness function*), or *criterion*. While this problem refers to *minimizing* a loss function, a maximization problem (e.g., maximizing profit) can be trivially converted to a minimization problem by changing the sign of the criterion. This chapter focuses on the problem of minimization. In some cases (i.e., differentiable L), the minimization problem can be converted to a root-finding problem of finding $\boldsymbol{\theta}$ such that $\boldsymbol{g}(\boldsymbol{\theta}) = \partial L(\boldsymbol{\theta})/\partial \boldsymbol{\theta} = \mathbf{0}$. Of course, this conversion must be done with care because such a root may not correspond to a global minimum of L.

The three remaining subsections in this section define some basic quantities, discuss some contrasts between (classical) deterministic optimization and stochastic optimization, and discuss some basic properties and fundamental limits. This section provides the foundation for interpreting the algorithm presentations in Sects. 7.2 to 7.4. There are many other references that give general reviews of various aspects of stochastic optimization. Among these are Arsham (1998), Fouskakis and Draper (2002), Fu (2002), Gosavi (2003), Michalewicz and Fogel (2000), Spall (2003), and Cochran (2011; see topic area "stochastic optimization").

7.1.2 Formal Problem Statement

The problem of minimizing a loss function $L = L(\boldsymbol{\theta})$can be formally represented as finding the set:

$$\Theta^* \equiv \arg\min_{\boldsymbol{\theta}\in\Theta} L(\boldsymbol{\theta}) = \{\boldsymbol{\theta}^* \in \Theta : L(\boldsymbol{\theta}^*) \leq L(\boldsymbol{\theta}) \text{ for all } \boldsymbol{\theta} \in \Theta\}\,, \qquad (7.1)$$

where $\boldsymbol{\theta}$ is the $p-$dimensional vector of parameters that are being adjusted and $\Theta \subseteq \mathbb{R}^p$. The "arg $\min_{\boldsymbol{\theta} \in \Theta}$" statement in (7.1) should be read as: Θ^* is the set of values $\boldsymbol{\theta} = \boldsymbol{\theta}^*$ ($\boldsymbol{\theta}$ the "argument" in "arg min") that minimize $L(\boldsymbol{\theta})$ subject to $\boldsymbol{\theta}^*$ satisfying the constraints represented in the set Θ. The elements $\boldsymbol{\theta}^* \in \Theta^* \subseteq \Theta$ are equivalent solutions in the sense that they yield identical values of the loss function. The solution set Θ^* in (7.1) may be a unique point, a countable (finite or infinite) collection of points, or a set containing an uncountable number of points.

For ease of exposition, this chapter generally focuses on continuous optimization problems, although some of the methods may also be used in discrete problems. In the continuous case, it is often assumed that L is a "smooth" (perhaps several times differentiable) function of $\boldsymbol{\theta}$. Continuous problems arise frequently in applications such as model fitting (parameter estimation), adaptive control, neural network training, signal processing, and experimental design. Discrete optimization (or *combinatorial optimization*) is a large subject unto itself (resource allocation, network routing, policy planning, etc.).

A major issue in optimization is distinguishing between global and local optima. All other factors being equal, one would always want a globally optimal solution to the optimization problem (i.e., at least one $\boldsymbol{\theta}^*$ in the set of values Θ^*). In practice, however, it may not be feasible to find a global solution and one must be satisfied with obtaining a *local* solution. For example, L may be shaped such that there is a clearly defined minimum point over a broad region of the domain Θ, while there is a very narrow spike at a distant point. If the trough of this spike is lower than any point in the broad region, the local optimal solution is better than any nearby $\boldsymbol{\theta}$, but it is not be the best possible $\boldsymbol{\theta}$.

It is usually only possible to ensure that an algorithm approaches a local minimum with a finite amount of resources being put into the optimization process. That is, it is easy to construct functions that will "fool" any known algorithm, unless the algorithm is given explicit prior information about the location of the global solution — certainly not a case of practical interest! However, since the local minimum may still yield a significantly improved solution (relative to no formal optimization process at all), the local minimum may be a fully acceptable solution for the resources available (human time, money, computer time, etc.) to be spent on the optimization. However, we discuss several algorithms (random search, stochastic approximation, and genetic algorithms) that are *sometimes* able to find global solutions from among multiple local solutions.

7.1.3 Contrast of Stochastic and Deterministic Optimization

As a chapter on *stochastic* optimization, the algorithms considered here apply where:

I. There is random noise in the measurements of $L(\boldsymbol{\theta})$

– and/or –

II. There is a random (Monte Carlo) choice made in the search direction as the algorithm iterates toward a solution.

In contrast, classical deterministic optimization assumes that perfect information is available about the loss function (and derivatives, if relevant) and that this information is used to determine the search direction in a deterministic manner at every step of the algorithm. In many practical problems, such information is not available. We discuss properties I and II below.

Let $\hat{\boldsymbol{\theta}}_k$ be the generic notation for the estimate for $\boldsymbol{\theta}$ at the kth iteration of whatever algorithm is being considered, $k = 0, 1, 2, \ldots$. Throughout this chapter, the *specific* mathematical form of $\hat{\boldsymbol{\theta}}_k$ will change as the algorithm being considered changes. The following notation is used to represent noisy measurements of L at a specific $\boldsymbol{\theta}$:

$$y(\boldsymbol{\theta}) \equiv L(\boldsymbol{\theta}) + \varepsilon(\boldsymbol{\theta}), \tag{7.2}$$

where ε represents the noise terms. Note that the noise terms show dependence on $\boldsymbol{\theta}$. This dependence is relevant for many applications. It indicates that the common statistical assumption of independent, identically distributed (i.i.d.) noise does not necessarily apply since $\boldsymbol{\theta}$ will be changing as the search process proceeds.

Relative to property I, noise fundamentally alters the search and optimization process because the algorithm is getting potentially misleading information throughout the search process. For example, as described in Example 1.4 of Spall (2003), consider the following loss function with a scalar θ: $L(\theta) = e^{-0.1\theta}\sin(2\theta)$. If the domain for optimization is $\Theta = [0, 7]$, the (unique) minimum occurs at $\theta^* = 3\pi/4 \approx 2.36$, as shown in Fig. 7.1. Suppose that the analyst carrying out the optimization is not able to calculate $L(\theta)$, obtaining instead only *noisy* measurements $y(\theta) = L(\theta) + \varepsilon$, where the noises ε are i.i.d. with distribution $N(0, 0.5^2)$ (a normal distribution with mean zero and variance 0.5^2). The analyst uses the $y(\theta)$ measurements in conjunction with an algorithm to attempt to find θ^*.

Consider the experiment depicted in Fig. 7.1 (with data generated via MATLAB). Based on the simple method of collecting one measurement at each increment of 0.1 over the interval defined by Θ (including the endpoints 0 and 7), the analyst would falsely conclude that the minimum is at $\theta = 5.9$. As shown, this false minimum is far from the actual θ^*.

Noise in the loss function measurements arises in almost any case where physical system measurements or computer simulations are used to approximate a steady-state criterion. Some specific areas of relevance include real-time estimation and control problems where data are collected "on the fly" as a system is operating and problems where large-scale simulations are run as estimates of actual system behavior.

Let us summarize two distinct problems involving noise in the loss function measurements: target tracking and simulation-based optimization. In the tracking problem there is a mean-squared error (MSE) criterion of the form

$$L(\boldsymbol{\theta}) = E\left(\|\text{actual output} - \text{desired output}\|^2\right).$$

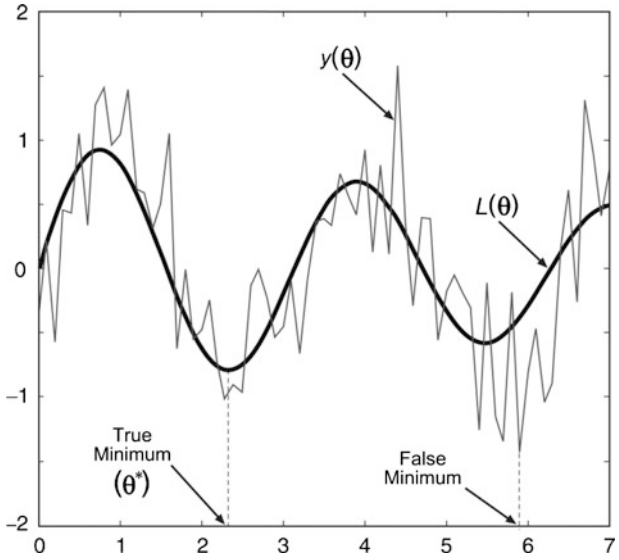


Fig. 7.1 Simple loss function $L(\theta)$ with indicated minimum θ^*. Note how noise causes the algorithm to be deceived into sensing that the minimum is at the indicated false minimum. (Reprinted from Spall, J.C.: Introduction to stochastic search and optimization: estimation, simulation, and control. Wiley, (2003) with permission of John Wiley & Sons, Inc.)

The stochastic optimization algorithm uses the actual (observed) squared error $y(\boldsymbol{\theta}) = \| \cdot \|^2$, which is equivalent to an observation of L embedded in noise. In the simulation problem, let $L(\boldsymbol{\theta})$ be the loss function representing some type of "average" performance for the system. A single run of a Monte Carlo simulation at a specific value of $\boldsymbol{\theta}$ provides a noisy measurement: $y(\boldsymbol{\theta}) = L(\boldsymbol{\theta})+$ noise at $\boldsymbol{\theta}$. (Note that it is rarely desirable to spend computational resources in averaging many simulation runs at a given value of $\boldsymbol{\theta}$; in optimization, it is typically necessary to consider many values of $\boldsymbol{\theta}$.) The above problems are described in more detail in Examples 1.5 and 1.6 in Spall (2003).

Relative to the other defining property of stochastic optimization, property II (i.e., randomness in the search direction), it is sometimes beneficial to deliberately introduce randomness into the search process as a means of speeding convergence and making the algorithm less sensitive to modeling errors. This injected (Monte Carlo) randomness is usually created via computer-based pseudorandom number generators. One of the roles of injected randomness in stochastic optimization is to allow for "surprise" movements to unexplored areas of the search space that may contain an unexpectedly good $\boldsymbol{\theta}$ value. This is especially relevant in seeking out a global optimum among multiple local solutions. Some algorithms that use injected randomness are random search (Sect. 7.2), simultaneous perturbation stochastic approximation (Sect. 7.3), and genetic algorithms (Sect. 7.4).

7.1.4 Some Principles of Stochastic Optimization

The discussion above is intended to motivate some of the issues and challenges in stochastic optimization. Let us now summarize some important issues for the implementation and interpretation of results in stochastic optimization.

The first issue we mention is the fundamental limits in optimization with only noisy information about the L function. Foremost, perhaps, is that the statistical error of the information fed into the algorithm – and the resulting error of the output of the algorithm – can only be reduced by incurring a significant cost in number of function evaluations. For the simple case of independent noise, the error decreases at the rate $1/\sqrt{N}$, where N represents the number of L measurements fed into the algorithm. This is a classical result in statistics, indicating that a 25-fold increase in function evaluations reduces the error by a factor of five.

A further limit for multivariate ($p > 1$) optimization is that the volume of the search region generally grows rapidly with dimension. This implies that one must usually exploit problem structure to have a hope of getting a reasonable solution in a high-dimensional problem.

All practical problems involve at least some restrictions on $\boldsymbol{\theta}$, although in some applications it may be possible to effectively ignore the constraints. Constraints can be encountered in many different ways, as motivated by the specific application. Note that the constraint set Θ does not necessarily correspond to the set of allowable values for $\boldsymbol{\theta}$ *in the search* since some problems allow for the "trial" values of the search to be outside the set of allowable final estimates. Constraints are usually handled in practice on an ad hoc basis, especially tuned to the problem at hand. There are few general, practical methods that apply broadly in stochastic optimization. Michalewicz and Fogel (2000, Chap. 9), for example, discuss some of the practical methods by which constraints are handled in evolutionary computation. Similar methods apply in other stochastic algorithms.

In general search and optimization, it is very difficult (perhaps impossible) to develop automated methods for indicating when the algorithm is close enough to the solution that it can be stopped. Without prior knowledge, there is always the possibility that $\boldsymbol{\theta}^*$ could lie in some unexplored region of the search space. This applies even when the functions involved are relatively benign; see Solis and Wets (1981) for mention of this in the context of twice-differentiable convex L. Difficulties are compounded when the function measurements include noise.

It is quite normal for the environment to change over time. Hence, the solution to a problem now may not be the best (or even a good) solution to the corresponding problem in the future. In some search and optimization problems, the algorithm will be explicitly designed to adapt to a changing environment and automatically provide a new estimate at the optimal value (e.g., a control system). In other cases, one needs to restart the process and find a new solution. In either sense, the problem solving may never stop!

In reading or contributing to the literature on stochastic optimization, it is important to recognize the limits of numerical comparisons by Monte Carlo. Monte

Carlo studies can be a sound scientific method of gaining insight and can be a useful supplement to theory, much of which is based on asymptotic (infinite sample) analysis. In fact, it is especially popular in certain branches of optimization to create "test suites" of problems, where various algorithms compete against each other. A danger arises, however, in making *broad*claims about the performance of individual algorithms based on the results of numerical studies. Performance can vary tremendously under even small changes in the form of the functions involved or the coefficient settings within the algorithms themselves. One must be careful about drawing conclusions beyond those directly supported by the specific numerical studies performed. For purposes of drawing objective conclusions about the relative performance of algorithms, it is preferable to use *both* theory and numerical studies.

Some real systems have one (unique) globally "best" operating point ($\boldsymbol{\theta}^*$) in the domain Θ while others have multiple global solutions (in either case, of course, there could be many *locally*optimal solutions). To avoid excessively cumbersome discussion of algorithms and supporting implementation issues and theory, we often refer to "the" solution $\boldsymbol{\theta}^*$ (versus "a" solution $\boldsymbol{\theta}^*$). In practice, an analyst may be quite satisfied to reach a solution at or close to *any* one $\boldsymbol{\theta}^* \in \Theta^*$.

The so-called *no free lunch* (NFL) theorems provide a formal basis for the intuitively appealing idea that there is a fundamental tradeoff between algorithm efficiency and algorithm robustness (reliability and stability in a broad range of problems). In essence, algorithms that are very efficient on one type of problem are not automatically efficient on problems of a different type. Hence, there can never be a universally best search algorithm just as there is rarely (never?) a universally best solution to any general problem of society. Wolpert and Macready (1997) provided a general formal structure for the NFL theorems, although the general ideas had been around for a long time prior to their paper (Wolpert and Macready were the ones to coin the expression "no free lunch" in this search and optimization context). The NFL theorems are established for discrete optimization with a finite (but arbitrarily large) number of options. However, their applicability includes most practical continuous problems because virtually all optimization is carried out on 32- or 64-bit digital computers. The theorems apply to the cases of both noise-free and noisy loss measurements. NFL states, in essence, that an algorithm that is effective on one class of problems is *guaranteed* to be ineffective on another class. Spall (2003, Sects. 1.2.2 and 10.6) provides more-detailed discussion on the basis and implications of NFL.

We are now in a position to discuss several popular stochastic optimization methods. The summaries here are just that — summaries. Much more complete discussions are available in the indicated references or in Spall (2003). We let $\hat{\boldsymbol{\theta}}_k$ represent the estimate for $\boldsymbol{\theta}$ at the kth iteration of an algorithm under consideration. Section 7.2 discusses random search methods, which are simple and surprisingly powerful in many applications. Section 7.3 discusses stochastic approximation and Sect. 7.4 discusses the popular genetic algorithms. Because of the relative brevity of this review, there are many methods of stochastic optimization not covered here, including simulated annealing, stochastic programming, evolutionary computation

other than genetic algorithms, temporal difference methods, and so on. Readers with an interest in one of those may consult the references mentioned at the end of Sect. 7.1.1.

7.2 Random Search

This section describes some simple methods based on the notion of randomly searching over the domain of interest. Section 7.2.1 gives a short discussion of general issues in direct random search methods. The algorithms discussed in Sect. 7.2.2 represent two versions of random search.

7.2.1 Some General Properties of Direct Random Search

Consider the problem of trying to find the optimal $\boldsymbol{\theta} \in \Theta$ based on noise-free measurements of $L = L(\boldsymbol{\theta})$. Random search methods are perhaps the simplest methods of stochastic optimization in such a setting and can be quite effective in many problems. Their relative simplicity is an appealing feature to both practitioners and theoreticians. These direct random search methods have a number of advantages relative to most other search methods. The advantages include relative ease of coding in software, the need to only obtain L measurements (versus gradients or other ancillary information), reasonable computational efficiency (especially for those direct search algorithms that make use of some local information in their search), broad applicability to non-trivial loss functions and/or to $\boldsymbol{\theta}$ that may be continuous, discrete, or some hybrid form, and a strong theoretical foundation. Some of these attributes were mentioned in the forward-looking paper of Karnopp (1963). A good recent survey of random search and related methods is Kolda et al. (2003).

7.2.2 Two Algorithms for Random Search

This section describes two direct random search techniques. These two algorithms represent only a tiny fraction of available methods. Solis and Wets (1981) and Zhigljavsky (1991) are among many references discussing these and other random search methods. The two algorithms here are intended to convey the essential flavor of most available direct random search algorithms. With the exception of some discussion at the end of the subsection, the methods here assume perfect (noise-free) values of L.

The first method we discuss is "blind random search." This is the simplest random search method, where the current sampling for $\boldsymbol{\theta}$ does not take into account the previous samples. That is, this blind search approach does not adapt the current

sampling strategy to information that has been garnered in the search process. The approach can be implemented in batch (non-recursive) form simply by laying down a number of points in Θ and taking the value of $\boldsymbol{\theta}$ yielding the lowest L value as our estimate of the optimum. The approach can be conveniently implemented in recursive form as we illustrate below.

The simplest setting for conducting the random sampling of new (candidate) values of $\boldsymbol{\theta}$ is when Θ is a hypercube and we are using uniformly generated values of $\boldsymbol{\theta}$. The uniform distribution is continuous or discrete for the elements of $\boldsymbol{\theta}$ depending on the definitions for these elements. *In fact, the blind search form of the algorithm is unique among all general stochastic optimization algorithms in that it is the only one without any adjustable algorithm coefficients that need to be "tuned" to the problem at hand*. (Of course, a de facto tuning decision has been made by choosing the uniform distribution for sampling.)

For a domain Θ that is not a hypercube or for other sampling distributions, one may use transformations, rejection methods, or Markov chain Monte Carlo to generate the sample $\boldsymbol{\theta}$ values (see, e.g., Gentle 2003). For example, if Θ is an irregular shape, one can generate a sample on a hypercube superset containing Θ and then reject the sample point if it lies outside of Θ.

The steps for a recursive implementation of blind random search are given below. This method applies when $\boldsymbol{\theta}$ has continuous, discrete, or hybrid elements.

Blind Random Search

step 0 **(Initialization)**Choose an initial value of $\boldsymbol{\theta}$, say $\hat{\boldsymbol{\theta}}_0 \in \Theta$, either randomly or deterministically. (If random, usually a uniform distribution on Θ is used.) Calculate $L(\hat{\boldsymbol{\theta}}_0)$. Set $k = 0$.

step 1 Generate a new independent value $\boldsymbol{\theta}_{\text{new}}(k+1) \in \Theta$, according to the chosen probability distribution. If $L(\boldsymbol{\theta}_{\text{new}}(k+1)) < L(\hat{\boldsymbol{\theta}}_k)$, set $\hat{\boldsymbol{\theta}}_{k+1} = \boldsymbol{\theta}_{\text{new}}(k+1)$. Else, take $\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k$.

step 2 Stop if the maximum number of L evaluations has been reached or the user is otherwise satisfied with the current estimate for $\boldsymbol{\theta}$ via appropriate stopping criteria; else, return to step 1 with the new k set to the former $k+1$.

The above algorithm converges almost surely (a.s.) to $\boldsymbol{\theta}^*$ under very general conditions (see, e.g., Spall 2003, pp. 40–41). Of course, convergence alone is an incomplete indication of the performance of the algorithm. It is also of interest to examine the *rate* of convergence. The rate is intended to tell the analyst how close $\hat{\boldsymbol{\theta}}_k$ is likely to be to $\boldsymbol{\theta}^*$ for a given cost of search. While blind random search is a reasonable algorithm when $\boldsymbol{\theta}$ is low dimensional, it can be shown that the method is generally a very slow algorithm for even moderately dimensioned $\boldsymbol{\theta}$ (see, e.g., Spall 2003, pp. 42–43). This is a direct consequence of the exponential increase in the size of the search space as p increases. As an illustration, Spall (2003, Example 2.2) considers a case where $\Theta = [0,1]^p$ (the p-dimensional hypercube

with minimum and maximum values of 0 and 1 for each component of $\boldsymbol{\theta}$) and where one wishes to guarantee with probability 0.90 that each element of $\boldsymbol{\theta}$ is within 0.04 units of the optimal value. As p increases from one to ten, there is an approximate 10^{10}-fold increase in the number of loss function evaluations required.

Blind search is the simplest random search in that the sampling generating the new $\boldsymbol{\theta}$ value does not take account of where the previous estimates of $\boldsymbol{\theta}$ have been. The random search algorithm below is slightly more sophisticated in that the random sampling is a function of the position of the current best estimate for $\boldsymbol{\theta}$. In this way, the search is more localized in the neighborhood of that estimate, allowing for a better exploitation of information that has previously been obtained about the shape of the loss function.

The localized algorithm is presented below. This algorithm was described in Matyas (1965). Note that the use of the term "localized" here pertains to the sampling strategy and does not imply that the algorithm is only useful for local (versus global) optimization in the sense described in Sect. 7.1. In fact, the algorithm has global convergence properties as described below. As with blind search, the algorithm may be used for continuous or discrete problems.

Localized Random Search

step 0 **(Initialization)**Pick an initial guess $\hat{\boldsymbol{\theta}}_0 \in \Theta$, either randomly or with prior information. Set $k = 0$.

step 1 Generate an independent random vector $\boldsymbol{d}_k \in \mathbb{R}^p$ and add it to the current $\boldsymbol{\theta}$ value, $\hat{\boldsymbol{\theta}}_k$. Check if $\hat{\boldsymbol{\theta}}_k + \boldsymbol{d}_k \in \Theta$. If $\hat{\boldsymbol{\theta}}_k + \boldsymbol{d}_k \notin \Theta$, generate a new $\boldsymbol{d}_k$ and repeat or, alternatively, move $\hat{\boldsymbol{\theta}}_k + \boldsymbol{d}_k$ to the nearest valid point within Θ. Let $\boldsymbol{\theta}_{\text{new}}(k+1)$ equal $\hat{\boldsymbol{\theta}}_k + \boldsymbol{d}_k \in \Theta$ or the aforementioned nearest valid point in Θ.

step 2 If $L(\boldsymbol{\theta}_{\text{new}}(k+1)) < L(\hat{\boldsymbol{\theta}}_k)$, set $\hat{\boldsymbol{\theta}}_{k+1} = \boldsymbol{\theta}_{\text{new}}(k+1)$; else, set $\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k$.

step 3 Stop if the maximum number of L evaluations has been reached or the user is otherwise satisfied with the current estimate for $\boldsymbol{\theta}$ via appropriate stopping criteria; else, return to step 1 with the new k set to the former $k+1$.

For continuous problems, Matyas (1965) and others have used the (multivariate) normal distribution for generating $\boldsymbol{d}_k$. However, the user is free to set the distribution of the deviation vector $\boldsymbol{d}_k$. The distribution should have mean zero and each component should have a variation (e.g., standard deviation) consistent with the magnitudes of the corresponding $\boldsymbol{\theta}$ elements. This allows the algorithm to assign roughly equal weight to each of the components of $\boldsymbol{\theta}$ as it moves through the search space. Although not formally allowed in the convergence theory, it is often advantageous in practice if the variability in $\boldsymbol{d}_k$ is reduced as k increases. This allows one to focus the search more tightly as evidence is accrued on the location of the solution (as expressed by the location of our current estimate $\hat{\boldsymbol{\theta}}_k$).

The convergence theory for the localized algorithms tends to be more restrictive than the theory for blind search. Solis and Wets (1981) provide a theorem for global convergence of localized algorithms, but the theorem conditions may not be verifiable in practice. An earlier theorem from Matyas (1965) (with proof corrected in Baba et al. 1977) provides for global convergence of the localized search above if L is a continuous function. The convergence is in the "in probability" sense. The theorem allows for more than one global minimum to exist in Θ. Therefore, in general, the result provides no guarantee of $\hat{\boldsymbol{\theta}}_k$ ever settling near any one value $\boldsymbol{\theta}^*$. We present the theorem statement below.

Convergence theorem for localized search. Let Θ^* represent the set of global minima for L (see Sect. 7.1). Suppose that L is continuous on a bounded domain Θ and that if $\hat{\boldsymbol{\theta}}_k + \boldsymbol{d}_k \notin \Theta$ at a given iteration, a new $\boldsymbol{d}_k$ is randomly generated. For any $\eta > 0$, let $R_\eta = \bigcup_{\boldsymbol{\theta}^* \in \Theta^*} \{\boldsymbol{\theta} | L(\boldsymbol{\theta}) - L(\boldsymbol{\theta}^*)| < \eta\}$. Then, for $\boldsymbol{d}_k$ having an i.i.d. $N(\mathbf{0}, \boldsymbol{I}_p)$ distribution, $\lim_{k\to\infty} P(\hat{\boldsymbol{\theta}}_k \in R_\eta) = 1$.

The above algorithm might be considered the most naïve of the localized random search algorithms. More sophisticated approaches are also easy to implement. For instance, if a search in one direction increases L, then it is likely to be beneficial to move in the opposite direction. Further, successive iterations in a direction that tend to consistently reduce L should encourage further iterations in the same direction. Many algorithms exploiting these simple properties exist (e.g., Solis and Wets 1981; Zhigljavsky 1991).

In spite of its simplicity, the localized search algorithm is surprisingly effective in a wide range of problems. Several demonstrations are given in Sects. 2.2.2 to 2.4 in Spall (2003).

The random search algorithms above are usually based on perfect (noise-free) measurements of the loss function. This is generally considered a critical part of such algorithms (Pflug 1996, p. 25). In contrast to the noise-free case, random search methods with noisy loss evaluations of the form $y(\boldsymbol{\theta}) = L(\boldsymbol{\theta}) + \varepsilon(\boldsymbol{\theta})$ generally do not formally converge.

There are, however, means by which the random search techniques can be modified to accommodate noisy measurements, at least on a heuristic basis. Some of the limited formal convergence theory for random search as applied to the noisy measurement case includes Yakowitz (1973, Sect. 4.4.4) and Zhigljavsky (1991, Chap. 3). Spall (2003, Sect. 2.3) discusses some practical methods for coping with noise, including simple averaging of the noisy loss function evaluations $y(\boldsymbol{\theta})$ at each value of $\boldsymbol{\theta}$ generated in the search process and a modification of the algorithm's key decision criterion (step 1 of blind random search and step 2 of localized random search) to build in some robustness to the noise. However, the averaging method can be costly since the error decreases only at the rate of $1/\sqrt{N}$ when averaging N function evaluations with independent noise. Likewise, the altered threshold may be costly by rejecting too many changes in $\boldsymbol{\theta}$ due to the conservative nature of the modified criterion. The presence of noise in the loss evaluations makes the optimization problem so much more challenging that there is little choice but to accept these penalties if one wants to use a simple random search. We see in the next section that

stochastic approximation tends to be more adept at coping with noise at the price of a more restrictive problem setting than the noise-free convergence theorem above.

7.3 Stochastic Approximation

7.3.1 Introduction

Stochastic approximation (SA) is a cornerstone of stochastic optimization. Robbins and Monro (1951) introduced SA as a general root-finding method when only noisy measurements of the underlying function are available. Let us now discuss some aspects of SA as applied to the more specific problem of root-finding in the context of optimization. With a differentiable loss function $L(\boldsymbol{\theta})$, recall the familiar set of p equations and p unknowns for use in finding a minimum $\boldsymbol{\theta}^*$:

$$\boldsymbol{g}(\boldsymbol{\theta}) = \frac{\partial L}{\partial \boldsymbol{\theta}} = \mathbf{0}. \tag{7.3}$$

(Of course, side conditions are required to guarantee that a root of (7.3) is a minimum, not a maximum or saddlepoint.) Note that (7.3) is nominally only directed at local optimization problems, although some extensions to global optimization are possible, as briefly discussed in Sect. 7.3.3. There are a number of approaches for solving the problem represented by (7.3) when *direct* (usually noisy) measurements of the gradient $\boldsymbol{g}$ are available. These typically go by the name of *stochastic gradient* methods (e.g., Spall 2003, Chap. 5). In contrast to the stochastic gradient approach – but consistent with the emphasis in the random search and genetic algorithms (Sects. 7.2 and 7.4 here) – let us focus on SA when only measurements of L are available. However, unlike the emphasis in random search and genetic algorithms, we consider *noisy* measurements of L.

To motivate the general SA approach, first recall the familiar form for the unconstrained deterministic steepest descent algorithm for solving (7.3):

$$\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k - a_k \boldsymbol{g}(\hat{\boldsymbol{\theta}}_k),$$

where the gain (or step size) satisfies $a_k > 0$ (see, e.g., Bazaraa et al. 1993, pp. 300-308 or any other book on mathematical programming; Spall 2003, Sect. 1.4). This algorithm requires exact knowledge of $\boldsymbol{g}$. Steepest descent converges to $\boldsymbol{\theta}^*$ under certain fairly general conditions. (A notable variation of steepest descent is the Newton-Raphson algorithm [sometimes called Newton's method; e.g., Bazaraa et al. 1993, pp. 308–312], which has the form $\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k - a_k \boldsymbol{H}(\hat{\boldsymbol{\theta}}_k)^{-1}\boldsymbol{g}(\hat{\boldsymbol{\theta}}_k)$, where $\boldsymbol{H}(\cdot)$ is the Hessian [second derivative] matrix of L. Under more restrictive conditions, the Newton–Raphson algorithm has a much faster rate of convergence to $\boldsymbol{\theta}^*$ than steepest descent. However, with its requirement for a Hessian matrix, it

is generally more challenging to implement. An SA version of Newton–Raphson is discussed briefly at the end of Sect. 7.3.3.

Unlike with steepest descent, it is assumed here that we have no direct knowledge of $\boldsymbol{g}$. The recursive procedure of interest is in the general SA form

$$\hat{\boldsymbol{\theta}}_{k+1} = \hat{\boldsymbol{\theta}}_k - a_k \hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k), \tag{7.4}$$

where $\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)$ is the estimate of $\boldsymbol{g}$ at the iterate $\hat{\boldsymbol{\theta}}_k$ based on measurements of the loss function. Hence, (7.4) is analogous to the steepest descent algorithm, with the gradient estimate $\hat{\boldsymbol{g}}_k(\boldsymbol{\theta})$ replacing the direct gradient $\boldsymbol{g}$ at $\boldsymbol{\theta} = \hat{\boldsymbol{\theta}}_k$. The gain $a_k > 0$ here also acts in a way similar to its role in the steepest descent form. Under appropriate conditions, the iteration in (7.4) converges to $\boldsymbol{\theta}^*$ in some stochastic sense (usually almost surely, a.s.). (There are constrained forms of SA, but we do not discuss those here; see, e.g., Spall 2003, Chaps. 4–8).

Sections 7.3.2 and 7.3.3 discuss two SA methods for carrying out the optimization task using noisy measurements of the loss function. Section 7.3.2 discusses the traditional finite-difference SA method and Sect. 7.3.3 discusses the more recent simultaneous perturbation method.

7.3.2 *Finite-Difference SA*

The essential part of (7.4) is the gradient approximation $\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)$. The traditional means of forming the approximation is the finite-difference method. Expression (7.4) with this approximation represents the finite-difference SA (FDSA) algorithm. One-sided gradient approximations involve measurements $y(\hat{\boldsymbol{\theta}}_k)$ and $y(\hat{\boldsymbol{\theta}}_k +$ perturbation), while two-sided approximations involve measurements of the form $y(\hat{\boldsymbol{\theta}}_k \pm$ perturbation). The two-sided FD approximation for use with (7.4) is

$$\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k) = \begin{bmatrix} \dfrac{y(\hat{\boldsymbol{\theta}}_k + c_k\boldsymbol{\xi}_1) - y(\hat{\boldsymbol{\theta}}_k - c_k\boldsymbol{\xi}_1)}{2c_k} \\ \vdots \\ \dfrac{y(\hat{\boldsymbol{\theta}}_k + c_k\boldsymbol{\xi}_p) - y(\hat{\boldsymbol{\theta}}_k - c_k\boldsymbol{\xi}_p)}{2c_k} \end{bmatrix}, \tag{7.5}$$

where $\boldsymbol{\xi}_i$ denotes a vector with a 1 in the ith place and 0's elsewhere and $c_k > 0$ defines the difference magnitude. The pair $\{a_k, c_k\}$ are the gains (or gain sequences) for the FDSA algorithm. The two-sided form in (7.5) is the obvious multivariate extension of the scalar two-sided form in Kiefer and Wolfowitz (1952). The initial multivariate method in Blum (1954) used a one-sided approximation.

It is of fundamental importance to determine conditions such that $\hat{\boldsymbol{\theta}}_k$ as shown in (7.4) and (7.5) converges to $\boldsymbol{\theta}^*$ in some appropriate stochastic sense. The convergence theory for the FDSA algorithm is similar to "standard" convergence

theory for the root-finding SA algorithm of Robbins and Monro (1951). Additional difficulties, however, arise due to a bias in $\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)$ as an estimator of $\boldsymbol{g}(\hat{\boldsymbol{\theta}}_k)$. That is, standard conditions for convergence of SA require unbiased estimates of $\boldsymbol{g}(\cdot)$ at all k. On the other hand, $\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)$, as shown in (7.5), is a biased estimator, with the bias having a magnitude of order c_k^2. We do not present the details of the convergence theory here, as it is available in many other references (e.g., Fabian 1971; Kushner and Yin 2003, Chaps. 5–8; Ruppert 1991; Spall 2003, Chap. 6). However, let us note that the standard conditions on the gain sequences are: $a_k > 0$, $c_k > 0$, $a_k \to 0$, $c_k \to 0$, $\sum_{k=0}^{\infty} a_k = \infty$, and $\sum_{k=0}^{\infty} a_k^2/c_k^2 < \infty$. The choice of these gain sequences is critical to the performance of the method. Common forms for the sequences are:

$$a_k = \frac{a}{(k+1+A)^{\alpha}} \quad \text{and} \quad c_k = \frac{c}{(k+1)^{\gamma}},$$

where the coefficients a, c, α, and γ are strictly positive and $A \geq 0$. The user must choose these coefficients, a process usually based on a combination of the theoretical restrictions above, trial-and-error numerical experimentation, and basic problem knowledge. In some cases, it is possible to partially automate the selection of the gains (see, e.g., Spall 2003, Sect. 6).

Let us summarize a numerical example based on the following $p = 10$ loss function:

$$L(\boldsymbol{\theta}) = \boldsymbol{\theta}^T \boldsymbol{B}^T \boldsymbol{B} \boldsymbol{\theta} + 0.1 \sum_{i=1}^{10} (\boldsymbol{B}\boldsymbol{\theta})_i^3 + 0.01 \sum_{i=1}^{10} (\boldsymbol{B}\boldsymbol{\theta})_i^4,$$

where $(\cdot)_i$ represents the ith component of the argument vector $\boldsymbol{B}\boldsymbol{\theta}$, and $\boldsymbol{B}$ is such that $10\boldsymbol{B}$ is an upper triangular matrix of 1's. The minimum occurs at $\boldsymbol{\theta}^* = \boldsymbol{0}$ with $L(\boldsymbol{\theta}^*) = 0$; all runs are initialized at $\hat{\boldsymbol{\theta}}_0 = [1, 1, \ldots, 1]^T$ (so $L(\hat{\boldsymbol{\theta}}_0) = 4.178$). Suppose that the measurement noise ε is independent, identically distributed (i.i.d.) $N(0, 1)$. All iterates $\hat{\boldsymbol{\theta}}_k$ are constrained to be in $\Theta = [-5, 5]^{10}$. If an iterate falls outside of Θ, each individual component of the candidate $\boldsymbol{\theta}$ that violates the interval $[-5, 5]$ is mapped to it nearest endpoint ± 5. The subsequent gradient estimate is formed at the modified (valid) $\boldsymbol{\theta}$ value. (The perturbed values $\hat{\boldsymbol{\theta}}_k \pm c_k \boldsymbol{\xi}_i$ are allowed to go outside of Θ.)

Using $n = 1,000$ loss measurements per run, we compare FDSA with the localized random search method of Sect. 7.2. Based on principles for gain selection in Spall (2003, Sect. 6) together with some limited trial-and-error experimentation, we chose $a = 0.5$, $c = 1$, $A = 5$, $\alpha = 0.602$, and $\gamma = 0.101$ for FDSA and an average of 20 loss measurements per iteration with normally distributed perturbations having distribution $N(\boldsymbol{0}, 0.5^2 \boldsymbol{I}_{10})$ for the random search method.

Figure 7.2 summarizes the results. Each curve represents the sample mean of 50 independent replications. An individual replication of one of the two algorithms has much more variation than the corresponding smoothed curve in the figure.

Figure 7.2 shows that both algorithms produce an overall reduction in the true loss function as the number of measurements approach 1,000. The curves illustrate

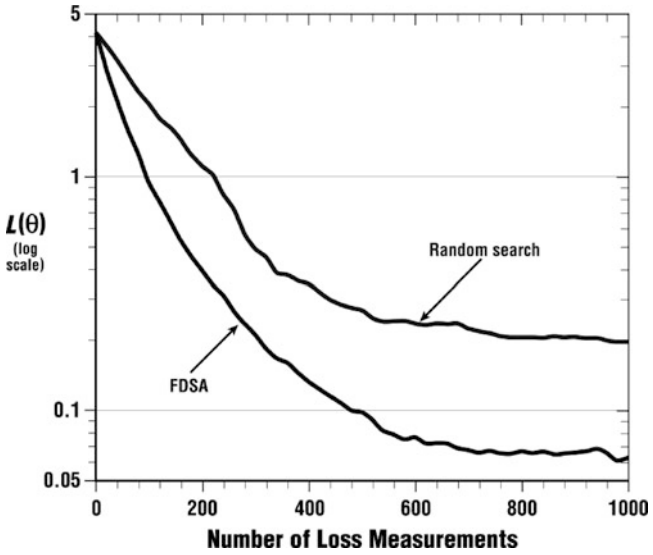


Fig. 7.2 Comparison of FDSA and localized random search. Each curve represents sample mean of 50 independent replications

that FDSA outperforms random search in this case. To make the comparison fair, attempts were made to tune each algorithm to provide approximately the best performance possible. Of course, one must be careful about using this example to infer that such a result holds in other problems as well.

7.3.3 Simultaneous Perturbation SA

The FDSA algorithm of Sect. 7.3.2 is a standard SA method for carrying out optimization with noisy measurement of the loss function. However, as the dimension p grows large, the number of loss measurements required may become prohibitive. That is, each two-sided gradient approximation requires $2p$ loss measurements. More recently, the simultaneous perturbation SA (SPSA) method was introduced, requiring only two measurements per iteration to form a gradient approximation independent of the dimension p. This provides the potential for a large savings in the overall cost of optimization.

Beginning with the generic SA form in (7.4), we now present the SP form of the gradient approximation. In this form, all elements of $\hat{\boldsymbol{\theta}}_k$ are randomly perturbed together to obtain two loss measurements $y(\cdot)$. For the two-sided SP gradient approximation, this leads to

$$\hat{\mathbf{g}}_k(\hat{\boldsymbol{\theta}}_k) = \begin{bmatrix} \dfrac{y(\hat{\boldsymbol{\theta}}_k + c_k\boldsymbol{\Delta}_k) - y(\hat{\boldsymbol{\theta}}_k - c_k\boldsymbol{\Delta}_k)}{2c_k\Delta_{k1}} \\ \vdots \\ \dfrac{y(\hat{\boldsymbol{\theta}}_k + c_k\boldsymbol{\Delta}_k) - y(\hat{\boldsymbol{\theta}}_k - c_k\boldsymbol{\Delta}_k)}{2c_k\Delta_{kp}} \end{bmatrix} \tag{7.6}$$
$$= \frac{y(\hat{\boldsymbol{\theta}}_k + c_k\boldsymbol{\Delta}_k) - y(\hat{\boldsymbol{\theta}}_k - c_k\boldsymbol{\Delta}_k)}{2c_k} \left[\Delta_{k1}^{-1}, \Delta_{k2}^{-1}, \ldots, \Delta_{kp}^{-1}\right]^T,$$

where the mean-zero p-dimensional random perturbation vector, $\boldsymbol{\Delta}_k = [\Delta_{k1}, \Delta_{k2}, \ldots, \Delta_{kp}]^T$, has a user-specified distribution satisfying certain conditions and c_k is a positive scalar (as with FDSA). Because the numerator is the same in all p components of $\hat{\mathbf{g}}_k(\hat{\boldsymbol{\theta}}_k)$, the number of loss measurements needed to estimate the gradient in SPSA is *two*, regardless of the dimension p.

Relative to FDSA, the p-fold measurement savings per iteration, of course, provides only the *potential* for SPSA to achieve large savings in the total number of measurements required to estimate $\boldsymbol{\theta}$ when p is large. This potential is realized if the number of iterations required for effective convergence to an optimum $\boldsymbol{\theta}^*$ does not increase in a way to cancel the measurement savings per gradient approximation. One can use asymptotic distribution theory to address this issue. In particular, both FDSA and SPSA are known to be asymptotically normally distributed under very similar conditions. One can use this asymptotic distribution result to characterize the mean-squared error $E(\|\hat{\boldsymbol{\theta}}_k - \boldsymbol{\theta}^*\|^2)$ for the two algorithms for large k. Fortunately, under fairly broad conditions, the p-fold savings *at* each iteration is preserved *across* iterations. In particular, based on asymptotic considerations:

> Under reasonably general conditions (see Spall 1992, or Spall 2003, Chap. 7), the SPSA and FDSA algorithms achieve the same level of statistical accuracy for a given number of iterations even though SPSA uses only $1/p$ times the number of function evaluations of FDSA (since each gradient approximation uses only $1/p$ the number of function evaluations).

The SPSA Web site www.jhuapl.edu/SPSA includes many references on the theory and application of SPSA. On this Web site, one can find many accounts of numerical studies that are consistent with the efficiency statement above. (Of course, given that the statement is based on asymptotic arguments and associated regularity conditions, one should not assume that the result always holds.) In addition, there are references describing many applications. These include queuing systems, pattern recognition, industrial quality improvement, aircraft design, simulation-based optimization, bioprocess control, neural network training, chemical process control, fault detection, human-machine interaction, sensor placement and configuration, and vehicle traffic management.

We do not present here the formal conditions for convergence and asymptotic normality of SPSA, as such conditions are available in many references (e.g., Dippon and Renz 1997; Gerencsér et al. 1999; Spall 1992; Spall 2003, Chap. 7).

These conditions are essentially identical to the standard conditions for convergence of SA algorithms, with the exception of the additional conditions on the user-generated perturbation vector $\boldsymbol{\Delta}_k$.

The choice of the distribution for generating the $\boldsymbol{\Delta}_k$ is important to the performance of the algorithm. The standard conditions for the elements Δ_{ki} of $\boldsymbol{\Delta}_k$ are that the $\{\Delta_{ki}\}$ are independent for all k, i, identically distributed for all i at each k, symmetrically distributed about zero and uniformly bounded in magnitude for all k. In addition, there is an important inverse moments condition:

$$E\left(\left|\frac{1}{\Delta_{ki}}\right|^{2+2\tau}\right) \leq C$$

for some $\tau > 0$ and $C > 0$. The role of this condition is to control the variation of the elements of $\hat{\boldsymbol{g}}_k(\hat{\boldsymbol{\theta}}_k)$ (which have Δ_{ki} in the denominator). One simple and popular distribution that satisfies the inverse moments condition is the symmetric Bernoulli ± 1 distribution. (In fact, as discussed in Spall 2003, Sect.7.7, this distribution can be shown to be optimal under general conditions when using asymptotic considerations.) Two common mean-zero distributions that do *not* satisfy the inverse moments condition are symmetric uniform and normal with mean zero. The failure of both of these distributions is a consequence of the amount of probability mass near zero. Exercise 7.3 in Spall (2003), illustrates the dramatic performance degradation that can occur through using distributions that violate the inverse moments condition.

As with any real-world implementation of stochastic optimization, there are important practical considerations when using SPSA. One is to attempt to define $\boldsymbol{\theta}$ so that the magnitudes of the $\boldsymbol{\theta}$ elements are similar to one another. This desire is apparent by noting that the magnitudes of all components in the perturbations $c_k\boldsymbol{\Delta}_k$ are identical in the case where identical Bernoulli distributions are used. Although it is not always possible to choose the definition of the elements in $\boldsymbol{\theta}$, in most cases an analyst will have the flexibility to specify the units for $\boldsymbol{\theta}$ to ensure similar magnitudes. Another important consideration is the choice of the gains a_k, c_k. The principles described for FDSA above apply to SPSA as well. Section 7.5 of Spall (2003), provides additional practical guidance.

There have been a number of important extensions of the basic SPSA method represented by the combination of (7.4) and (7.5). Three such extensions are to the problem of global (versus local) optimization, to discrete (versus continuous) problems, and to include second-order-type information (Hessian matrix) with the aim of creating a stochastic analogue to the deterministic Newton–Raphson method.

The use of SPSA for *global* minimization among multiple local minima is discussed in Maryak and Chin (2008). One of their approaches relies on injecting Monte Carlo noise in the right-hand side of the basic SPSA updating step in (7.4). This approach is a common way of converting SA algorithms to global optimizers through the additional "bounce" introduced into the algorithm (Yin 1999). Maryak and Chin (2008) also show that basic SPSA *without* injected noise (i.e., (7.4) and (7.6) without modification) may, under certain conditions, be a global optimizer. Formal justification for this result follows because the random error in the SP

gradient approximation acts in a way that is statistically equivalent to the injected noise mentioned above.

Discrete optimization problems (where $\boldsymbol{\theta}$ may take on discrete or combined discrete/continuous values) are discussed in Gerencsér et al. (1999), Hill (2005), and Wang and Spall (2011). Discrete SPSA relies on a fixed-gain (constant a_k and c_k) version of the standard SPSA method. The parameter estimates produced are constrained to lie on a discrete-valued grid. Although gradients do not exist in this setting, the approximation in (7.6) (appropriately modified) is still useful as an efficient measure of slope information.

Finally, using the simultaneous perturbation idea, it is possible to construct a simple method for estimating the Hessian (or Jacobian) matrix of L while, concurrently, estimating the primary parameters of interest ($\boldsymbol{\theta}$). This adaptive SPSA (ASP) approach produces a stochastic analogue to the deterministic Newton-Raphson algorithm (e.g., Bazaraa et al. 1993, pp. 308–312), leading to a recursion that is optimal or near-optimal in its rate of convergence and asymptotic error. The approach applies in both the gradient-free setting emphasized in this section and in the root-finding/stochastic gradient-based (Robbins-Monro) setting reviewed in Spall (2003, Chaps. 4 and 5). Like the standard SPSA algorithm, the ASP algorithm requires only a small number of loss function (or gradient, if relevant) measurements per iteration – independent of the problem dimension – to adaptively estimate the Hessian and parameters of primary interest. Further information is available at Spall (2000) or Spall (2003, Sect. 7.8). A recent paper (Spall 2009) presents two enhancements to ASP, one related to feedback to reduce the error and the other enhancement related to optimal weighting of input information. Both enhancements are aimed at improving the quality of the estimates for underlying Hessian (or Jacobian) matrices, thereby improving the quality of the estimates for the primary parameters of interest $\boldsymbol{\theta}$.

The Hessian estimation aspect of ASP is also useful in non-SA applications, such as calculating the Fisher information matrix (FIM) for problems where the FIM is difficult to obtain analytically (e.g., Spall 2005; Das et al. 2010). The FIM has wide applications in areas such as uncertainty calculation (Ljung 1999, pp. 215–219), experimental design (Spall 2003, Chap. 17; Spall 2010), and Bayesian prior distribution selection (Jeffreys 1946). The Hessian estimation provides an efficient Monte Carlo method for determining the FIM in difficult high-dimensional problems.

7.4 Genetic Algorithms

7.4.1 Introduction

Genetic algorithms (GAs) represent a popular approach to stochastic optimization, especially as relates to the global optimization problem of finding the best solution among multiple local mimima. (GAs may be used in general search problems that

are not directly represented as stochastic optimization problems, but we focus here on their use in optimization.) GAs represent a special case of the more general class of evolutionary computation algorithms (which also includes methods such as evolutionary programming and evolution strategies). The GA applies when the elements of $\boldsymbol{\theta}$ are real-, discrete-, or complex-valued. As suggested by the name, the GA is based loosely on principles of natural evolution and survival of the fittest. In fact, in GA terminology, an equivalent *maximization*criterion, such as $-L(\boldsymbol{\theta})$ (or its analogue based on a bit-string form of $\boldsymbol{\theta}$), is often referred to as the *fitness function* to emphasize the evolutionary concept of the fittest of a species.

A fundamental difference between GAs and the random search and SA algorithms considered in Sects. 7.2 and 7.3 is that GAs work with a *population* of candidate solutions to the problem. The previous algorithms worked with one solution and moved toward the optimum by updating this one estimate. GAs simultaneously consider multiple candidate solutions to the problem of minimizing L and iterate by moving this population of solutions toward a global optimum. The terms *generation* and *iteration* are used interchangeably to describe the process of transforming one population of solutions to another. Figure 7.3 illustrates the successful operations of a GA for a population of size 12 with problem dimension $p = 2$. In this conceptual illustration, the population of solutions eventually come together at the global optimum.

The use of a population versus a single solution affects in a basic way the range of practical problems that can be considered. In particular, GAs tend to be best suited to problems where the loss function evaluations are computer-based calculations such as complex function evaluations or simulations. This contrasts with the single-solution approaches discussed earlier, where the loss function evaluations may represent computer-based calculations *or* physical experiments. Population-based approaches are not generally feasible when working with real-time physical experiments. Implementing a GA with physical experiments requires that either there be multiple identical experimental setups (parallel processing) or that the single experimental apparatus be set to the same state prior to each population member's loss evaluation (serial processing). These situations do not occur often in practice.

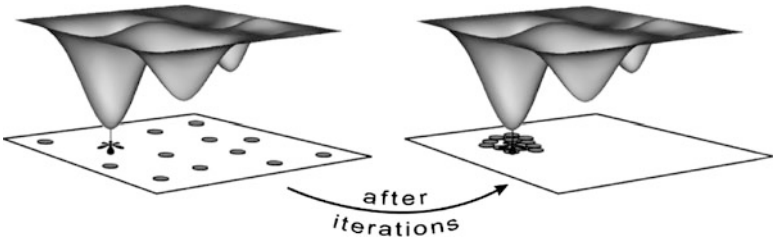


Fig. 7.3 Minimization of multimodal loss function. Successful operations of a GA with a population of 12 candidate solutions clustering around the global minimum after some number of iterations (generations). (Reprinted from Spall, J.C.: Introduction to stochastic search and optimization: estimation, simulation, and control. Wiley, (2003) with permission of John Wiley & Sons, Inc.)

Specific values of $\boldsymbol{\theta}$ in the population are referred to as *chromosomes*. The central idea in a GA is to move a set (population) of chromosomes from an initial collection of values to a point where the fitness function is optimized. We let N denote the population size (number of chromosomes in the population). Most of the early work in the field came from those in the fields of computer science and artificial intelligence. More recently, interest has extended to essentially all branches of business, engineering, and science where search and optimization are of interest. The widespread interest in GAs appears to be due to the success in solving many difficult optimization problems. Unfortunately, to an extent greater than with other methods, some interest appears also to be due to a regrettable amount of "salesmanship" and exaggerated claims. (For example, in a recent software advertisement, the claim is made that the software ". . .uses GAs to solve *any* optimization problem." Such statements are provably false.) While GAs are important tools within stochastic optimization, there is no formal evidence of consistently superior performance – relative to other appropriate types of stochastic algorithms – in any broad, identifiable class of problems.

Let us now give a very brief historical account. The reader is directed to Goldberg (1989, Chap. 4), Mitchell (1996, Chap. 1),Michalewicz (1996, pp. 1–10), Fogel (2000, Chap. 3), and Spall (2003, Sect. 9.2) for more complete historical discussions. There had been some success in creating mathematical analogues of biological evolution for purposes of search and optimization since at least the 1950s (e.g., Box 1957). The cornerstones of modern evolutionary computation – evolution strategies, evolutionary programming, and GAs – were developed independently of each other in the 1960s and 1970s. John Holland at the University of Michigan published the seminal monograph *Adaptation in Natural and Artificial Systems* (Holland 1975). There was subsequently a sprinkle of publications, leading to the first full-fledged textbook Goldberg (1989). Activity in GAs grew rapidly beginning in the mid-1980s, roughly coinciding with resurgent activity in other artificial intelligence-type areas such as neural networks and fuzzy logic. There are now many conferences and books in the area of evolutionary computation (especially GAs), together with countless other publications.

7.4.2 Chromosome Coding and the Basic GA Operations

This section summarizes some aspects of the encoding process for the population chromosomes and discusses the *selection*, *elitism*, *crossover*, and *mutation* operations. These operations are combined to produce the steps of the GA.

An essential aspect of GAs is the encoding of the N values of $\boldsymbol{\theta}$ appearing in the population. This encoding is critical to the GA operations and the associated decoding to return to the natural problem space in $\boldsymbol{\theta}$. Standard binary (0, 1) bit strings have traditionally been the most common encoding method, but other methods include gray coding (which also uses (0, 1) strings, but differs in the way the bits are arranged) and basic computer-based floating-point representation of the

real numbers in $\boldsymbol{\theta}$. This 10-character coding is often referred to as *real-number coding* since it operates as if working with $\boldsymbol{\theta}$ directly. Based largely on successful numerical implementations, this natural representation of $\boldsymbol{\theta}$ has grown more popular over time. Details and further references on the above and other coding schemes are given in Michalewicz (1996, Chap. 5), Mitchell (1996, Sects. 5.1.1 and 5.1.2), Fogel (2000, Sects. 3.5 and 4.3), and Spall (2003, Sect. 9.3).

Let us now describe the basic operations mentioned above. For consistency with standard GA terminology, let us assume that $L(\boldsymbol{\theta})$ has been transformed to a fitness function with higher values being better. A common transformation is to simply set the fitness function to $-L(\boldsymbol{\theta}) + C$, where $C \geq 0$ is a constant that ensures that the fitness function is nonnegative on Θ (nonnegativity is only required in some GA implementations). Hence, the operations below are described for a *maximization* problem. It is also assumed here that the fitness evaluations are noise-free. Unless otherwise noted, the operations below apply with any coding scheme for the chromosomes.

The *selection* and *elitism* steps occur after evaluating the fitness function for the current population of chromosomes. A subset of chromosomes is selected to use as parents for the succeeding generation. This operation is where the survival of the fittest principle arises, as the parents are chosen according to their fitness value. While the aim is to emphasize the fitter chromosomes in the selection process, it is important that not *too* much priority is given to the chromosomes with the highest fitness values early in the optimization process. Too much emphasis of the fitter chromosomes may tend to reduce the diversity needed for an adequate search of the domain of interest, possibly causing premature convergence in a local optimum. Hence methods for selection allow with some nonzero probability the selection of chromosomes that are suboptimal.

Associated with the selection step is the optional "elitism" strategy, where the $N_e < N$ best chromosomes (as determined from their fitness evaluations) are placed directly into the next generation. This guarantees the preservation of the N_e best chromosomes at each generation. Note that the elitist chromosomes in the original population are also eligible for selection and subsequent recombination.

As with the coding operation for $\boldsymbol{\theta}$, many schemes have been proposed for the selection process of choosing parents for subsequent recombination. One of the most popular methods is *roulette wheel selection* (also called *fitness proportionate selection*). In this selection method, the fitness functions must be nonnegative on Θ. An individual's slice of a Monte Carlo-based roulette wheel is an area proportional to its fitness. The "wheel" is spun in a simulated fashion $N - N_e$ times and the parents are chosen based on where the pointer stops. Another popular approach is called *tournament selection*. In this method, chromosomes are compared in a "tournament," with the better chromosome being more likely to win. The tournament process is continued by sampling (with replacement) from the original population until a full complement of parents has been chosen. The most common tournament method is the binary approach, where one selects two pairs of chromosomes and chooses as the two parents the chromosome in each pair having the higher fitness value. Empirical evidence suggests that the tournament

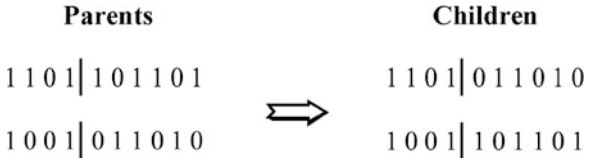


Fig. 7.4 Example of crossover operator under binary coding with one splice point

selection method often performs better than roulette selection. (Unlike tournament selection, roulette selection is very sensitive to the scaling of the fitness function.) Mitchell (1996, Sect. 5.4) provides a good survey of several other selection methods.

The *crossover* operation creates offspring of the pairs of parents from the selection step. A crossover probability P_c is used to determine if the offspring represents a blend of the chromosomes of the parents. If no crossover takes place, then the two offspring are clones of the two parents. If crossover does take place, then the two offspring are produced according to an interchange of parts of the chromosome structure of the two parents. Figure 7.4 illustrates this for the case of a ten-bit binary representation of the chromosomes. This example shows one-point crossover, where the bits appearing after one randomly chosen dividing (splice) point in the chromosome are interchanged. In general, one can have a number of splice points up to the number of bits in the chromosomes minus one, but one-point crossover appears to be the most commonly used.

Note that the crossover operator also applies directly with real-number coding since there is nothing directly connected to binary coding in crossover. All that is required are two lists of compatible symbols. For example, one-point crossover applied to the chromosomes ($\boldsymbol{\theta}$ values) $[6.7, -7.4, 4.0, 3.9|6.2, -1.5]$ and $[-3.8, 5.3, 9.2, -0.6|8.4, -5.1]$ yields the two children: $[6.7, -7.4, 4.0, 3.9, 8.4, -5.1]$ and $[-3.8, 5.3, 9.2, -0.6, 6.2, -1.5]$.

The final operation we discuss is *mutation*. Because the initial population may not contain enough variability to find the solution via crossover operations alone, the GA also uses a mutation operator where the chromosomes are randomly changed. For the binary coding, the mutation is usually done on a bit-by-bit basis where a chosen bit is flipped from 0 to 1, or vice versa. Mutation of a given bit occurs with small probability P_m. Real-number coding requires a different type of mutation operator. That is, with a (0, 1)-based coding, an opposite is uniquely defined, but with a real number, there is no clearly defined opposite (e.g., it does not make sense to "flip" the 2.74 element). Probably the most common type of mutation operator is simply to add small independent normal (or other) random vectors to each of the chromosomes (the $\boldsymbol{\theta}$ values) in the population.

As discussed in Sect. 7.1.4, there is no easy way to know when a stochastic optimization algorithm has effectively converged to an optimum. This includes GAs. The one obvious means of stopping a GA is to end the search when a budget of fitness (equivalently, loss) function evaluations has been spent. Alternatively, termination may be performed heuristically based on subjective and objective

impressions about convergence. In the case where noise-free fitness measurements are available, criteria based on fitness evaluations may be most useful. For example, a fairly natural criterion suggested in Schwefel (1995, p. 145) is to stop when the maximum and minimum fitness values over the N population values within a generation are sufficiently close to one another. However, this criterion provides no formal guarantee that the algorithm has found a global solution.

7.4.3 The Core Genetic Algorithm

The steps of a basic form of the GA are given below. These steps are general enough to govern many (perhaps most) modern implementations of GAs, including those in modern commercial software. Of course, the performance of a GA typically depends greatly on the implementation details, just as with other stochastic optimization algorithms. Some of these practical implementation issues are taken up in the next section.

Core GA Steps for Noise-Free Fitness Evaluations

step 0 **(Initialization)** Randomly generate an initial population of N chromosomes and evaluate the fitness function (the conversion of $L(\boldsymbol{\theta})$ to a function to be maximized for the encoded version of $\boldsymbol{\theta}$) for each of the chromosomes.

step 1 **(Parent selection)** Set $N_e = 0$ if elitism strategy is not used; $0 < N_e < N$ otherwise. Select with replacement $N - N_e$ parents from the full population (including the N_e elitist elements). The parents are selected according to their fitness, with those chromosomes having a higher fitness value being selected more often.

step 2 **(Crossover)** For each pair of parents identified in step 1, perform crossover on the parents at a randomly (perhaps uniformly) chosen splice point (or *points* if using multi-point crossover) with probability P_c. If no crossover takes place (probability $1 - P_c$), then form two offspring that are exact copies (clones) of the two parents.

step 3 **(Replacement and mutation)** While retaining the N_e best chromosomes from the previous generation, replace the remaining $N - N_e$ chromosomes with the current population of offspring from step 2. For the bit-based implementations, mutate the individual bits with probability P_m; for real coded implementations, use an alternative form of "small" modification (in either case, one has the option of choosing whether to make the N_e elitist chromosomes candidates for mutation).

step 4 **(Fitness and end test)** Compute the fitness values for the new population of N chromosomes. Terminate the algorithm if the stopping criterion is met or if the budget of fitness function evaluations is exhausted; else return to step 1.

7.4.4 Some Implementation Aspects

While the above steps provide the broad outline for many modern implementations of GAs, there are some important implementation aspects that must be decided before a practical implementation. This section outlines a few of those aspects. More detailed discussions are given in Mitchell (1996, Chap. 5), Michalewicz (1996, Chaps. 4– 6), Fogel (2000, Chaps. 3 and 4), Goldberg (2002, Chap. 12), and other references mentioned below. A countless number of numerical studies have been reported in the literature; we do not add to that list here.

As with other stochastic optimization methods, the choice of algorithm-specific coefficients has a significant impact on performance. With GAs, there is a relatively large number of user decisions required. The following must be set: the choice of chromosome encoding, the population size (N), the probability distribution generating the initial population, the strategy for parent selection (roulette wheel or otherwise), the number of splice points in the crossover, the crossover probability (P_c), the mutation probability (P_m), the number of retained chromosomes in elitism (N_e), and some termination criterion. Some typical values for these quantities are discussed, for example, in Mitchell (1996, pp. 175–177) and Spall (2003, Sect. 9.6).

Constraints on $L(\boldsymbol{\theta})$ (or the equivalent fitness function) and/or $\boldsymbol{\theta}$ are of major importance in practice. The bit-based implementation of GAs provide a natural way of implementing component-wise lower and upper bounds on the elements of $\boldsymbol{\theta}$ (i.e., a hypercube constraint). More general approaches to handling constraints are discussed in Michalewicz (1996, Chap. 8 and Sects. 4.5 and 15.3) and Michalewicz and Fogel (2000, Chap. 9).

Until now, it has been assumed that the fitness function is observed without noise. One of the two possible defining characteristics of stochastic optimization, however, is optimization with noise in the function measurements (property I in Sect. 7.1.3). There appears to be relatively little formal analysis of GAs in the presence of noise, although the application and testing of GAs in such cases has been carried out since at least the mid-1970s (e.g., De Jong 1975, p. 203). A large number of numerical studies are in the literature (e.g., the references and studies in Spall (2003, Sects. 9.6 and 9.7). As with other algorithms, there is a fundamental tradeoff of more accurate information for each function input (typically, via an averaging of the inputs) and fewer function inputs versus less accurate ("raw") information to the algorithm together with a greater number of inputs to the algorithm. There appears to be no rigorous comparison of GAs with other algorithms regarding relative robustness to noise. Regarding noise, Michalewicz and Fogel (2000, p. 325) state: "There really are no effective heuristics to guide the choices to be made that will work in general."

7.4.5 Some Comments on the Theory for GAs

One of the key innovations in Holland (1975) was the attempt to put GAs on a stronger theoretical footing than the previous ad hoc treatments. He did

this by the introduction of schema theory. While many aspects and implications of schema theory have subsequently been challenged (Reeves and Rowe 2003, Chap. 3; Spall 2003, Sect. 10.3), some aspects remain viable. In particular, schema theory *itself* is generally correct (subject to a few modifications), although many of the assumed *implications* have not been correct. With the appropriate caveats and restrictions, schema theory provides some intuitive explanation for the good performance that is frequently observed with GAs.

More recently, Markov chains have been used to provide a formal structure for analyzing GAs. First, let us mention one negative result. Markov chains can be used to show that a canonical GA *without elitism* is (in general) provably nonconvergent (Rudolph 1994). That is, with a GA that does not hold onto the best solution at each generation, there is the possibility (through crossover and mutation) that a chromosome corresponding to $\boldsymbol{\theta}^*$ will be lost. (Note that the GA without elitism corresponds to the form in Holland 1975.)

On the other hand, conditions for the formal convergence of GAs to an optimal $\boldsymbol{\theta}^*$ (or its coded equivalent) are presented in Vose (1999, Chaps. 13 and 14), Fogel (1999, Chap. 4), (Reeves and Rowe 2003, Chap. 6), and Spall (2003, Sect. 10.5), among other references. Consider a binary bit-coded GA with a population size of N and a string length of B bits per chromosome. Then the total number of possible *unique* populations is:

$$N_P \equiv \binom{N + 2^B - 1}{N} = \frac{(N + 2^B - 1)!}{(2^B - 1)!N!}$$

(Suzuki 1995). It is possible to construct an $N_P \times N_P$ Markov transition matrix $\boldsymbol{P}$, where the ijth element is the probability of transitioning from the ith population of N chromosomes to the jth population of the same size. These elements depend in a nontrivial way on N, the crossover rate, and the mutation rate; the number of elite chromosomes is assumed to be $N_e = 1$ (Suzuki 1995). Let $\boldsymbol{p}_k$ be an $N_P \times 1$ vector having jth component $p_k(j)$ equal to the probability that the kth generation will result in population j, $j = 1, 2, \ldots, N_P$.

From basic Markov chain theory,

$$\boldsymbol{p}_{k+1}^T = \boldsymbol{p}_k^T \boldsymbol{P} = \boldsymbol{p}_0^T \boldsymbol{P}^{k+1},$$

where $\boldsymbol{p}_0$ is an initial probability distribution. A standard result in Markov chain theory is that if the chain is irreducible and ergodic (see, e.g., Spall 2003, Appendix E, Theorem E.1), then the limiting distribution of the GA exists and satisfies the stationarity equation. (Recall from basic Markov chain theory that irreducibility indicates that any state may be reached from any other state after a finite number of steps.) However, the chain for a GA is *not* irreducible because the GA cannot move to a population whose best fitness value is lower than the current best fitness (hence, the convergence Theorem E.1 in Spall 2003, does not apply). Nevertheless, the chain *does* have a unique limiting value $\bar{\boldsymbol{p}}^T$ satisfying the stationarity equation $\bar{\boldsymbol{p}}^T = \bar{\boldsymbol{p}}^T \boldsymbol{P}$. An individual element in $\boldsymbol{P}$ can be computed according to the formulas in Suzuki (1995) and Stark and Spall (2003). These elements depend in

a nontrivial way on N, the crossover rate, and the mutation rate; the number of elite chromosomes is assumed to be $N_e = 1$.

Suppose that $\boldsymbol{\theta}^*$ is unique (i.e., Θ^* is the singleton $\boldsymbol{\theta}^*$). Let $J \subseteq \{1, 2, \ldots, N_P\}$ be the set of indices corresponding to the populations that contain at least one chromosome representing $\boldsymbol{\theta}^*$. So, for example, if $J = \{1, 6, N_P - 3\}$, then each of the three populations indexed by 1, 6 and $N_P - 3$ contains at least one chromosome that, when decoded, is equal to $\boldsymbol{\theta}^*$. Under the above-mentioned assumptions of irreducibility and ergodicity, $\sum_{i \in J} \bar{p}_i = 1$, where $\bar{p}_i$ is the ith element of $\bar{\boldsymbol{p}}$. Hence, a GA with $N_e = 1$ and a transition matrix that is irreducible and ergodic converges in probability to $\boldsymbol{\theta}^*$.

To establish the fact of convergence alone, it may not be necessary to compute the $\boldsymbol{P}$ matrix. Rather, it suffices to know that the chain is irreducible and ergodic. (For example, Rudolph 1997, p. 125, shows that the Markov chain approach yields convergence when $0 < P_m < 1$.) However, $\boldsymbol{P}$ must be explicitly computed to get the *rate* of convergence information that is available from $\boldsymbol{p}_k$. This is rarely possible in practice because the number of states in the Markov chain (and hence dimension of the Markov transition matrix) grows *very* rapidly with increases in the population size and/or the number of bits used in coding for the population elements. For example, in even a trivial problem of $N = B = 6$, there are $\sim 10^8$ states and $\sim 10^{16}$ elements in the transition matrix; this problem is much smaller than any practical GA, which can easily have 50 to 100 population elements and 15 to 40 bits per population element (leading to well over 10^{100} states, with each element in the corresponding row and column in the transition matrix requiring significant computation).

7.5 Concluding Remarks

Stochastic optimization is a major branch of computational statistics. This chapter has been a whirlwind tour through some important issues and methods in stochastic optimization. Stochastic optimization applies when there are noisy measurements of the criterion being optimized and/or there is an injected Monte Carlo randomness as part of the algorithm. Of necessity, we cover only a small fraction of available methods in this relatively brief review, although the methods described (random search, stochastic approximation, and genetic algorithms) are representative of a broad range of important and widely used algorithms. Further, the treatment here on the specific algorithms is relatively brief. In particular, the subjects covered in this chapter of approximately 30 pages are treated in over 160 pages in Spall (2003, Chaps. 1–2, 6–8, and 9–10) and are given an even more detailed treatment in the many specialized books or other references.

There are many challenges to carrying out real-world optimization, including the presence of noise in the function evaluations, the difficulties in distinguishing a globally optimal solution from locally optimal solutions, the "curse of dimensionality," the difficulties associated with nontrivial constraints, and the lack of stationarity in

the solution as a result of the conditions of the problem changing over time. Stochastic optimization methods are especially useful in treating some of these challenges. In particular, by definition, they are designed for noisy function evaluations. Further, when considering injected (Monte Carlo) randomness (property II in Sect. 7.1.3), certain stochastic optimization algorithms will (under conditions, of course) serve as global optimizers. That is, the injected randomness provides enough "bounce" to the algorithm to allow for escape from local minima en route to achieving a global minimum.

In summary, while classical deterministic optimization methods (linear and nonlinear programming) are effective for a range of problems, stochastic methods are able to handle many of the problems for which deterministic methods are inappropriate. It is hoped that this summary gives the reader a flavor of the issues, algorithms, and challenges in carrying out optimization in the face of stochastic effects.

Acknowledgements I appreciate the helpful comments of Dr. Stacy Hill on a draft version of this chapter. Funding was provided by the U.S. Navy (contract N00024-03-D-6606) and the JHU/APL Independent Research and Development (IRAD) Program. Selected parts of this article have been reprinted, by permission, from J.C. Spall, *Introduction to Stochastic Search and Optimization*, ©2003 by John Wiley and Sons, Inc.

References

Arsham, H.: Techniques for Monte Carlo optimizing. Monte Carlo Methods Appl. **4**, 181–229 (1998)

Baba, N., Shoman, T., Sawaragi, Y.: A modified convergence theorem for a random optimization method. Inf. Sci. **13**, 159–166 (1977)

Bazaraa, M.S., Sherali, H.D., Shetty, C.M.: Nonlinear programming: theory and algorithms. Wiley, New York (1993)

Blum, J.R.: Multidimensional stochastic approximation methods. Ann. Math. Stat. **25**, 737–744 (1954)

Box, G.E.P.: Evolutionary operation: a method for increasing industrial productivity. J. R. Stat. Soc. Ser. C: Appl. Stat. **6**, 81–101 (1957)

Cochran, J.J. (ed.): Encyclopedia of operations research and management science. Wiley, Hoboken (2011)

Das, S., Spall, J.C., Ghanem, R.: Efficient Monte Carlo computation of Fisher information matrix using prior information. Comput. Stat. Data Anal. **54**(2), 272–289 (2010)

De Jong, K.A.: An analysis of the behavior of a class of genetic adaptive systems. Ph.D. dissertation, University of Michigan (University Microfilms no. 76–9381) (1975)

Dippon, J., Renz, J.: Weighted means in stochastic approximation of minima. SIAM J. Contr. Optim. **35**, 1811–1827 (1997)

Fabian, V.: Stochastic approximation. In: Rustigi, J.S. (ed.) Optimizing methods in statistics, pp. 439–470. Academic, New York (1971)

Fogel, D.B.: Evolutionary computation: toward a new philosophy of machine intelligence. IEEE, Piscataway (2000)

Fouskakis, D., Draper, D.: Stochastic optimization: a review. Int. Stat. Rev. **70**, 315–349 (2002)

Fu, M.C.: Optimization for simulation: theory vs. practice. INFORMS J. Comput. **14**, 192–227 (2002)

Gentle, J.E.: Random number generation and Monte Carlo methods. Springer, New York (2003)

Gerencsér, L.: Convergence rate of moments in stochastic approximation with simultaneous perturbation gradient approximation and resetting. IEEE Trans. Autom. Control **44**, 894–905 (1999)
Gerencsér, L., Hill, S.D., Vágó, Z.: Optimization over discrete sets via SPSA. In: Proceedings of the IEEE Conference on Decision and Control, Phoenix, 7–10 December 1999, 1791–1795
Goldberg, D.E.: Genetic algorithms in search, optimization, and machine learning. Addison-Wesley, Reading (1989)
Goldberg, D.E.: The design of innovation: lessons from and for competent genetic algorithms. Kluwer Academic, Boston (2002)
Gosavi, A.: Simulation-based optimization: parametric optimization techniques and reinforcement learning. Kluwer Academic, Boston (2003)
Hill, S.D.: Discrete stochastic approximation with application to resource allocation. Johns Hopkins APL Tech. Dig. **26**, 15–21 (2005)
Holland, J.H.: Adaptation in natural and artificial systems. University of Michigan Press, Ann Arbor (1975)
Jeffreys, H.: An invariant form for the prior probability in estimation problems. Proc. R. Soc. Lond. A: Math. Phys. Sci. **186**, 453–461 (1946)
Karnopp, D.C.: Random search techniques for optimization problems. Automatica **1**, 111–121 (1963)
Kiefer, J., Wolfowitz, J.: Stochastic estimation of a regression function. Ann. Math. Stat. **23**, 462–466 (1952)
Kolda, T.G., Lewis, R.M., Torczon, V.: Optimization by direct search: new perspectives on some classical and modern methods. SIAM Rev. **45**, 385–482 (2003)
Kushner, H.J., Yin, G.G.: Stochastic approximation and recursive algorithms and applications. Springer, New York (2003)
Ljung, L.: System identification – theory for the user. Prentice Hall PTR, Upper Saddle River (1999)
Maryak, J.L., Chin, D.C.: Global random optimization by simultaneous perturbation stochastic approximation. IEEE Trans. Autom. Control **53**, 780–783 (2008)
Matyas, J.: Random optimization. Autom. Remote Control **26**, 244–251 (1965)
Michalewicz, Z.: Genetic algorithms + data structures = evolution programs. Springer, New York (1996)
Michalewicz, Z., Fogel, D.B.: How to solve it: modern heuristics. Springer, New York (2000)
Mitchell, M.: An introduction to genetic algorithms. MIT Press, Cambridge (1996)
Nelder, J. A., Mead, R., A simplex method for function minimization. Comput. J., **7**, 308–313 (1965)
Pflug, G.C.h.: Optimization of stochastic models: the interface between simulation and optimization. Kluwer Academic, Boston (1996)
Reeves, C.R., Rowe, J.E.: Genetic algorithms – principles and perspectives: a guide to GA theory. Kluwer Academic, Boston (2003)
Robbins, H., Monro, S.: A stochastic approximation method. Ann. Math. Stat. **22**, 400–407 (1951)
Rudolph, G.: Convergence analysis of Canonical genetic algorithms. IEEE Trans. Neural Netw. **5**, 96–101 (1994)
Rudolph, G.: Convergence properties of evolutionary algorithms. Kovac, Hamburg (1997)
Ruppert, D.: Stochastic approximation. In: Ghosh, B.K., Sen, P.K. (eds.) Handbook of sequential analysis, pp. 503–529. Dekker, New York (1991)
Schwefel, H.P.: Evolution and optimum seeking. Wiley, New York (1995)
Solis, F.J., Wets, J.B.: Minimization by random search techniques. Math. Oper. Res. **6**, 19–30 (1981)
Spall, J.C.: Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. IEEE Trans. Autom. Control **37**, 332–341 (1992)
Spall, J.C.: Adaptive stochastic approximation by the simultaneous perturbation method. IEEE Trans. Autom. Control **45**, 1839–1853 (2000)

Spall, J.C.: Introduction to stochastic search and optimization: estimation, simulation, and control. Wiley, Hoboken (2003)
Spall, J.C.: Monte Carlo computation of the Fisher information matrix in nonstandard settings. J. Comput. Graph. Stat. **14**(4), 889–909 (2005)
Spall, J.C.: Feedback and weighting mechanisms for improving Jacobian estimates in the adaptive simultaneous perturbation algorithm. IEEE Trans. Autom. Control **54**(6), 1216–1229 (2009)
Spall, J.C.: Factorial design for choosing input values in experimentation: generating informative data for system identification. IEEE Control Syst. Mag. **30**(5), 38–53 (2010)
Stark, D.R., Spall, J.C.: Rate of convergence in evolutionary computation. In: Proceedings of the American Control Conference, Denver, 4–6 June 2003
Suzuki, J.: A Markov chain analysis on simple genetic algorithms. IEEE Trans. Syst. Man. Cybern. **25**, 655–659 (1995)
Vose, M.: The simple genetic algorithm. MIT Press, Cambridge (1999)
Wang, Q., Spall, J.C.: Discrete simultaneous perturbation stochastic approximation on loss functions with noisy measurements. In: Proceedings of the American Control Conference, San Francisco, 29 June–1 July 2011, pp. 4520–4525 (paper FrB10.3) (2011)
Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Trans. Evol. Comput. **1**, 67–82 (1997)
Yakowitz, S.J., Fisher, L.: On sequential search for the maximum of an unknown function. J. Math. Anal, Appl. **41**, 234–259 (1973)
Yin, G.: Rates of convergence for a class of global stochastic optimization algorithms. SIAM J. Optim. **10**, 99–120 (1999)
Zhigljavsky, A.A.: Theory of global random search. Kluwer Academic, Boston (1991)

Chapter 8
Transforms in Statistics

Brani Vidakovic

It is not an overstatement to say that statistics is based on various transformations of data. Basic statistical summaries such as the sample mean, variance, z-scores, histograms, etc., are all transformed data. Some more advanced summaries, such as principal components, periodograms, empirical characteristic functions, etc., are also examples of transformed data. To give a just coverage of transforms utilized in statistics will take a size of a monograph. In this chapter we will focus only on several important transforms with the emphasis on novel multiscale transforms (wavelet transforms and its relatives).

Transformations in statistics are utilized for several reasons, but unifying arguments are that transformed data:

(1) Are easier to report, store, and analyze,
(2) Comply better with a particular modeling framework, and
(3) Allow for an additional insight to the phenomenon not available in the domain of non-transformed data.

For example, variance stabilizing transformations, symmetrizing transformations, transformations to additivity, Laplace, Fourier, Wavelet, Gabor, Wigner–Ville, Hugh, Mellin, transforms all satisfy one or more of points listed in (1–3).

We emphasize that words *transformation* and *transform* are often used interchangeably. However, the semantic meaning of the two words seem to be slightly different. For the word transformation, the synonyms are alteration, evolution, change, reconfiguration. On the other hand, the word transform carries the meaning of a more radical change in which the nature and/or structure of the transformed object are altered. In our context, it is natural that processes which alter the data leaving them unreduced in the same domain should be called transformations (for

B. Vidakovic (✉)
The Wallace H. Coulter Department of Biomedical Engineering, Georgia Institute of Technology
2101 Whitaker Building, 313 Ferst Drive, Atlanta, GA 30332-0535, USA
e-mail: brani@bme.gatech.edu

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3__8,

example Box–Cox transformation) and the processes that radically change the nature, structure, domain, and dimension of data should be called transforms (for example Wigner–Ville transform).

In this chapter we focus mainly on transforms providing an additional insight on data. After the introduction discussing three examples, several important transforms are overviewed. We selected discrete Fourier, Hilbert, and Wigner–Ville transforms, discussed in Sect. 8.2, and given their recent popularity, continuous and discrete wavelet transforms discussed in Sects. 8.3 and 8.4.

8.1 Introduction

As an "appetizer" we give two simple examples of use of transformations in statistics, Fisher z and Box–Cox transformations as well as the empirical Fourier–Stieltjes transform.

Example 1. Assume that we are looking for variance transformation $Y = \vartheta(X)$, in the case where $\mathrm{Var}\, X = \sigma_X^2(\mu)$ is a function of the mean $\mu = \mathrm{E}\, X$. The first order Taylor expansion of $\vartheta(X)$ about mean μ is

$$\vartheta(X) = \vartheta(\mu) + (X - \mu)\vartheta'(\mu) + O\left[(X - \mu)^2\right] .$$

Ignoring quadratic and higher order terms we see that

$$\mathrm{E}\,\vartheta(X) \approx 0 , \quad \mathrm{Var}\,\vartheta(X) \approx \mathrm{E}\left[(X - \mu)^2\vartheta'(\mu)\right] = \left[\vartheta'(x)\right]^2 \sigma_X^2(\mu) .$$

If $\mathrm{Var}\,(\vartheta(X))$ is to be c^2, we obtain

$$\left[\vartheta'(x)\right]^2 \sigma_X^2(\mu) = c^2$$

resulting in

$$\vartheta(x) = c \int \frac{dx}{\sigma_X(x)}\, dx .$$

This is a theoretical basis for the so-called Fisher z-transformation.

Let $(X_{11}, X_{21}), \ldots, (X_{1n}, X_{2n})$ be a sample from bivariate normal distribution $N_2(\mu_1, \mu_2, \sigma_1^2, \sigma_2^2, \rho)$, and $\bar{X}_i = 1/n \sum_{j=1}^n X_{ij}$, $i = 1, 2$.

The Pearson coefficient of linear correlation

$$r = \frac{\sum_{i=1}^n (X_{1i} - \bar{X}_1)(X_{2i} - \bar{X}_2)}{\left[\sum_{i=1}^n \left(X_{1i} - \bar{X}_1\right)^2 \cdot \sum_{i=1}^n \left(X_{2i} - \bar{X}_2\right)^2\right]^{1/2}}$$

has a complicated distribution involving special functions, e.g., Anderson (1984, p. 113). However, it is well known that the asymptotic distribution for r is normal $N(\rho, \frac{(1-\rho^2)^2}{n})$. Since the variance is a function of mean,

$$\begin{aligned}\vartheta(\rho) &= \int \frac{c\sqrt{n}}{1-\rho^2} d\rho \\ &= \frac{c\sqrt{n}}{2} \int \left(\frac{1}{1-\rho} + \frac{1}{1+\rho} \right) d\rho \\ &= \frac{c\sqrt{n}}{2} \log \left(\frac{1+\rho}{1-\rho} \right) + k\end{aligned}$$

is known as Fisher z-transformation for the correlation coefficient (usually for $c = 1/\sqrt{n}$ and $k = 0$). Assume that r and ρ are mapped to z and ζ as

$$z = \frac{1}{2} \log \left(\frac{1+r}{1-r} \right) = \operatorname{arctanh} r \ , \quad \zeta = \frac{1}{2} \log \left(\frac{1+\rho}{1-\rho} \right) = \operatorname{arctanh} \rho \ .$$

The distribution of z is approximately normal $N(\zeta, 1/(n-3))$ and this approximation is quite accurate when ρ^2/n^2 is small and when n is as low as 20. The use of Fisher z-transformation is illustrated on finding the confidence intervals for ρ and testing hypotheses about ρ.

To exemplify the above, we generated $n = 30$ pairs of normally distributed random samples with theoretical correlation $\sqrt{2}/2$. This was done by generating two i.i.d. normal samples a, and b of length 30 and taking the transformation $x_1 = a + b$, $x_2 = b$. The sample correlation coefficient r is found. This was repeated $M = 10{,}000$ times. The histogram of 10,000 sample correlation coefficients is shown in Fig. 8.1a. The histogram of z-transformed r's is shown in Fig. 8.1b with superimposed normal approximation $N(\operatorname{arctanh}(\sqrt{2}/2), 1/(30-3))$.

(1) For example, $(1-\alpha)100\%$ confidence interval for ρ is:

$$\left[\tanh \left(z - \frac{\Phi^{-1}(1-\alpha/2)}{\sqrt{n-3}} \right), \tanh \left(z + \frac{\Phi^{-1}(1-\alpha/2)}{\sqrt{n-3}} \right) \right] ,$$

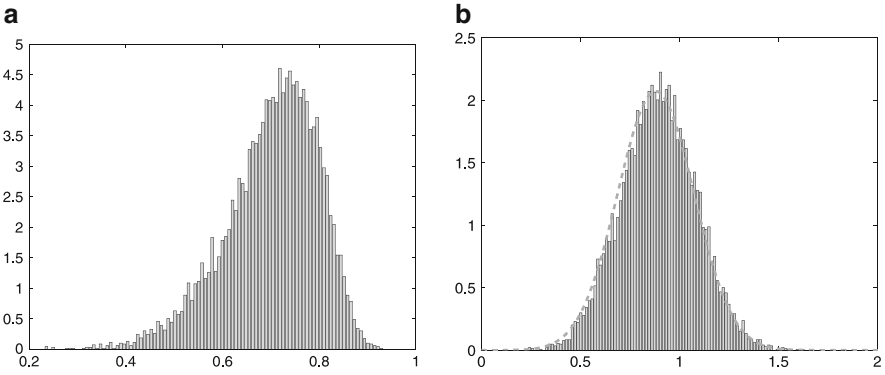


Fig. 8.1 (**a**) Simulational run of 10,000 r's from the bivariate population having theorethical $\rho = \sqrt{2}/2$.; (**b**) The same r's transformed to z's with the normal approximation superimposed

where $z = \operatorname{arctanh}(r)$ and $\tanh x = (e^x - e^{-x})/(e^x + e^{-x})$ and Φ stands for the standard normal cumulative distribution function.

If $r = -0.5687$ and $n = 28$ $z = -0.6456$, $z_L = -0.6456 - 1.96/5 = -1.0376$ and $z_U = -0.6456 + 1.96/5 = -0.2536$. In terms of ρ the 95% confidence interval is $[-0.7769, -0.2483]$.

(2) Assume that two samples of size n_1 and n_2, respectively, are obtained form two different bivariate normal populations. We are interested in testing $H_0 : \rho_1 = \rho_2$ against the two sided alternative. After observing r_1 and r_2 and transforming them to z_1 and z_2, we conclude that the p-value of the test is $2\Phi(-|z_1 - z_2|/\sqrt{1/(n_1 - 3) + 1/(n_2 - 3)})$.

Example 2. Box and Cox (1964) introduced a family of transformations, indexed by real parameter λ, applicable to positive data $X_1, \ldots, X_n$,

$$Y_i = \begin{cases} \dfrac{X_i^\lambda - 1}{\lambda}\,, & \lambda \neq 0 \\ \log X_i\,, & \lambda = 0\,. \end{cases} \tag{8.1}$$

This transformation is mostly applied to responses in linear models exhibiting non-normality and/or heteroscedasticity. For properly selected λ, transformed data $Y_1, \ldots, Y_n$ may look "more normal" and amenable to standard modeling techniques. The parameter λ is selected by maximizing the log-likelihood,

$$(\lambda - 1)\sum_{i=1}^{n} \log X_i - \frac{n}{2}\log\left[\frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \bar{Y}_i\right)^2\right]\,, \tag{8.2}$$

where Y_i are given in (8.1) and $\bar{Y}_i = 1/n \sum_{i=1}^{n} Y_i$.

As an illustration, we apply the Box–Cox transformation to apparently skewed data of CEO salaries.

Forbes magazine published data on the best small firms in 1993. These were firms with annual sales of more than five and less than \$350 million. Firms were ranked by five-year average return on investment. One of the variables extracted is the annual salary of the chief executive officer for the first 60 ranked firms (since one datum is missing, the sample size is 59). Figure 8.2a shows the histogram of row data (salaries). The data show moderate skeweness to the right. Figure 8.2b gives the values of likelihood in (8.2) for different values of λ. Note that (8.2) is maximized for λ approximately equal to 0.45. Figure 8.2c gives the transformed data by Box–Cox transformation with $\lambda = 0.45$. The histogram of transformed salaries is notably symetrized.

Example 3. As an example of transforms utilized in statistics, we provide an application of empirical Fourier–Stieltjes transform (empirical characteristic function) in testing for the independence.

The characteristic function of a probability distribution F is defined as its Fourier–Stieltjes transform,

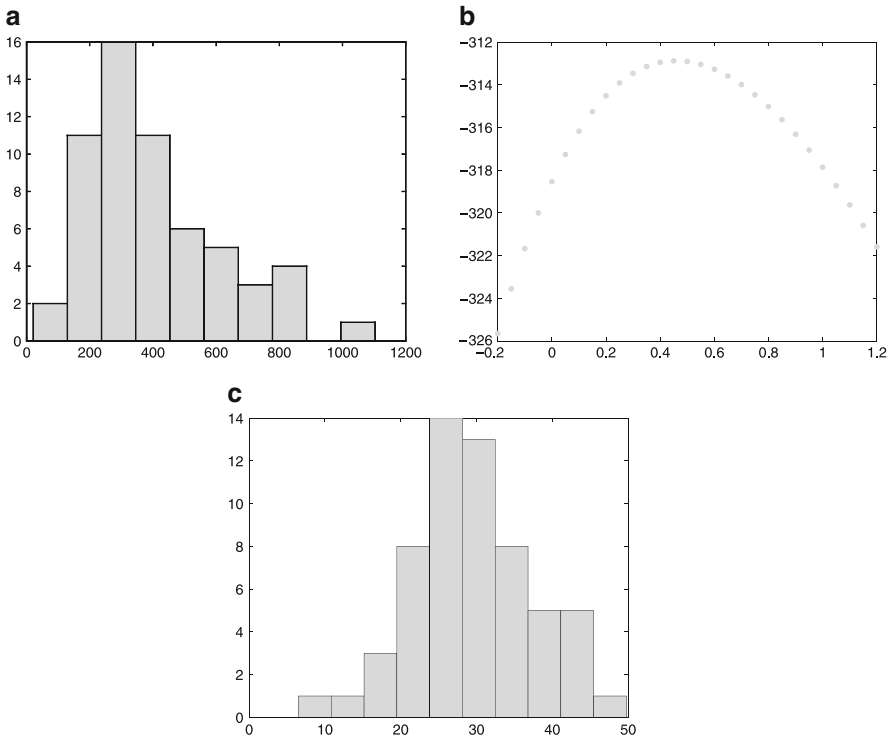


Fig. 8.2 (**a**) Histogram of row data (CEO salaries); (**b**) Log-likelihood is maximized at $\lambda = 0.45$; and (**c**) Histogram of Box–Cox-transformed data

$$\varphi_X(t) = \mathrm{E}\,\exp(\mathrm{i}tX)\ , \tag{8.3}$$

where E is expectation and random variable X has distribution function F. It is well known that the correspondence of characteristic functions and distribution functions is 1–1, and that closeness in the domain of characteristic functions corresponds to closeness in the domain of distribution functions. In addition to uniqueness, characteristic functions are bounded. The same does not hold for moment generating functions which are Laplace transforms of distribution functions.

For a sample $X_1, X_2, \ldots, X_n$ one defines empirical characteristic function $\varphi^*(t)$ as

$$\varphi_X^*(t) = \frac{1}{n}\sum_{j=1}^{n} \exp(itX_j)\ .$$

The result by Feuerverger and Mureika (1977) establishes the large sample properties of the empirical characteristic function.

Theorem 1. *For any* $T < \infty$

$$P\left[\lim_{n\to\infty} \sup_{|t|\le T} |\varphi^*(t) - \varphi(t)| = 0\right] = 1$$

holds. Moreover, when $n \to \infty$*, the stochastic process*

$$Y_n(t) = \sqrt{n}\left(\varphi^*(t) - \varphi(t)\right) \ , \quad |t| \le T \ ,$$

converges in distribution to a complex-valued Gaussian zero-mean process $Y(t)$ *satisfying* $Y(t) = \overline{Y(-t)}$ *and*

$$E\left(Y(t)\overline{Y(s)}\right) = \varphi(t+s) - \varphi(t)\varphi(s) \ ,$$

where $\overline{Y(t)}$ *denotes complex conjugate of* $Y(t)$.

Following Murata (2001) we describe how the empirical characteristic function can be used in testing for the independence of two components in bivariate distributions.

Given the bivariate sample (X_i, Y_i), $i = 1, \ldots, n$, we are interested in testing for independence of the components X and Y. The test can be based on the following bivariate process,

$$Z_n(t,s) = \sqrt{n}\left(\varphi^*_{X,Y}(t+s) - \varphi^*_X(t)\varphi^*_Y(s)\right) \ ,$$

where $\varphi^*_{X,Y}(t+s) = 1/n \sum_{j=1}^n \exp(\mathrm{i}tX_j + \mathrm{i}sY_j)$.

Murata (2001) shows that $Z_n(t,s)$ has Gaussian weak limit and that

$$\mathrm{Var}\, Z_n(t,s) \approx \left[\varphi^*_X(2t) - \left(\varphi^*_X(t)\right)^2\right]\left[\varphi^*_Y(2s) - \left(\varphi^*_Y(s)\right)^2\right] \ , \quad \text{and}$$

$$\mathrm{Cov}\left(Z_n(t,s), \overline{Z_n(t,s)}\right) \approx \left(1 - |\varphi^*_X(t)|^2\right)\left(1 - |\varphi^*_Y(s)|^2\right) \ ,$$

The statistics

$$T(t,s) = (\Re Z_n(t,s) \quad \Im Z_n(t,s))\ \Sigma^{-1}\ (\Re Z_n(t,s) \quad \Im Z_n(t,s))'$$

has approximately χ^2 distribution with 2 degrees of freedom for any t and s finite. Symbols $\Re$ and $\Im$ stand for the real and imaginary parts of a complex number. The matrix Σ is 2×2 matrix with entries

$$\varsigma_{11} = \frac{1}{2}\left[\Re \mathrm{Var}\ (Z_n(t,s)) + \mathrm{Cov}\left(Z_n(t,s), \overline{Z_n(t,s)}\right)\right]$$

$$\varsigma_{12} = \varsigma_{21} = \frac{1}{2}\Im \mathrm{Var}\left(Z_n(t,s)\right), \quad \text{and}$$

$$\varsigma_{22} = \frac{1}{2}\left[-\Re \mathrm{Var}\left(Z_n(t,s)\right) + \mathrm{Cov}\left(Z_n(t,s), \overline{Z_n(t,s)}\right)\right].$$

Any fixed pair t, s gives a valid test, and in the numerical example we selected $t = 1$ and $s = 1$ for calculational convenience.

We generated two independent components from the Beta$(1, 2)$ distribution of size $n = 2{,}000$ and found T statistics and corresponding p-values $M = 2{,}000$ times. Figure 8.3a,b depicts histograms of T statistics and p values based on 2,000 simulations. Since the generated components X and Y are independent, the histogram for T agrees with asymptotic χ^2_2 distribution, and of course, the p-values are uniform on $[0, 1]$. In Fig. 8.3c we show the p-values when the components X and Y are not independent. Using two independent Beta$(1, 2)$ components X and Y', the second component Y is constructed as $Y = 0.03X + 0.97Y'$. Notice that for

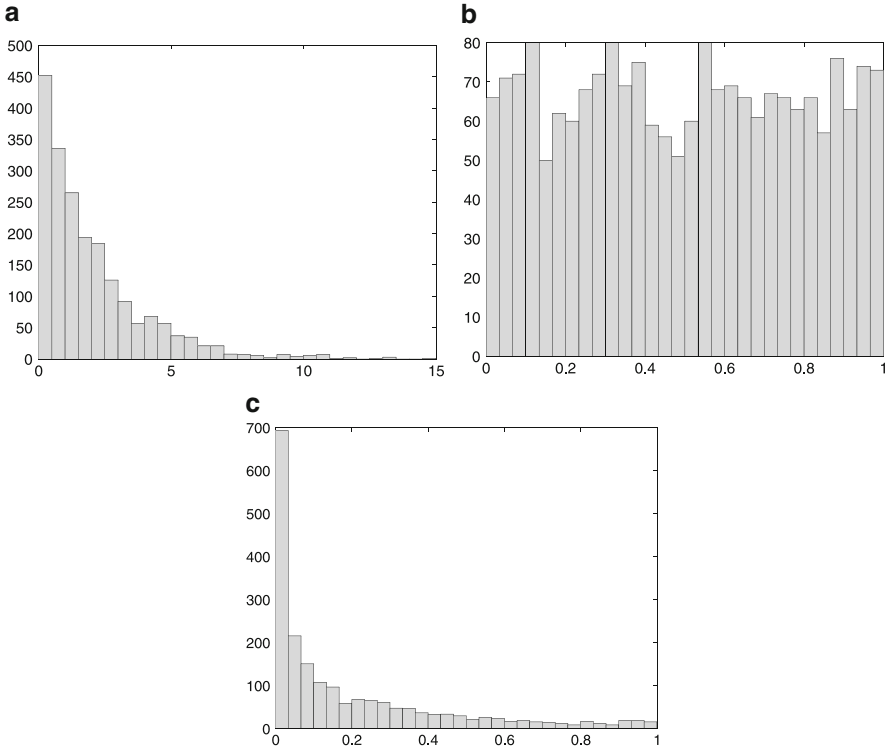


Fig. 8.3 (**a**) Histogram of observed T statistics with theoretical χ^2_2 distribution; (**b**) p-values of the test when components are independent; and (**c**) p-values if the test when the second component is a mixture of an independent sample and 3% of the first component

majority of simulational runs the independence hypothesis is rejected, i.e., the p-values cluster around 0.

8.2 Fourier and Related Transforms

Functional series have a long history that can be traced back to the early nineteenth century. French mathematician (and politician) Jean-Baptiste-Joseph Fourier, decomposed a continuous, periodic on $[-\pi, \pi]$ function $f(x)$ into the series od sines and cosines,

$$\frac{a_0}{2} + \sum_{n=1}^{\infty} a_n \cos nx + b_n \sin nx \ ,$$

where the coefficients a_n and b_n are defined as

$$a_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos nx \, dx \ , \quad n = 0, 1, 2, \ldots$$
$$b_n = \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin nx \, dx \ , \quad n = 1, 2, \ldots \ .$$

The sequences $\{a_n, \ n = 0, 1, \ldots\}$ and $\{b_n, \ n = 1, 2, \ldots\}$ can be viewed as a transform of the original function f. It is interesting that at the time of Fourier's discovery the very notion of function was not precisely defined. Fourier methods have long history in statistics especially in the theory of nonparametric function and density estimation and characteristic functions.

There are three types of Fourier transforms: integral, serial, and discrete. Next, we focus on discrete transforms and some modifications of the integral transform.

8.2.1 Discrete Fourier Transform

The discrete Fourier transform (DFT) of a sequence $\boldsymbol{f} = \{f_n, \ n = 0, 1, \ldots, N-1\}$ is defined as

$$\boldsymbol{F} = \left\{ \sum_{n=0}^{N-1} f_n w_N^{nk} \ , \quad k = 0, \ldots, N-1 \right\} \ ,$$

where $w_N = e^{-i2\pi/N}$. The inverse is

$$\boldsymbol{f} = \left\{ \frac{1}{N} \sum_{k=0}^{N-1} F_k w_N^{-nk} \ , \quad n = 0, \ldots, N-1 \right\} \ .$$

The DFT can be interpreted as the multiplication of the input vector by a matrix; therefore, the discrete Fourier transform is a linear operator. If $Q = \{Q_{nk} = \mathrm{e}^{-\mathrm{i}2\pi nk}\}_{N\times N}$, then $\boldsymbol{F} = Q \cdot \boldsymbol{f}$. The matrix Q is unitary (up to a scale factor), i.e., $Q^*Q = NI$, where I is the identity matrix and Q^* is the conjugate transpose of Q.

There are many uses of discrete Fourier transform in statistics. It turns cyclic convolutions into component-wise multiplication, and the fast version of DFT has a low computational complexity of $O(n\log(n))$, meaning that the number of operations needed to transform an input of size n is proportional to $n\log(n)$. For a theory and various other uses of DFT in various fields reader is directed to Brigham (1988).

We focus on estimation of a spectral density from the observed data, as an important statistical task in a variety of applied fields in which the information about frequency behavior of the phenomena is of interest.

Let $\{X_t, t \in Z\}$ be a a real, weakly stationary time series with zero mean and autocovariance function $\gamma(h) = EX(t+h)X(t)$. An absolutely summable complex-valued function $\gamma(\cdot)$ defined on integers is the autocovariance function of X_t if and only if the function

$$f(\omega) = \frac{1}{2\pi}\sum_{h=-\infty}^{\infty}\gamma(h)\mathrm{e}^{-\mathrm{i}h\omega} \tag{8.4}$$

is non-negative for all $\omega \in [-\pi, \pi]$. The function $f(\omega)$ is called the spectral density associated with covariance function $\gamma(h)$, and is in fact a version of discrete Fourier transform of the autocovariance function $\gamma(h)$. The spectral density of a stationary process is a symmetric and non-negative function. Given the spectral density, the autocovariance function can uniquely be recovered via inverse Fourier transform,

$$\gamma(h) = \int_{-\pi}^{\pi} f(\omega)\mathrm{e}^{\mathrm{i}h\omega}d\omega\ , \quad h = 0, \pm 1, \pm 2, \ldots\ .$$

A traditional statistic used as an estimator of the spectral density is the *periodogram*. The periodogram $I(\omega)$, based on a sample $X_0, \ldots, X_{T-1}$ is defined as

$$I(\omega_j) = \frac{1}{2\pi T}\left|\sum_{t=0}^{T-1} X_t\mathrm{e}^{-\mathrm{i}t\omega_j}\right|^2\ , \tag{8.5}$$

where ω_j is the Fourier frequency $\omega_j = 2\pi j/T$, $j = [-T/2] + 1, \ldots, -1, 0, 1, \ldots, [T/2]$. By a discrete version of the sampling theorem it holds that $I(\omega)$ is uniquely determined for all $\omega \in [-\pi, \pi]$, given its values at Fourier frequencies.

Calculationally, the periodogram is found by using fast Fourier transform. A simple `matlab` m-function calculating the periodogram is

```
function out = periodogram(ts)
out = abs(fftshift(fft(ts - mean(ts)))).^2/(2*pi*length(ts));
```

An application of spectral and log-spectral estimation involves famous Wolf's sunspot data set. Although in this situation the statistician does not know the "true" signal, the theory developed by solar scientists helps to evaluate performance of the algorithm.

The Sun's activity peaks every 11 years, creating storms on the surface of our star that disrupt the Earth's magnetic field. These "solar hurricanes" can cause severe problems for electricity transmission systems. An example of influence of such periodic activity to everyday life is 1989 power blackout in the American northeast.

Efforts to monitor the amount and variation of the Sun's activity by counting spots on it have a long and rich history. Relatively complete visual estimates of daily activity date back to 1818, monthly averages can be extrapolated back to 1,749, and estimates of annual values can be similarly determined back to 1,700. Although Galileo made observations of sunspot numbers in the early seventeenth century, the modern era of sunspot counting began in the mid-1,800s with the research of Bern Observatory director Rudolph Wolf, who introduced what he called the *Universal Sunspot Number* as an estimate of the solar activity. The square root of Wolf's yearly sunspot numbers are given in Fig. 8.4a, data from Tong (1996), p. 471. Because of wavelet data processing we selected a sample of size a power of two, i.e., only 256 observations from 1733 till 1998. The square root transformation was applied to symmetrize and de-trend the Wolf's counts. Figure 8.4b gives a raw periodogram, while Fig. 8.4c shows the estimator of log-spectral density (Pensky and Vidakovic 2003).

The estimator reveals a peak at frequency $\omega^* \approx 0.58$, corresponding to the Schwabe's cycle ranging from 9 to 11.5 (years), with an average of $2\pi/0.58 \approx 10.8$ years. The Schwabe cycle is the period between two subsequent maxima or minima the solar activity, although the solar physicists often think in terms of a 22-year magnetic cycle since the sun's magnetic poles reverse direction every 11 years.

8.2.2 Windowed Fourier Transform

Windowed Fourier Transforms are important in providing simultaneous insight in time and frequency behavior of the functions. Standard Fourier Transforms describing the data in the "Fourier domain" are precise in frequency, but not in time. Small changes in the signal (data) at one location cause change in the Fourier domain globally. It was of interest to have transformed domains that are simultaneously precise in both time and frequency domains. Unfortunately, the precision of such an insight is limited by the Heisenberg's Uncertainty Principle.

Suppose $f(t)$ is a signal of finite energy. In mathematical terms, the integral of its modulus squared is finite, or shortly, f belongs to $\mathbb{L}_2(\mathbb{R})$ space.

The integral Fourier transform of the signal

$$\mathcal{F}(f)(\xi) = \hat{f}(\xi) = \int_{\mathbb{R}} f(t)\mathrm{e}^{-\mathrm{i}t\xi}\,dt\,, \tag{8.6}$$

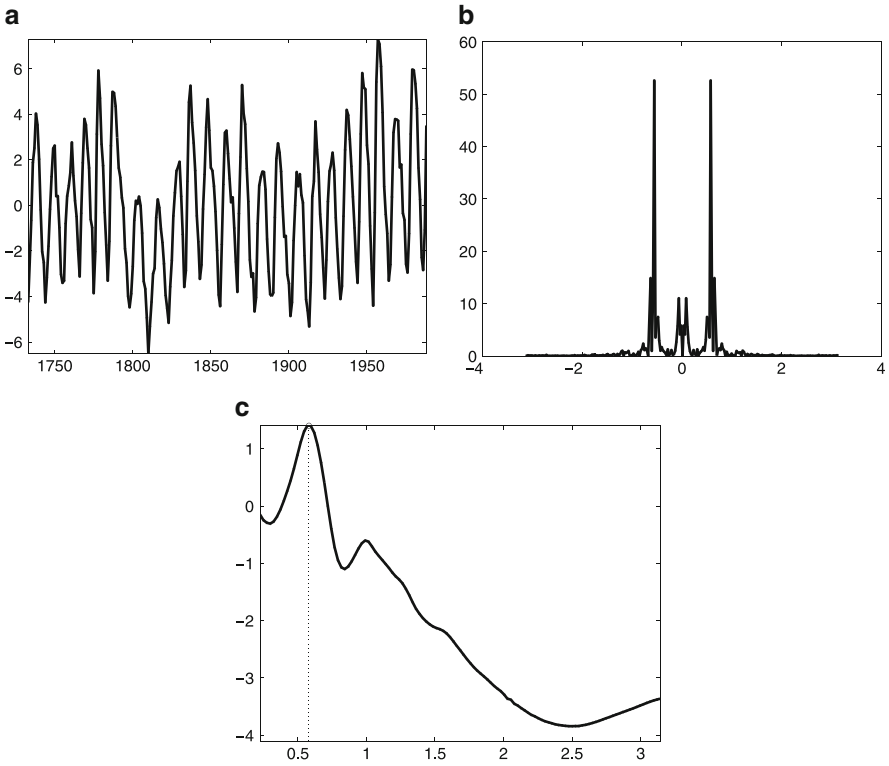


Fig. 8.4 (**a**) Square roots of Wolf's yearly sunspot numbers from 1732–1988 (256 observations); (**b**) Raw periodogram; (**c**) An estimator of the log-spectra. The frequency $\omega^* \approx 0.58$ corresponds to Schwabe's period of 10.8 (years)

describes the allocation of energy content of a signal at different frequencies, but the time-related information is lost.

Windowed Fourier transform (also called *short time Fourier transform,* STFT) was introduced by Gabor (1946), to measure time-localized frequencies of sound. An atom in Gabor's decomposition is defined via:

$$g_{u,\xi}(t) = \mathrm{e}^{\mathrm{i}\xi t} g(t-u) \ ,$$

where g is a real, symmetric, and properly normalized "window" function. [$||g|| = 1$ so that $||g_{u,\xi}|| = 1$]

If $f \in \mathbb{L}_2(\mathbb{R})$, then windowed Fourier transform is defined as

$$Sf(u,\xi) = \langle f, g_{u,\xi} \rangle = \int_{\mathbb{R}} f(t) g(t-u) \mathrm{e}^{-\mathrm{i}\xi t} dt \ . \tag{8.7}$$

The chief use of windowed Fourier transforms is to analyze time/frequency distribution of signal energy, via a *spectrogram.*

The spectrogram,

$$P_S f(u,\xi) = |Sf(u,\xi)|^2 = \left| \int_{-\infty}^{\infty} f(t) g(t-u) \mathrm{e}^{-\mathrm{i}\xi t} dt \right|^2 ,$$

expresses the energy distribution in the signal f, with respect to time and frequency simultaneously.

The following are some basic properties of STFT. Let $f \in \mathbb{L}_2(\mathbb{R}^2)$. Then

$$\text{[Inverse STFT]} \qquad f(t) = \frac{1}{2\pi} \int_{\mathbb{R}} \int_{\mathbb{R}} Sf(u,\xi) g(t-u) \mathrm{e}^{\mathrm{i}\xi t} d\xi du , \tag{8.8}$$

and

$$\text{[Energy Conservation]} \qquad \int_{\mathbb{R}} |f(t)|^2 dt = \frac{1}{2\pi} \int_{\mathbb{R}} \int_{\mathbb{R}} |Sf(u,\xi)|^2 d\xi du . \tag{8.9}$$

The following is a characterizing property of STFT:

Let $\Phi \in \mathbb{L}_2(\mathbb{R}^2)$. There exist $f \in \mathbb{L}_2(\mathbb{R}^2)$ such that $\Phi(u,\xi) = Sf(u,\xi)$ if and only if

$$\Phi(u_0,\xi_0) = \frac{1}{2\pi} \int_{\mathbb{R}} \int_{\mathbb{R}} \Phi(u,\xi) \mathbb{K}(u_0,u,\xi_0,\xi) \, du d\xi , \tag{8.10}$$

where

$$\mathbb{K}(u_0,u,\xi_0,\xi) = \langle g_{u,\xi}, g_{u_0,\xi_0} \rangle = \int_{\mathbb{R}} g(t-u) g(t-u_0) \mathrm{e}^{-\mathrm{i}(\xi_0-\xi)t} dt . \tag{8.11}$$

8.2.3 Hilbert Transform

We next describe the Hilbert transform and its use in defining instantaneous frequency, an important measure in statistical analysis of signals.

The Hilbert transform of the function signal $g(t)$ is defined by

$$H_g(t) = \frac{1}{\pi} (VP) \int_{-\infty}^{\infty} \frac{g(\tau)}{t-\tau} d\tau . \tag{8.12}$$

Because of the possible singularity at $\tau = t$, the integral is to be considered as a Cauchy principal value, (VP). From (8.12) we see that $H_g(t)$ is a convolution, $1/(\pi t) * g(t)$.

The spectrum of $H_g(t)$ is related to that of $g(t)$. From the convolution equation,

$$\mathcal{F}(H(t)) = \mathcal{F}\left(\frac{1}{\pi t}\right) \mathcal{F}(g(t)) .$$

where $\mathcal{F}$ is the Fourier transform. With a real signal $g(t)$ one can associate a complex function with the real part equal to $g(t)$ and the imaginary part equal to $H(g(t))$, $h(t) = g(t) - \mathrm{i}H(g(t))$.

In statistical signal analysis this associated complex function $h(t)$ is known as analytic signal (or causal signal, since $\hat{h}(\xi) = 0$, for $\xi < 0$). Analytic signals are important since they possess unique phase $\phi(t)$ which leads to the definition of the instantaneous frequency.

If $h(t)$ is represented as $a(t) \cdot \exp\{\mathrm{i}\phi(t)\}$, then the quantity $d\phi/dt$ is instantaneous frequency of the signal $g(t)$, at time t. For more discussion and use of instantaneous frequency, the reader is directed to Flandrin (1992, 1999).

8.2.4 Wigner–Ville Transforms

Wigner–Ville Transform (or Distribution) is the method to represent data (signals) in the time/frequency domain. In statistics, Wigner–Ville transform provide a tool to define localized spectral density for the nonstationary processes (Fig. 8.5).

Ville (1948) introduced the quadratic form that measures a local time-frequency energy:

$$P_V f(u,\xi) = \int_{\mathbb{R}} f\left(u + \frac{\tau}{2}\right) f^*\left(u - \frac{\tau}{2}\right) \mathrm{e}^{-\mathrm{i}\tau\xi} d\tau \ ,$$

where f^* is conjugate of f.

The Wigner–Ville transform is always real since $f(u + \frac{\tau}{2}) f^*(u - \frac{\tau}{2})$ has Hermitian symmetry in τ.

Time and frequency are symmetric in $P_V f(u,\xi)$, by applying Parseval formula one gets,

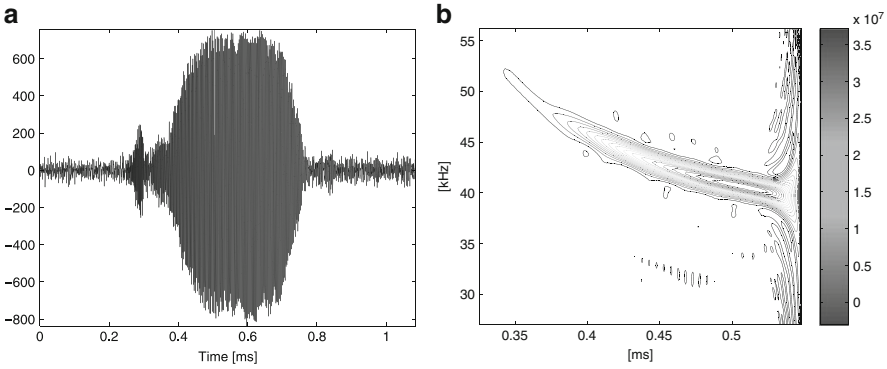


Fig. 8.5 (**a**) Sonar signal from flying bat; (**b**) its Wigner–Ville transform

$$P_V f(u,\xi) = \frac{1}{2\pi}\int_{\mathbb{R}} \hat{f}\left(\xi+\frac{\gamma}{2}\right)\hat{f}^*\left(\xi-\frac{\gamma}{2}\right)\mathrm{e}^{-\mathrm{i}\gamma u}d\gamma\ , \tag{8.13}$$

For any $f \in \mathbb{L}_2(\mathbb{R})$

$$\int_{\mathbb{R}} P_V f(u,\xi)du = |\hat{f}(\xi)|^2\ , \tag{8.14}$$

i.e., the time marginalization reproduces power spectrum, and

$$\int_{\mathbb{R}} P_V f(u,\xi)d\xi = 2\pi|f(u)|^2\ , \tag{8.15}$$

i.e, the frequency marginalization is proportional to the squared modulus of the signal.

Integral (8.13) states that one-dimensional Fourier transform of $g_\xi(u) = P_V f(u,\xi)$, with respect to u is,

$$\hat{g}_\xi(\gamma) = \hat{f}\left(\xi+\frac{\gamma}{2}\right)\hat{f}^*\left(\xi-\frac{\gamma}{2}\right)\ .$$

If $\gamma = 0$, $\hat{g}_\xi(0) = \int_{\mathbb{R}} g_\xi(u)du$, which proves (8.14). Similarly for (8.15).

For example:

(1) If $f(t) = \mathbf{1}(-T \le t \le T)$, then

$$P_V f(u,\xi) = \frac{2\sin[2(T-|u|)\xi]}{\xi}\mathbf{1}(-T \le u \le T)\ .$$

Plot $P_V f(u,\xi)$.

(2) If $f(t) = \exp\{\mathrm{i}\lambda(t+\alpha t^2/2)\}$, then $P_V(u,\xi) = 2\pi\delta(\xi-\lambda(1+\alpha u))$.

(3) A Gaussian $f(t) = (\sigma^2\pi)^{-1/4}\ \exp(-t^2/(2\sigma^2))$ is transformed into

$$P_V f(u,\xi) = \frac{1}{\pi}\exp\left(-\frac{u^2}{\sigma^2}-\sigma^2\xi^2\right)\ .$$

In this case, $P_V f(u,\xi) = |f(u)|^2\cdot|\hat{f}(\xi)|^2$. The Gaussian is the only (up to time and frequency shifts) distribution for which Wigner–Ville transform remains positive. Some basic properties of Wigner–Ville transforms are listed in Table 8.1.

Next we show that expected value of Wigner–Ville transform of a random process can serve as a definition for generalized spectrum of a non-stationary process. Let $X(t)$ be real-valued zero-mean random process with covariance function

$$EX(t)X(s) = R(t,s) = R\left(u+\frac{\tau}{2},u-\frac{\tau}{2}\right) = C(u,\tau)\ ,$$

Table 8.1 Properties of Wigner–Ville transform

Function	Wigner–Ville
$f(t)$	$P_V f(u,\xi)$
$e^{i\phi} f(t)$	$P_V f(u,\xi)$
$f(t-u_0)$	$P_V f(u-u_0,\xi)$
$e^{i\xi_0 t} f(t)$	$P_V f(u,\xi-\xi_0)$
$e^{iat^2} f(t)$	$P_V f(u,\xi-2au)$
$\frac{1}{\sqrt{s}} f(t/s)$	$P_V f(u/s, s\xi)$

after substitution $\tau = t - s$ and $u = (t + s)/2$.

Now, if the process $X(t)$ is stationary, then $C(u,\tau)$ is a function of τ only and

$$P_X(\xi) = \int_{-\infty}^{\infty} C(\tau) e^{-i\xi\tau} d\tau$$

is its power spectrum.

For arbitrary process Flandrin (1999) defined "power spectrum" as

$$P_X(\xi) = \int_{-\infty}^{\infty} C(u,\tau) e^{-i\xi\tau} d\tau \ .$$

Thus, $P_X(\xi)$ can be represented as $\mathrm{E}\, P_V X(u,\xi)$, where

$$P_V X(u,\xi) = \int_{-\infty}^{\infty} X\left(u+\frac{\tau}{2}\right) X\left(u-\frac{\tau}{2}\right) e^{-i\xi\tau} d\tau \ .$$

For more information on Wigner–Ville transforms and their statistical use the reader is directed to Baraniuk (1994), Carmona et al. (1998), Flandrin (1999), Mallat (1999), among others.

8.3 Wavelets and Other Multiscale Transforms

Given their recent popularity and clear evidence of wide applicability the most of the space in this chapter is devoted to Wavelet transforms. Statistical multiscale modeling has, in recent decade, become a well established area in both theoretical and applied statistics, with impact to developments in statistical methodology.

Wavelet-based methods are important in statistics in areas such as regression, density and function estimation, factor analysis, modeling and forecasting in time series analysis, in assessing self-similarity and fractality in data, in spatial statistics.

The attention of the statistical community was attracted in late 1980s and early 1990s, when Donoho, Johnstone, and their coauthors demonstrated that wavelet thresholding, a simple denoising procedure, had desirable statistical optimality properties. Since then, wavelets have proved useful in many statistical disciplines, notably in nonparametric statistics and time series analysis. Bayesian concepts and modeling approaches have, more recently, been identified as providing promising contexts for wavelet-based denoising applications.

In addition to replacing traditional orthonormal bases in a variety statistical problems, wavelets brought novel techniques and invigorated some of the existing ones.

8.3.1 A Case Study

We start first with a statistical application of wavelet transforms. This example emphasizes specificity of wavelet-based denoising not shared by standard state-of-art denoising techniques.

A researcher in geology was interested in predicting earthquakes by the level of water in nearby wells. She had a large ($8{,}192 = 2^{13}$ measurements) data set of water levels taken every hour in a period of time of about one year in a California well. Here is the description of the problem.

> The ability of water wells to act as strain meters has been observed for centuries. The Chinese, for example, have records of water flowing from wells prior to earthquakes. Lab studies indicate that a seismic slip occurs along a fault prior to rupture. Recent work has attempted to quantify this response, in an effort to use water wells as sensitive indicators of volumetric strain. If this is possible, water wells could aid in earthquake prediction by sensing precursory earthquake strain.
>
> We have water level records from six wells in southern California, collected over a six year time span. At least 13 moderate size earthquakes (magnitude 4.0–6.0) occurred in close proximity to the wells during this time interval. There is a significant amount of noise in the water level record which must first be filtered out. Environmental factors such as earth tides and atmospheric pressure create noise with frequencies ranging from seasonal to semidiurnal. The amount of rainfall also affects the water level, as do surface loading, pumping, recharge (such as an increase in water level due to irrigation), and sonic booms, to name a few. Once the noise is subtracted from the signal, the record can be analyzed for changes in water level, either an increase or a decrease depending upon whether the aquifer is experiencing a tensile or compressional volume strain, just prior to an earthquake.

A plot of the raw data for hourly measurements over one year ($8{,}192 = 2^{13}$ observations) is given in Fig. 8.6a, with a close-up in Fig. 8.6b. After applying the wavelet transform and further processing the wavelet coefficients (thresholding), we obtained a fairly clean signal with a big jump at the earthquake time. The wavelet-denoised data are given in Fig. 8.6d. The magnitude of the water level change at the earthquake time did not get distorted in contrast to traditional smoothing techniques. This local adaptivity is a desirable feature of wavelet methods.

For example, Fig. 8.6c, is denoised signal after applying `supsmo` smoothing procedure. Note that the earthquake jump is smoothed, as well.

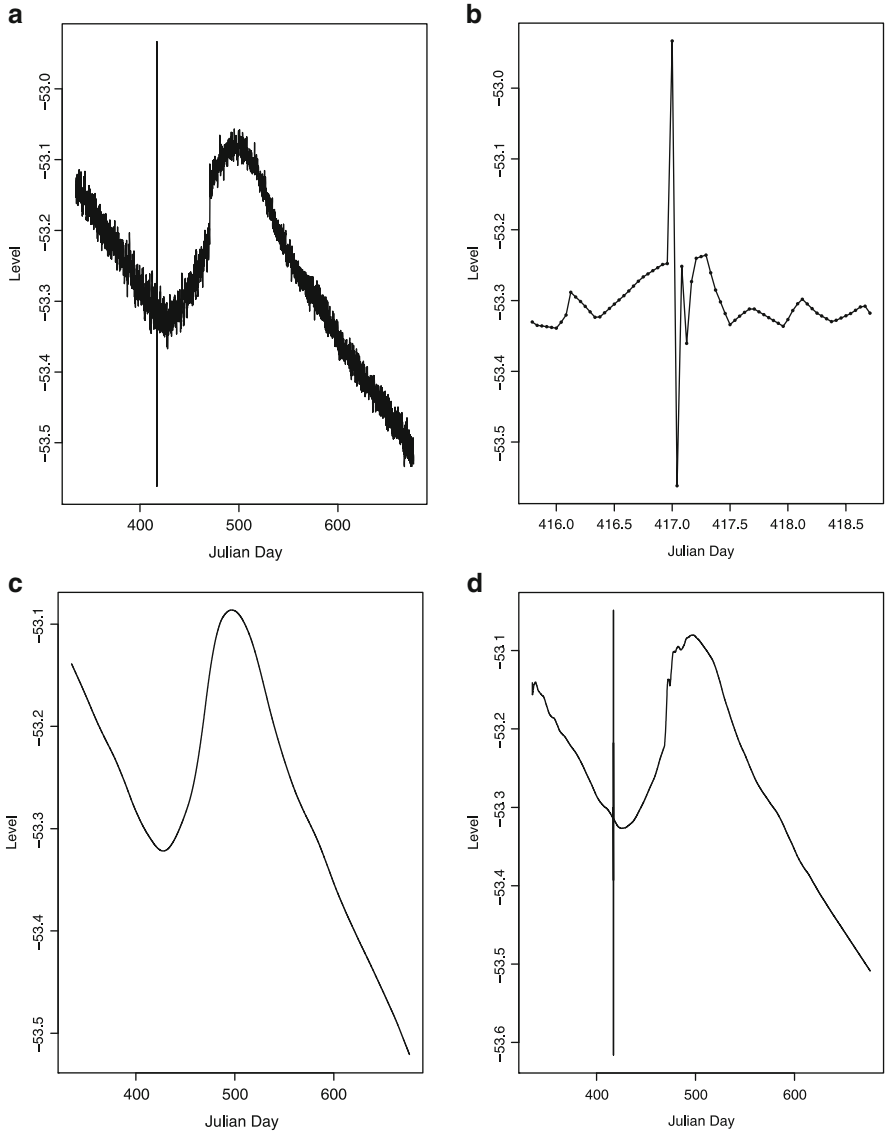


Fig. 8.6 (**a**) shows $n = 8{,}192$ hourly measurements of the water level for a well in an earthquake zone. Notice the wide range of water levels at the time of an earthquake around $t = 417$. (**b**) focusses on the data around the earthquake time. (**c**) demonstrates action of a standard smoother `supsmo`, and (**d**) depicts a wavelet based reconstruction

8.3.2 Continuous Wavelet Transform

The first theoretical results in wavelets had been concerned with continuous wavelet decompositions of functions and go back to the early 1980s. Papers of Morlet

et al. (1982) and Grossmann and Morlet (1984, 1985) were among the first on this subject.

Let $\psi_{a,b}(x)$, $a \in \mathbb{R}\backslash\{0\}, b \in \mathbb{R}$ be a family of functions defined as translations and re-scales of a single function $\psi(x) \in \mathbb{L}_2(\mathbb{R})$,

$$\psi_{a,b}(x) = \frac{1}{\sqrt{|a|}}\psi\left(\frac{x-b}{a}\right) . \tag{8.16}$$

Normalization constant $1/\sqrt{|a|}$ ensures that the norm $||\psi_{a,b}(x)||$ is independent of a and b. The function ψ (called *the wavelet function* is assumed to satisfy the *admissibility condition*,

$$C_\psi = \int_{\mathbb{R}} \frac{|\Psi(\omega)|^2}{|\omega|} d\omega < \infty , \tag{8.17}$$

where $\Psi(\omega) = \int_R \psi(x)\mathrm{e}^{-\mathrm{i}x\omega}dx$ is the Fourier transform of $\psi(x)$. The admissibility condition (8.17) implies

$$0 = \Psi(0) = \int \psi(x)dx .$$

Also, if $\int \psi(x)dx = 0$ and $\int(1 + |x|^\alpha)|\psi(x)|dx < \infty$ for some $\alpha > 0$, then $C_\psi < \infty$.

Wavelet functions are usually normalized to "have unit energy", i.e., $||\psi_{a,b}(x)|| = 1$.

For example, the second derivative of the Gaussian function,

$$\psi(x) = \frac{d^2}{dx^2}\left[-C\,\mathrm{e}^{-x^2/2}\right] = C\left(1-x^2\right)\mathrm{e}^{-x^2/2} , \quad C = \frac{2}{\sqrt{3\sqrt{\pi}}} ,$$

is an example of an admissible wavelet, called Mexican Hat or Marr's wavelet, see Fig. 8.7.

For any square integrable function $f(x)$, the continuous wavelet transform is defined as a function of two variables

$$\mathcal{CWT}_f(a,b) = \langle f, \psi_{a,b}\rangle = \int f(x)\overline{\psi_{a,b}(x)}dx .$$

Here the dilation and translation parameters, a and b, respectively, vary continuously over $\mathbb{R}\backslash\{0\} \times \mathbb{R}$.

Figure 8.8 gives the `doppler` test function, $f = 1/(t+0.05)\sqrt{t(1-t)}\sin(2\pi \cdot 1.05)$, $0 \le t \le 1$, and its continuous wavelet transform. The wavelet used was Mexican Hat. Notice the distribution of "energy" in the time/frequency plane in Fig. 8.8b.

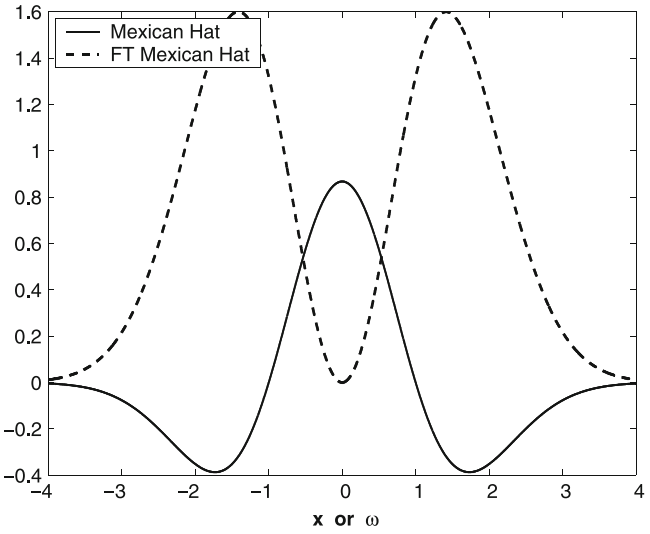


Fig. 8.7 Mexican hat wavelet (*solid line*) and its Fourier transform (*dashed line*)

Resolution of Identity

When the admissibility condition is satisfied, i.e., $C_\psi < \infty$, it is possible to find the inverse continuous transform via the relation known as *resolution of identity* or *Calderón's reproducing identity,*

$$f(x) = \frac{1}{C_\psi} \int_{\mathbb{R}^2} \mathcal{CWT}_f(a,b)\psi_{a,b}(x) \frac{da\, db}{a^2} \ .$$

The continuous wavelet transform of a function of one variable is a function of two variables. Clearly, the transform is redundant. To "minimize" the transform one can select discrete values of a and b and still have a lossless transform. This is achieved by so called *critical sampling.*

The critical sampling defined by

$$a = 2^{-j} \ , \quad b = k2^{-j} \ , \quad j,k \in \mathbb{Z} \ , \tag{8.18}$$

will produce the minimal, but complete basis. Any coarser sampling will not produce a unique inverse transform. Moreover under mild conditions on the wavelet function ψ, such sampling produces an orthogonal basis $\{\psi_{jk}(x) = 2^{j/2}\psi(2^j x - k), j,k \in \mathbb{Z}\}$. To formally describe properties of minimal and orthogonal wavelet bases a multiresolution formalism is needed.

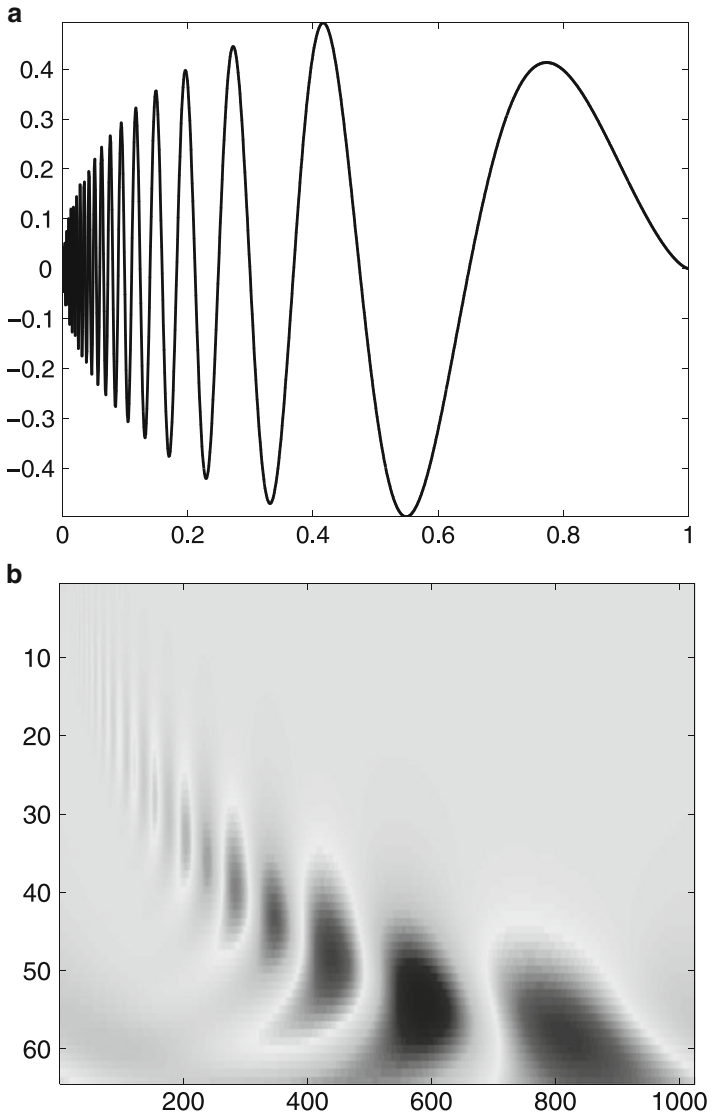


Fig. 8.8 (**a**) Doppler signal; (**b**) Continuous wavelet transform of `doppler` signal by the Mexican hat wavelet

8.3.3 Multiresolution Analysis

Fundamental for construction of critically sampled orthogonal wavelets is a notion of multiresolution analysis introduced by Mallat (1989a,b). A multiresolution analysis (MRA) is a sequence of closed subspaces $V_n, n \in \mathbb{Z}$ in $\mathbb{L}_2(\mathbb{R})$ such that they lie in a containment hierarchy

$$\cdots \subset V_{-2} \subset V_{-1} \subset V_0 \subset V_1 \subset V_2 \subset \cdots \,. \tag{8.19}$$

The nested spaces have an intersection that contains only the zero function and a union that contains all square integrable functions.

$$\cap_n V_j = \{\mathbf{0}\} \,, \quad \overline{\cup_j V_j} = \mathbb{L}_2(\mathbb{R}) \,.$$

(With $\overline{A}$ we denoted the closure of a set A). The hierarchy (8.19) is constructed such that V-spaces are self-similar,

$$f\left(2^j x\right) \in V_j \quad \text{iff} \quad f(x) \in V_0 \,. \tag{8.20}$$

with the requirement that there exists *a scaling function* $\phi \in V_0$ whose integer-translates span the space V_0,

$$V_0 = \left\{ f \in \mathbb{L}_2(\mathbb{R}) |\ \ f(x) = \sum_k c_k \phi(x-k) \right\} \,,$$

and for which the family $\{\phi(\bullet - k),\, k \in \mathbb{Z}\}$ is an orthonormal basis. It can be assumed that $\int \phi(x)dx \geq 0$. With this assumption this integral is in fact equal to 1. Because of containment $V_0 \subset V_1$, the function $\phi(x) \in V_0$ can be represented as a linear combination of functions from V_1, i.e.,

$$\phi(x) = \sum_{k \in \mathbb{Z}} h_k \sqrt{2} \phi(2x-k) \,, \tag{8.21}$$

for some coefficients h_k, $k \in \mathbb{Z}$. This equation called the *scaling equation* (or two-scale equation) is fundamental in constructing, exploring, and utilizing wavelets.

Theorem 2. *For the scaling function it holds*

$$\int_{\mathbb{R}} \phi(x)dx = 1 \,,$$

or, equivalently,

$$\Phi(0) = 1 \,,$$

where $\Phi(\omega)$ *is Fourier transform of* ϕ, $\int_{\mathbb{R}} \phi(x) \mathrm{e}^{-\mathrm{i}\omega x} dx$.

The coefficients h_n in (8.21) are important in efficient application of wavelet transforms. The (possibly infinite) vector $\boldsymbol{h} = \{h_n, n \in \mathbb{Z}\}$ will be called a *wavelet filter*. It is a low-pass (averaging) filter as will become clear later by its analysis in the Fourier domain.

To further explore properties of multiresolution analysis subspaces and their bases, we will often work in the Fourier domain.

It will be convenient to use Fourier domain for subsequent analysis of wavelet paradigm. Define the function m_0 as follows:

$$m_0(\omega) = \frac{1}{\sqrt{2}} \sum_{k \in \mathbb{Z}} h_k \mathrm{e}^{-\mathrm{i}k\omega} = \frac{1}{\sqrt{2}} H(\omega) \ . \tag{8.22}$$

The function in (8.22) is sometimes called the *transfer function* and it describes the behavior of the associated filter $\boldsymbol{h}$ in the Fourier domain. Notice that the function m_0 is 2π-periodic and that filter taps $\{h_n, n \in \mathbb{Z}\}$ are in fact the Fourier coefficients in the Fourier serias of $H(\omega) = \sqrt{2} m_0(\omega)$.

In the Fourier domain the relation (8.21) becomes

$$\Phi(\omega) = m_0\left(\frac{\omega}{2}\right) \Phi\left(\frac{\omega}{2}\right) \ , \tag{8.23}$$

where $\Phi(\omega)$ is the Fourier transform of $\phi(x)$. Indeed,

$$\begin{aligned}
\Phi(\omega) &= \int_{-\infty}^{\infty} \phi(x) \mathrm{e}^{-\mathrm{i}\omega x} dx \\
&= \sum_k \sqrt{2} h_k \int_{-\infty}^{\infty} \phi(2x-k) \mathrm{e}^{-\mathrm{i}\omega x} dx \\
&= \sum_k \frac{h_k}{\sqrt{2}} \mathrm{e}^{-\mathrm{i}k\omega/2} \int_{-\infty}^{\infty} \phi(2x-k) \mathrm{e}^{-\mathrm{i}(2x-k)\omega/2} d(2x-k) \\
&= \sum_k \frac{h_k}{\sqrt{2}} \mathrm{e}^{-\mathrm{i}k\omega/2} \Phi\left(\frac{\omega}{2}\right) \\
&= m_0\left(\frac{\omega}{2}\right) \Phi\left(\frac{\omega}{2}\right) \ .
\end{aligned}$$

By iterating (8.23), one gets

$$\Phi(\omega) = \prod_{n=1}^{\infty} m_0\left(\frac{\omega}{2^n}\right) \ , \tag{8.24}$$

which is convergent under very mild conditions concerning the rates of decay of the scaling function ϕ.

Next, we prove two important properties of wavelet filters associated with an orthogonal multiresolution analysis, *normalization* and *orthogonality.*

Normalization

$$\sum_{k\in\mathbb{Z}} h_k = \sqrt{2}\,. \tag{8.25}$$

Proof:

$$\begin{aligned}\int \phi(x)dx &= \sqrt{2}\sum_k h_k \int \phi(2x-k)dx \\ &= \sqrt{2}\sum_k h_k \frac{1}{2}\int \phi(2x-k)d(2x-k) \\ &= \frac{\sqrt{2}}{2}\sum_k h_k \int \phi(x)dx\,.\end{aligned}$$

Since $\int \phi(x)dx \neq 0$ by assumption, (8.25) follows.
This result also follows from $m_0(0) = 1$.

Orthogonality

For any $l \in \mathbb{Z}$,

$$\sum_k h_k h_{k-2l} = \delta_l\,. \tag{8.26}$$

Proof: Notice first that from the scaling equation (8.21) it follows that

$$\begin{aligned}\phi(x)\phi(x-l) &= \sqrt{2}\sum_k h_k \phi(2x-k)\phi(x-l) \\ &= \sqrt{2}\sum_k h_k \phi(2x-k)\sqrt{2}\sum_m h_m \phi(2(x-l)-m)\,.\end{aligned} \tag{8.27}$$

By integrating the both sides in (8.27) we obtain

$$\begin{aligned}\delta_l &= 2\sum_k h_k \left[\sum_m h_m \frac{1}{2}\int \phi(2x-k)\phi(2x-2l-m)\,d(2x)\right] \\ &= \sum_k \sum_m h_k h_m \delta_{k,2l+m} \\ &= \sum_k h_k h_{k-2l}\,.\end{aligned}$$

The last line is obtained by taking $k = 2l + m$.

An important special case is $l = 0$ for which (8.26) becomes

$$\sum_k h_k^2 = 1 \,. \tag{8.28}$$

The fact that the system $\{\phi(\bullet - k), k \in \mathbb{Z}\}$ constitutes an orthonormal basis for V_0 can be expressed in the Fourier domain in terms of either $\Phi(\omega)$ or $m_0(\omega)$.

In terms of $\Phi(\omega)$,

$$\sum_{l=-\infty}^{\infty} |\Phi(\omega + 2\pi l)|^2 = 1 \,. \tag{8.29}$$

From the Plancherel identity and the 2π-periodicity of $\mathrm{e}^{\mathrm{i}\omega k}$ it follows

$$\begin{aligned} \delta_k &= \int_{\mathbb{R}} \phi(x)\overline{\phi(x-k)}dx \\ &= \frac{1}{2\pi}\int_{\mathbb{R}} \Phi(\omega)\overline{\Phi(\omega)}\mathrm{e}^{\mathrm{i}\omega k}d\omega \\ &= \frac{1}{2\pi}\int_0^{2\pi} \sum_{l=-\infty}^{\infty} |\Phi(\omega + 2\pi l)|^2 \mathrm{e}^{\mathrm{i}\omega k}d\omega \,. \end{aligned} \tag{8.30}$$

The last line in (8.30) is the Fourier coefficient a_k in the Fourier series decomposition of

$$f(\omega) = \sum_{l=-\infty}^{\infty} |\Phi(\omega + 2\pi l)|^2 \,.$$

Due to the uniqueness of Fourier representation, $f(\omega) = 1$. As a side result, we obtain that $\Phi(2\pi n) = 0$, $n \neq 0$, and $\sum_n \phi(x - n) = 1$. The last result follows from inspection of coefficients c_k in the Fourier decomposition of $\sum_n \phi(x-n)$, the series $\sum_k c_k \mathrm{e}^{2\pi \mathrm{i} k x}$. As this function is 1-periodic,

$$c_k = \int_0^1 \left(\sum_n \phi(x-n)\right) \mathrm{e}^{-2\pi \mathrm{i} k x}dx = \int_{-\infty}^{\infty} \phi(x)\mathrm{e}^{-2\pi \mathrm{i} k x}dx = \Phi(2\pi k) = \delta_{0,k} \,.$$

Remark 1. Utilizing the identity (8.29), any set of independent functions spanning V_0, $\{\phi(x - k), k \in \mathbb{Z}\}$, can be orthogonalized in the Fourier domain. The orthonormal basis is generated by integer-shifts of the function

$$\mathcal{F}^{-1}\left[\frac{\Phi(\omega)}{\sqrt{\sum_{l=-\infty}^{\infty} |\Phi(\omega + 2\pi l)|^2}}\right] \,. \tag{8.31}$$

This normalization in the Fourier domain is used in constructing of some wavelet bases.

Orthogonality condition 8.29 can be expressed in terms of m_0 as:

$$|m_0(\omega)|^2 + |m_0(\omega + \pi)|^2 = 1 \,. \tag{8.32}$$

Since $\sum_{l=-\infty}^{\infty} |\Phi(2\omega + 2l\pi)|^2 = 1$, then by (8.23)

$$\sum_{l=-\infty}^{\infty} |m_0(\omega + l\pi)|^2 \, |\Phi(\omega + l\pi)|^2 = 1 \,. \tag{8.33}$$

Now split the sum in (8.33) into two sums – one with odd and the other with even indices, i.e.,

$$1 = \sum_{k=-\infty}^{\infty} |m_0(\omega + 2k\pi)|^2 \, |\Phi(\omega + 2k\pi)|^2 + \sum_{k=-\infty}^{\infty} |m_0(\omega + (2k+1)\pi)|^2 \, |\Phi(\omega + (2k+1)\pi)|^2 \,.$$

To simplify the above expression, we use (8.29) and the 2π-periodicity of $m_0(\omega)$.

$$\begin{aligned} 1 &= |m_0(\omega)|^2 \sum_{k=-\infty}^{\infty} |\Phi(\omega + 2k\pi)|^2 + |m_0(\omega + \pi)|^2 \sum_{k=-\infty}^{\infty} |\Phi((\omega + \pi) + 2k\pi)|^2 \\ &= |m_0(\omega)|^2 + |m_0(\omega + \pi)|^2 \,. \end{aligned}$$

Whenever a sequence of subspaces satisfies MRA properties, there exists (though not unique) an orthonormal basis for $\mathbb{L}_2(\mathbb{R})$,

$$\left\{\psi_{jk}(x) = 2^{j/2}\psi\left(2^j x - k\right) \,, \quad j, k \in \mathbb{Z}\right\} \tag{8.34}$$

such that $\{\psi_{jk}(x),\ j\text{-fixed}, k \in \mathbb{Z}\}$ is an orthonormal basis of the "difference space" $W_j = V_{j+1} \ominus V_j$. The function $\psi(x) = \psi_{00}(x)$ is called a *wavelet function* or informally *the mother wavelet*.

Next, we discuss the derivation of a wavelet function from the scaling function. Since $\psi(x) \in V_1$ (because of the containment $W_0 \subset V_1$), it can be represented as

$$\psi(x) = \sum_{k \in \mathbb{Z}} g_k \sqrt{2}\phi(2x - k) \,, \tag{8.35}$$

for some coefficients $g_k,\ k \in \mathbb{Z}$.

Define

$$m_1(\omega) = \frac{1}{\sqrt{2}} \sum_k g_k \mathrm{e}^{-\mathrm{i}k\omega} \,. \tag{8.36}$$

By mimicking what was done with m_0, we obtain the Fourier counterpart of (8.35),

$$\Psi(\omega) = m_1\left(\frac{\omega}{2}\right)\Phi\left(\frac{\omega}{2}\right) . \tag{8.37}$$

The spaces W_0 and V_0 are orthogonal by construction. Therefore,

$$\begin{aligned} 0 = \int \psi(x)\phi(x-k)dx &= \frac{1}{2\pi}\int \Psi(\omega)\overline{\Phi(\omega)}\mathrm{e}^{\mathrm{i}\omega k}d\omega \\ &= \frac{1}{2\pi}\int_0^{2\pi}\sum_{l=-\infty}^{\infty}\Psi(\omega+2l\pi)\overline{\Phi(\omega+2l\pi)}\mathrm{e}^{\mathrm{i}\omega k}d\omega . \end{aligned}$$

By repeating the Fourier series argument, as in (8.29), we conclude

$$\sum_{l=-\infty}^{\infty}\Psi(\omega+2l\pi)\overline{\Phi(\omega+2l\pi)} = 0 .$$

By taking into account the definitions of m_0 and m_1, and by the derivation as in (8.32), we find

$$m_1(\omega)\overline{m_0(\omega)} + m_1(\omega+\pi)\overline{m_0(\omega+\pi)} = 0 . \tag{8.38}$$

From (8.38), we conclude that there exists a function $\lambda(\omega)$ such that

$$(m_1(\omega), m_1(\omega+\pi)) = \lambda(\omega)\left(\overline{m_0(\omega+\pi)}, -\overline{m_0(\omega)}\right) . \tag{8.39}$$

By substituting $\xi = \omega + \pi$ and by using the 2π-periodicity of m_0 and m_1, we conclude that

$$\begin{aligned} \lambda(\omega) &= -\lambda(\omega+\pi) , \quad \text{and} \\ \lambda(\omega) &\quad \text{is } 2\pi\text{-periodic} . \end{aligned} \tag{8.40}$$

Any function $\lambda(\omega)$ of the form $\mathrm{e}^{\pm\mathrm{i}\omega}S(2\omega)$, where S is an $\mathbb{L}_2([0,2\pi])$, 2π-periodic function, will satisfy (8.38); however, only the functions for which $|\lambda(\omega)| = 1$ will define an orthogonal basis ψ_{jk} of $\mathbb{L}_2(\mathbb{R})$.

To summarize, we choose $\lambda(\omega)$ such that:

(1) $\lambda(\omega)$ is 2π-periodic,
(2) $\lambda(\omega) = -\lambda(\omega+\pi)$, and
(3) $|\lambda(\omega)|^2 = 1$.

Standard choices for $\lambda(\omega)$ are $-\mathrm{e}^{-\mathrm{i}\omega}$, $\mathrm{e}^{-\mathrm{i}\omega}$, and $\mathrm{e}^{\mathrm{i}\omega}$; however, any other function satisfying (1)–(3) will generate a valid m_1. We choose to define $m_1(\omega)$ as

$$m_1(\omega) = -\mathrm{e}^{-\mathrm{i}\omega}\overline{m_0(\omega+\pi)}\ . \tag{8.41}$$

since it leads to a convenient and standard connection between the filters $\boldsymbol{h}$ and $\boldsymbol{g}$.

The form of m_1 and (8.29) imply that $\{\psi(\bullet - k), k \in \mathbb{Z}\}$ is an orthonormal basis for W_0.

Since $|m_1(\omega)| = |m_0(\omega+\pi)|$, the orthogonality condition (8.32) can be rewritten as

$$|m_0(\omega)|^2 + |m_1(\omega)|^2 = 1\ . \tag{8.42}$$

By comparing the definition of m_1 in (8.36) with

$$\begin{aligned} m_1(\omega) &= -\mathrm{e}^{-\mathrm{i}\omega}\frac{1}{\sqrt{2}}\sum_k h_k \mathrm{e}^{\mathrm{i}(\omega+\pi)k} \\ &= \frac{1}{\sqrt{2}}\sum_k (-1)^{1-k} h_k \mathrm{e}^{-\mathrm{i}\omega(1-k)} \\ &= \frac{1}{\sqrt{2}}\sum_n (-1)^n h_{1-n}\mathrm{e}^{-\mathrm{i}\omega n}\ , \end{aligned}$$

we relate g_n and h_n as

$$g_n = (-1)^n\ h_{1-n}\ . \tag{8.43}$$

In signal processing literature, (8.43) is known as the *quadrature mirror relation* and the filters $\boldsymbol{h}$ and $\boldsymbol{g}$ as *quadrature mirror filters.*

Remark 2. Choosing $\lambda(\omega) = \mathrm{e}^{\mathrm{i}\omega}$ leads to the rarely used high-pass filter $g_n = (-1)^{n-1}h_{-1-n}$. It is convenient to define g_n as $(-1)^n h_{1-n+M}$, where M is a "shift constant." Such re-indexing of $\boldsymbol{g}$ affects only the shift-location of the wavelet function.

8.3.4 Haar Wavelets

In addition to their simplicity and formidable applicability, Haar wavelets have tremendous educational value. Here we illustrate some of the relations discussed in the Sect. 8.3.3 using the Haar wavelet. We start with scaling function $\phi(x) = \mathbf{1}(0 \le x \le 1)$ and pretend that everything else is unknown. By inspection of simple graphs of two scaled Haar wavelets $\phi(2x)$ and $\phi(2x+1)$ stuck to each other, we conclude that the scaling (8.21) is

$$\begin{aligned} \phi(x) &= \phi(2x) + \phi(2x-1) \\ &= \frac{1}{\sqrt{2}}\sqrt{2}\phi(2x) + \frac{1}{\sqrt{2}}\sqrt{2}\phi(2x-1)\ , \end{aligned} \tag{8.44}$$

which yields the wavelet filter coefficients:

$$h_0 = h_1 = \frac{1}{\sqrt{2}}\,.$$

The transfer functions are

$$m_0(\omega) = \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}\mathrm{e}^{-\mathrm{i}\omega 0}\right) + \frac{1}{\sqrt{2}}\left(\frac{1}{\sqrt{2}}\mathrm{e}^{-\mathrm{i}\omega 1}\right) = \frac{1+\mathrm{e}^{-\mathrm{i}\omega}}{2}\,.$$

and

$$m_1(\omega) = -\mathrm{e}^{-\mathrm{i}\omega}\overline{m_0(\omega+\pi)} = -\mathrm{e}^{-i\,\omega}\left(\frac{1}{2} - \frac{1}{2}\mathrm{e}^{\mathrm{i}\omega}\right) = \frac{1-\mathrm{e}^{-\mathrm{i}\omega}}{2}\,.$$

Notice that $m_0(\omega) = |m_0(\omega)|\mathrm{e}^{\mathrm{i}\varphi(\omega)} = \cos(\omega/2)\cdot \mathrm{e}^{-\mathrm{i}\omega/2}$ (after $\cos x = (\mathrm{e}^{\mathrm{i}x} + \mathrm{e}^{-\mathrm{i}x})/2$). Since $\varphi(\omega) = -\frac{\omega}{2}$, the Haar wavelet has *linear phase,* i.e., the scaling function is symmetric in the time domain. The orthogonality condition $|m_0(\omega)|^2 + |m_1(\omega)|^2 = 1$ is easily verified, as well.

Relation (8.37) becomes

$$\Psi(\omega) = \frac{1-\mathrm{e}^{-\mathrm{i}\omega/2}}{2}\Phi\left(\frac{\omega}{2}\right) = \frac{1}{2}\Phi\left(\frac{\omega}{2}\right) - \frac{1}{2}\Phi\left(\frac{\omega}{2}\right)\mathrm{e}^{-\mathrm{i}\omega/2}\,,$$

and by applying the inverse Fourier transform we obtain

$$\psi(x) = \phi(2x) - \phi(2x-1)$$

in the time-domain. Therefore we "have found" the Haar wavelet function ψ. From the expression for m_1 or by inspecting the representation of $\psi(x)$ by $\phi(2x)$ and $\phi(2x-1)$, we "conclude" that $g_0 = -g_{-1} = \frac{1}{\sqrt{2}}$.

Although the Haar wavelets are well localized in the time domain, in the frequency domain they decay at the slow rate of $O(1/n)$ and are not effective in approximating smooth functions.

8.3.5 *Daubechies' Wavelets*

The most important family of wavelets was discovered by Ingrid Daubechies and fully described in Daubechies (1992). This family is compactly supported with various degrees of smoothness.

The formal derivation of Daubechies' wavelets goes beyond the scope of this chapter, but the filter coefficients of some of its family members can be found by following considerations.

For example, to derive the filter taps of a wavelet with N vanishing moments, or equivalently, $2N$ filter taps, we use the following equations.

The normalization property of scaling function implies

$$\sum_{i=0}^{2N-1} h_i = \sqrt{2}\,,$$

requirement for vanishing moments for wavelet function ψ leads to

$$\sum_{i=0}^{2N-1} (-1)^i i^k h_i = 0\,, \quad k = 0, 1, \ldots, N-1\,,$$

and, finally, the orthogonality property can be expressed as

$$\sum_{i=0}^{2N-1} h_i h_{i+2k} = \delta_k \quad k = 0, 1, \ldots, N-1\,.$$

We obtained $2N + 1$ equations with $2N$ unknowns; however the system is solvable since the equations are not linearly independent.

Example 4. For $N = 2$, we obtain the system:

$$\begin{cases} h_0 + h_1 + h_2 + h_3 = \sqrt{2} \\ h_0^2 + h_1^2 + h_2^2 + h_3^2 = 1 \\ -h_1 + 2h_2 - 3h_3 = 0, \\ h_0\, h_2 + h_1\, h_3 = 0 \end{cases},$$

which has a solution $h_0 = \frac{1+\sqrt{3}}{4\sqrt{2}}$, $h_1 = \frac{3+\sqrt{3}}{4\sqrt{2}}$, $h_2 = \frac{3-\sqrt{3}}{4\sqrt{2}}$, and $h_3 = \frac{1-\sqrt{3}}{4\sqrt{2}}$.

For $N = 4$, the system is

$$\begin{cases} h_0 + h_1 + h_2 + h_3 + h_4 + h_5 + h_6 + h_7 = \sqrt{2} \\ h_0^2 + h_1^2 + h_2^2 + h_3^2 + h_4^2 + h_5^2 + h_6^2 + h_7^2 = 1 \\ h_0 - h_1 + h_2 - h_3 + h_4 - h_5 + h_6 - h_7 = 0 \\ h_0 h_2 + h_1 h_3 + h_2 h_4 + h_3 h_5 + h_4 h_6 + h_5 h_7 = 0 \\ h_0 h_4 + h_1 h_5 + h_2 h_6 + h_3 h_7 = 0 \\ h_0 h_6 + h_1 h_7 = 0 \\ 0h_0 - 1h_1 + 2h_2 - 3h_3 + 4h_4 - 5h_5 + 6h_6 - 7h_7 = 0 \\ 0h_0 - 1h_1 + 4h_2 - 9h_3 + 16h_4 - 25h_5 + 36h_6 - 49h_7 = 0 \\ 0h_0 - 1h_1 + 8h_2 - 27h_3 + 64h_4 - 125h_5 + 216h_6 - 343h_7 = 0\,. \end{cases}$$

Figure 8.9 depicts two scaling function and wavelet pairs from the Daubechies family. Figure 8.9a,b depict the pair with two vanishing moments, while Fig. 8.9c,d depict the pair with four vanishing moments.

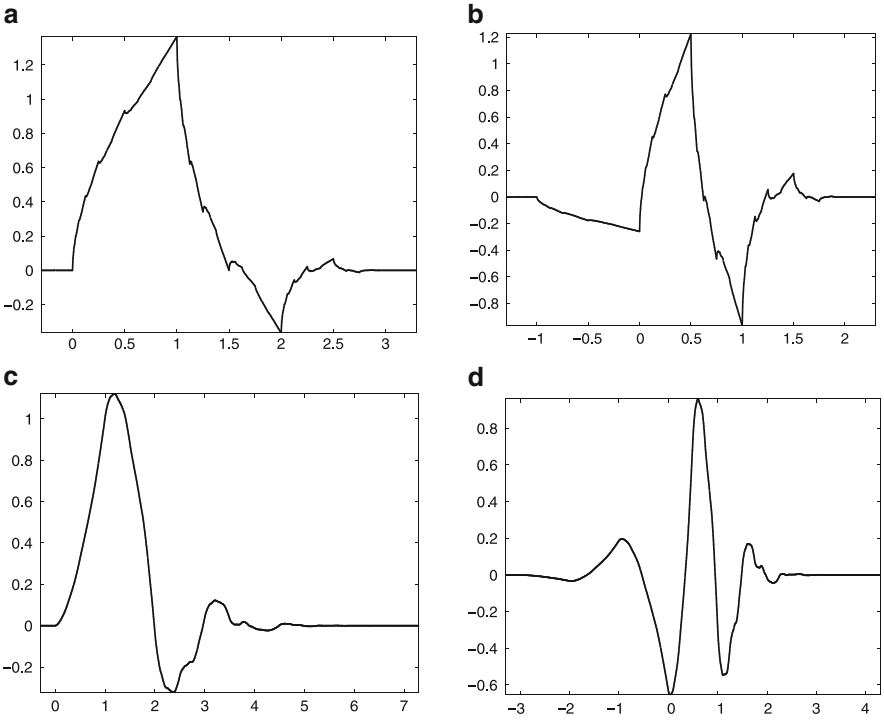


Fig. 8.9 Wavelet functions from Daubechies family. (**a**) Daubechies scaling function, 2 vanishing moments, 4 tap filter (**b**) Wavelet function corresponding to (**a**), (**c**) Daubechies scaling function, 4 vanishing moments, 8 tap filter (**d**) Wavelet function corresponding to (**c**)

8.4 Discrete Wavelet Transforms

Discrete wavelet transforms (DWT) are applied to discrete data sets and produce discrete outputs. Transforming signals and data vectors by DWT is a process that resembles the fast Fourier transform (FFT), the Fourier method applied to a set of discrete measurements.

The analogy between Fourier and wavelet methods is even more complete (Table 8.2) when we take into account the continuous wavelet transform and wavelet series expansions.

Discrete wavelet transforms map data from the time domain (the original or input data vector) to the wavelet domain. The result is a vector of the same size. Wavelet transforms are linear and they can be defined by matrices of dimension $n \times n$ if they are applied to inputs of size n. Depending on boundary conditions, such matrices can be either orthogonal or "close" to orthogonal. When the matrix is orthogonal, the corresponding transform is a rotation in $\mathbb{R}^n$ in which the data (a n-typle) is a point in $\mathbb{R}^n$. The coordinates of the point in the rotated space comprise the

Table 8.2 The analogy between Fourier and wavelet methods

Fourier	Fourier	Fourier	Discrete
Methods	Integrals	Series	Fourier transforms
Wavelet	Continuous	Wavelet	Discrete
Methods	Wavelet transforms	Series	Wavelet transforms

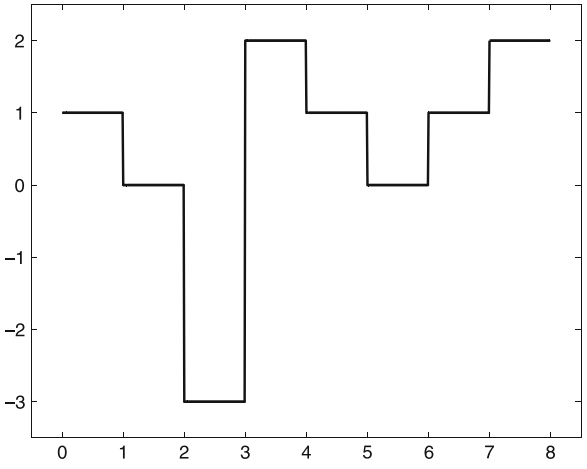


Fig. 8.10 A function interpolating $\boldsymbol{y}$ on $[0, 8)$

discrete wavelet transform of the original coordinates. Here we provide two toy examples.

Example 5. Let the vector be $(1, 2)$ and let $M(1, 2)$ be the point in $\mathbb{R}^2$ with coordinates given by the data vector. The rotation of the coordinate axes by an angle of $\pi/4$ can be interpreted as a DWT in the Haar wavelet basis. The rotation matrix is

$$W = \begin{pmatrix} \cos\frac{\pi}{4} & \sin\frac{\pi}{4} \\ \cos\frac{\pi}{4} & -\sin\frac{\pi}{4} \end{pmatrix} = \begin{pmatrix} \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} & -\frac{1}{\sqrt{2}} \end{pmatrix},$$

and the discrete wavelet transform of $(1, 2)'$ is $W \cdot (1, 2)' = (3/\sqrt{2}, -1/\sqrt{2})'$. Notice that *the energy* (squared distance of the point from the origin) is preserved, $1^2 + 2^2 = (1/2)^2 + (\sqrt{3}/2)^2$, since W is a rotation.

Example 6. Let $\boldsymbol{y} = (1, 0, -3, 2, 1, 0, 1, 2)$. The associated function f is given in Fig. 8.10. The values $f(n) = y_n$, $n = 0, 1, \ldots, 7$ are interpolated by a piecewise constant function. We assume that f belongs to Haar's multiresolution space V_0.

The following matrix equation gives the connection between $\boldsymbol{y}$ and the wavelet coefficients (data in the wavelet domain).

$$\begin{bmatrix} 1 \\ 0 \\ -3 \\ 2 \\ 1 \\ 0 \\ 1 \\ 2 \end{bmatrix} = \begin{bmatrix} \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2} & 0 & \frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & \frac{1}{2} & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 & 0 \\ \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2} & 0 & 0 & \frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{2\sqrt{2}} & \frac{1}{2\sqrt{2}} & -\frac{1}{2} & 0 & 0 & -\frac{1}{\sqrt{2}} & 0 & 0 \\ \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & 0 & \frac{1}{2} & 0 & 0 & \frac{1}{\sqrt{2}} & 0 \\ \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & 0 & \frac{1}{2} & 0 & 0 & -\frac{1}{\sqrt{2}} & 0 \\ \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & 0 & -\frac{1}{2} & 0 & 0 & 0 & \frac{1}{\sqrt{2}} \\ \frac{1}{2\sqrt{2}} & -\frac{1}{2\sqrt{2}} & 0 & -\frac{1}{2} & 0 & 0 & 0 & -\frac{1}{\sqrt{2}} \end{bmatrix} \cdot \begin{bmatrix} c_{00} \\ d_{00} \\ d_{10} \\ d_{11} \\ d_{20} \\ d_{21} \\ d_{22} \\ d_{23} \end{bmatrix} .$$

The solution is

$$\begin{bmatrix} c_{00} \\ d_{00} \\ d_{10} \\ d_{11} \\ d_{20} \\ d_{21} \\ d_{22} \\ d_{23} \end{bmatrix} = \begin{bmatrix} \sqrt{2} \\ -\sqrt{2} \\ 1 \\ -1 \\ \frac{1}{\sqrt{2}} \\ -\frac{5}{\sqrt{2}} \\ \frac{1}{\sqrt{2}} \\ -\frac{1}{\sqrt{2}} \end{bmatrix} .$$

Thus,

$$\begin{aligned} f = {} & \sqrt{2}\phi_{-3,0} - \sqrt{2}\psi_{-3,0} + \psi_{-2,0} - \psi_{-2,1} \\ & + \frac{1}{\sqrt{2}}\psi_{-1,0} - \frac{5}{\sqrt{2}}\psi_{-1,1} + \frac{1}{\sqrt{2}}\psi_{-1,2} - \frac{1}{\sqrt{2}}\psi_{-1,3} . \end{aligned} \tag{8.45}$$

The solution is easy to verify. For example, when $x \in [0, 1)$,

$$f(x) = \sqrt{2} \cdot \frac{1}{2\sqrt{2}} - \sqrt{2} \cdot \frac{1}{2\sqrt{2}} + 1 \cdot \frac{1}{2} + \frac{1}{\sqrt{2}} \cdot \frac{1}{\sqrt{2}} = \frac{1}{2} + \frac{1}{2} = 1 \, (= y_0) .$$

Applying wavelet transforms by multiplying the input vector with an appropriate orthogonal matrix is conceptually straightforward task, but of limited practical value. Storing and manipulating the transformation matrices for long inputs ($n >$ 2,000) may not even be feasible.

This obstacle is solved by the link of discrete wavelet transforms with fast filtering algorithms from the field of signal and image processing.

8.4.1 The Cascade Algorithm

Mallat (1989a,b) was the first to link wavelets, multiresolution analyses and cascade algorithms in a formal way. Mallat's cascade algorithm gives a constructive and efficient recipe for performing the discrete wavelet transform. It relates the wavelet coefficients from different levels in the transform by filtering with wavelet filter $\boldsymbol{h}$ and and its mirror counterpart $\boldsymbol{g}$.

It is convenient to link the original data with the space V_J, where J is often 0 or $\log n$, where n is a dyadic size of data. Then, coarser smooth and complementing detail spaces are (V_{J-1}, W_{J-1}), (V_{J-2}, W_{J-2}), etc. Decreasing the index in V-spaces is equivalent to coarsening the approximation to the data.

By a straightforward substitution of indices in the scaling (8.21) and (8.35), one obtains

$$\phi_{j-1,l}(x) = \sum_{k\in\mathbb{Z}} h_{k-2l}\phi_{jk}(x) \quad \text{and} \quad \psi_{j-1,l}(x) = \sum_{k\in\mathbb{Z}} g_{k-2l}\phi_{jk}(x)\ . \tag{8.46}$$

The relations in (8.46) are fundamental in developing the cascade algorithm.

In a multiresolution analysis, $\ldots \subset V_{j-1} \subset V_j \subset V_{j+1} \subset \ldots$. Since $V_j = V_{j-1} \oplus W_{j-1}$, any function $v_j \in V_j$ can be represented uniquely as $v_j(x) = v_{j-1}(x) + w_{j-1}(x)$, where $v_{j-1} \in V_{j-1}$ and $w_{j-1} \in W_{j-1}$. It is customary to denote the coefficients associated with $\phi_{jk}(x)$ and $\psi_{jk}(x)$ by c_{jk} and d_{jk}, respectively.

Thus,

$$\begin{aligned} v_j(x) &= \sum_k c_{j,k}\phi_{j,k}(x) \\ &= \sum_l c_{j-1,l}\phi_{j-1,l}(x) + \sum_l d_{j-1,l}\psi_{j-1,l}(x) \\ &= v_{j-1}(x) + w_{j-1}(x)\ . \end{aligned}$$

By using the general scaling (8.46), orthogonality of $w_{j-1}(x)$ and $\phi_{j-1,l}(x)$ for any j and l, and additivity of inner products, we obtain

$$\begin{aligned} c_{j-1,l} &= \langle v_j, \phi_{j-1,l}\rangle \\ &= \left\langle v_j, \sum_k h_{k-2l}\phi_{j,k} \right\rangle \\ &= \sum_k h_{k-2l}\langle v_j, \phi_{j,k}\rangle \\ &= \sum_k h_{k-2l}c_{j,k}\ . \end{aligned} \tag{8.47}$$

Similarly $d_{j-1,l} = \sum_k g_{k-2l} c_{j,k}$.

The cascade algorithm works in the reverse direction as well. Coefficients in the next finer scale corresponding to V_j can be obtained from the coefficients corresponding to V_{j-1} and W_{j-1}. The relation

$$\begin{aligned} c_{j,k} &= \langle v_j, \phi_{j,k} \rangle \\ &= \sum_l c_{j-1,l} \langle \phi_{j-1,l}, \phi_{j,k} \rangle + \sum_l d_{j-1,l} \langle \psi_{j-1,l}, \phi_{j,k} \rangle \\ &= \sum_l c_{j-1,l} h_{k-2l} + \sum_l d_{j-1,l} g_{k-2l} \ , \end{aligned} \tag{8.48}$$

describes a single step in the reconstruction algorithm.

The discrete wavelet transform can be described in terms of operators. Let the operators $\mathcal{H}$ and $\mathcal{G}$ acting on a sequence $a = \{a_n, n \in \mathbb{Z}\}$, satisfy the following coordinate-wise relations:

$$(\mathcal{H}a)_k = \sum_n h_{n-2k} a_n \quad (\mathcal{G}a)_k = \sum_n g_{n-2k} a_n \ ,$$

and their adjoint operators $\mathcal{H}^*$ and $\mathcal{G}^*$ satisfy:

$$(\mathcal{H}^*a)_n = \sum_k h_{n-2k} a_k \quad (\mathcal{G}^*a)_n = \sum_k g_{n-2k} a_k \ ,$$

where $\boldsymbol{h} = \{h_n\}$ is wavelet filter and $\boldsymbol{g} = \{g_n\}$ its quadrature-mirror counterpart.

Denote the original signal by $\boldsymbol{c}^{(J)} = \{c_k^{(J)}\}$. If the signal is of length 2^J, then $\boldsymbol{c}^{(J)}$ can be interpolated by the function $f(x) = \sum c_k^{(J)} \phi(x-k)$ from V_J. In each step of the wavelet transform, we move to the next coarser approximation (level) $\boldsymbol{c}^{(j-1)}$ by applying the operator $\mathcal{H}$, $\boldsymbol{c}^{(j-1)} = \mathcal{H}\boldsymbol{c}^{(j)}$. The "detail information," lost by approximating $\boldsymbol{c}^{(j)}$ by the "averaged" $\boldsymbol{c}^{(j-1)}$, is contained in vector $\boldsymbol{d}^{(j-1)} = \mathcal{G}\boldsymbol{c}^{(j)}$.

The discrete wavelet transform of a sequence $\boldsymbol{y} = \boldsymbol{c}^{(J)}$ of length 2^J can then be represented as

$$\left(\boldsymbol{c}^{(J-k)}, \boldsymbol{d}^{(J-k)}, \boldsymbol{d}^{(J-k+1)}, \ldots, \boldsymbol{d}^{(J-2)}, \boldsymbol{d}^{(J-1)}\right) \ . \tag{8.49}$$

Notice that the lengths of $\boldsymbol{y}$ and its transform in (8.49) coincide. Because of decimation, the length of $\boldsymbol{c}^{(j)}$ is twice the length of $\boldsymbol{c}^{(j-1)}$, and $2^J = 2^{J-k} + \sum_{i=1}^{k} 2^{J-i}$, $1 \le k \le J$.

For an illustration of (8.49), see Fig. 8.11. By utilizing the operator notation, it is possible to summarize the discrete wavelet transform (curtailed at level k) in a single line:

$$\boldsymbol{y} \mapsto \left(\mathcal{H}^k \boldsymbol{y}, \mathcal{G}\mathcal{H}^{k-1}\boldsymbol{y}, \ldots, \mathcal{G}\mathcal{H}^2\boldsymbol{y}, \mathcal{G}\mathcal{H}\boldsymbol{y}, \mathcal{G}\boldsymbol{y}\right) \ .$$

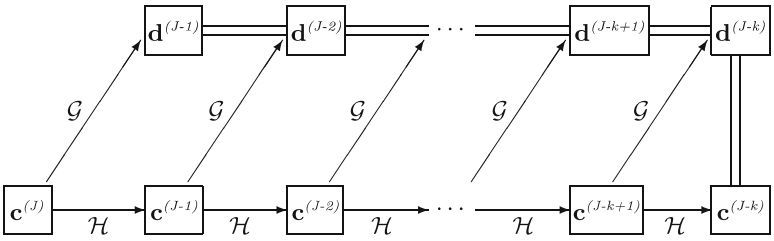


Fig. 8.11 Forward wavelet transform of depth k (DWT is a vector of coefficients connected by *double lines*)

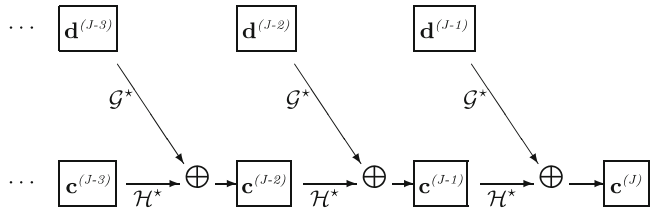


Fig. 8.12 Inverse transform

The number k can be any arbitrary integer between 1 and J and it is associated with the coarsest "smooth" space, V_{J-k}, up to which the transform was curtailed. In terms of multiresolution spaces, (8.49) corresponds to the multiresolution decomposition $V_{J-k} \oplus W_{J-k} \oplus W_{J-k+1} \oplus \ldots \oplus W_{J-1}$. When $k = J$ the vector $\boldsymbol{c}^{(0)}$ contains a single element, $c^{(0)}$.

If the wavelet filter length exceeds 2, one needs to define actions of the filter beyond the boundaries of the sequence to which the filter is applied. Different policies are possible. The most common is a periodic extension of the original signal.

The reconstruction formula is also simple in terms of operators $\mathcal{H}^*$ and $\mathcal{G}^*$. They are applied on $\boldsymbol{c}^{(j-1)}$ and $\boldsymbol{d}^{(j-1)}$, respectively, and the results are added (Fig. 8.12). The vector $\boldsymbol{c}^{(j)}$ is reconstructed as

$$\boldsymbol{c}^{(j)} = \mathcal{H}^* \boldsymbol{c}^{(j-1)} + \mathcal{G}^* \boldsymbol{d}^{(j-1)} , \tag{8.50}$$

Recursive application of (8.50) leads to

$$\begin{aligned} &\left(\mathcal{H}^k \boldsymbol{y}, \mathcal{G}\mathcal{H}^{k-1} \boldsymbol{y}, \ldots, \mathcal{G}\mathcal{H}^2 \boldsymbol{y}, \mathcal{G}\mathcal{H} \boldsymbol{y}, \mathcal{G} \boldsymbol{y}\right) \\ &\quad = \left(\boldsymbol{c}^{(J-k)}, \boldsymbol{d}^{(J-k)}, \boldsymbol{d}^{(J-k+1)}, \ldots, \boldsymbol{d}^{(J-2)}, \boldsymbol{d}^{(J-1)}\right) \\ &\quad \mapsto \sum_{i=1}^{k-1} \left(\mathcal{H}^*\right)^{k-1-i} \mathcal{G}^* \boldsymbol{d}^{(J-k+i)} + \left(\mathcal{H}^*\right)^k \boldsymbol{c}^{(J-k)} = \boldsymbol{y} \ . \end{aligned}$$

Example 7. Let $\boldsymbol{y} = (1, 0, -3, 2, 1, 0, 1, 2)$ be an exemplary set we want to transform by Haar's DWT. Let $k = J = 3$, i.e., the coarsest approximation and detail levels will contain a single point each. The decomposition algorithm applied on $\boldsymbol{y} = (1, 0, -3, 2, 1, 0, 1, 2)$ is given schematically in Fig. 8.13.

For the Haar wavelet, the operators $\mathcal{H}$ and $\mathcal{G}$ are given by $(\mathcal{H}a)_k = \sum_n h_{n-2k}a_n = \sum_m h_m a_{m+2k} = h_0 a_{2k} + h_1 a_{2k+1} = (a_{2k} + a_{2k+1})/\sqrt{2}$. Similarly, $(\mathcal{G}a)_k = \sum_n g_{n-2k}a_n = \sum_m g_m a_{m+2k} = g_0 a_{2k} + g_1 a_{2k+1} = (a_{2k} - a_{2k+1})/\sqrt{2}$.

The reconstruction algorithm is given in Fig. 8.14. In the process of reconstruction, $(\mathcal{H}^* a)_n = \sum_k h_{n-2k}a_k$, and $(\mathcal{G}^* a)_n = \sum_k g_{n-2k}a_k$. For instance, the first line in Fig. 8.14 recovers the object $\{1, 1\}$ from $\sqrt{2}$ by applying $\mathcal{H}^*$. Indeed, $(\mathcal{H}^*\{a_0\})_0 = h_0\sqrt{2} = 1$ and $(\mathcal{H}^*\{a_0\})_1 = h_1\sqrt{2} = 1$.

We already mentioned that when the length of the filter exceeds 2, boundary problems occur since the convolution goes outside the range of data.

There are several approaches to resolving the boundary problem. The signal may be continued in a periodic way $(\ldots, y_{n-1}, y_n | y_1, y_2, \ldots)$, symmetric way $(\ldots, y_{n-1}, y_n | y_{n-1}, y_{n-2}, \ldots)$, padded by a constant, or extrapolated as a polynomial. Wavelet transforms can be confined to an interval (in the sense of Cohen et al. (1993) and periodic and symmetric extensions can be viewed as special cases. Periodized wavelet

transforms are also defined in a simple way.

If the length of the data set is not a power of 2, but of the form $M \cdot 2^K$, for M odd and K a positive integer, then only K steps in the decomposition algorithm can be

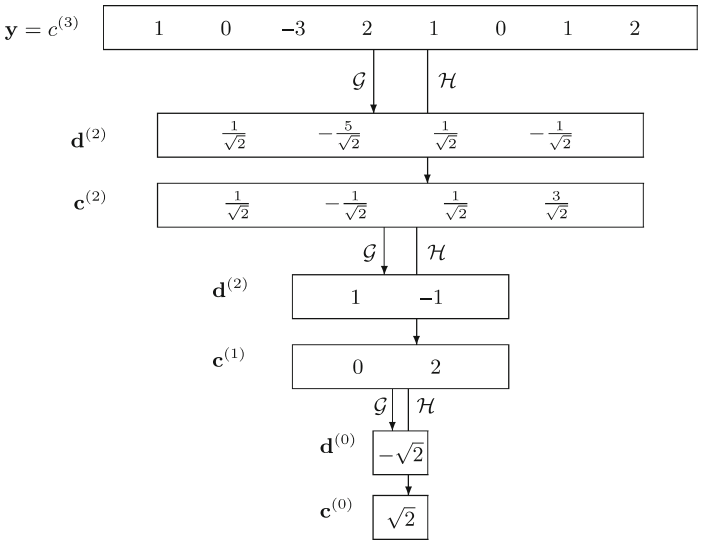


Fig. 8.13 An illustration of a decomposition procedure

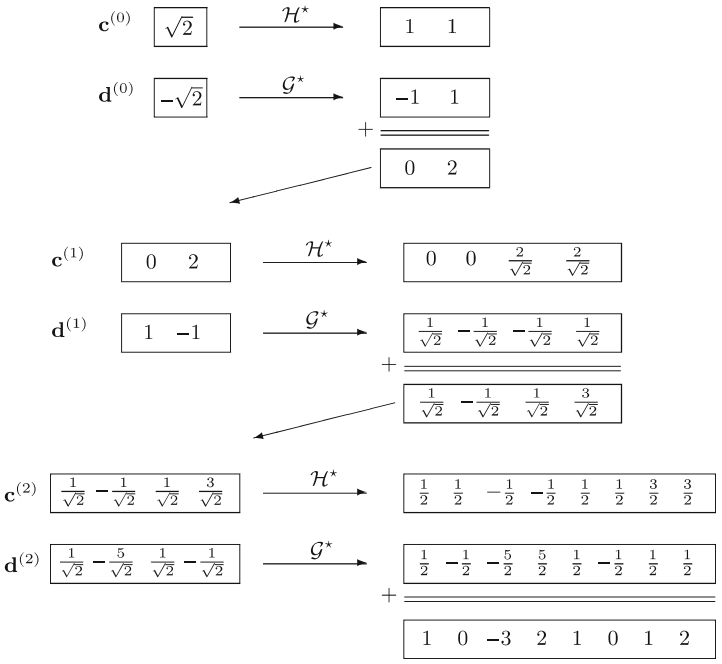


Fig. 8.14 An illustration of a reconstruction procedure

performed. For precise descriptions of conceptual and calculational hurdles caused by boundaries and data sets whose lengths are not a power of 2, we direct the reader to the monograph by Wickerhauser (1994).

In this section we discussed the most basic wavelet transform. Various generalizations include biorthogonal wavelets, multiwavelets, nonseparable multidimensional wavelet transforms, complex wavelets, lazy wavelets, and many more.

For various statistical applications of wavelets (nonparametric regression, density estimation, time series, deconvolutions, etc.) we direct the reader to Antoniadis (1997), Hï£¡rdle et al. (1998), Vidakovic (1999). An excellent monograph by Walter and Shen (2000) discusses statistical applications of wavelets and various other orthogonal systems.

8.4.2 Matlab Implementation of Cascade Algorithm

The following two `matlab` m-files implement discrete wavelet transform and its inverse, with periodic handling of boundaries. The data needs to be of dyadic size (power of 2). The programs are didactic, rather than efficient. For an excellent and

comprehensive wavelet package, we direct the reader to `wavelab802` module (http://www-stat.stanford.edu/~wavelab/) maintained by Donoho and his coauthors.

```
function dwtr = dwtr(data, L, filterh)
%  function dwtr = dwt(data, filterh, L);
%  Calculates the DWT of periodic data set
%  with scaling filter  filterh  and  L  scales.
%
%   Example of Use:
%   data = [1 0 -3 2 1 0 1 2]; filter = [sqrt(2)/2 sqrt(2)/2];
%   wt = DWTR(data, 3, filter)
%-------------------------------------------------------------------

n = length(filterh);                %Length of wavelet filter
C = data;                           %Data \qut{live} in V_J
dwtr = [];                          %At the beginning dwtr empty
H  =  fliplr(filterh);              %Flip because of convolution
G  =  filterh;                      %Make quadrature mirror
G(1:2:n) = -G(1:2:n);               %    counterpart
for j = 1:L                         %Start cascade
  nn = length(C);                   %Length needed to
  C = [C(mod((-(n-1):-1),nn)+1) C]; % make periodic
  D = conv(C,G);                    %Convolve,
  D = D([n:2:(n+nn-2)]+1);          %  keep periodic, decimate
  C = conv(C,H);                    %Convolve,
  C = C([n:2:(n+nn-2)]+1);          %  keep periodic, decimate
  dwtr = [D,dwtr];                  %Add detail level to dwtr
end;                                %Back to cascade or end
dwtr = [C, dwtr];                   %Add the last \qut{smooth} part
```

```
function  data = idwtr(wtr, L, filterh)
% function data = idwt(wtr, L, filterh);
% Calculates the IDWT of wavelet
% transform wtr using wavelet filter
% \qut{filterh}  and  L  scales.
% Example:
%>> max(abs(data - IDWTR(DWTR(data,3,filter), 3,filter)))
%ans = 4.4409e-016
%-------------------------------------------------------------------
nn = length(wtr);   n = length(filterh); %Lengths
if nargin==2, L = round(log2(nn)); end;  %Depth of transform
H = filterh;                             %Wavelet H filter
G = fliplr(H); G(2:2:n) = -G(2:2:n);     %Wavelet G filter
LL = nn/(2^L);                           %Number of scaling coeffs
C =  wtr(1:LL);                          %Scaling coeffs
for j = 1:L                              %Cascade algorithm
   w  = mod(0:n/2-1,LL)+1;               %Make periodic
   D  = wtr(LL+1:2*LL);                  %Wavelet coeffs
   Cu(1:2:2*LL+n) = [C C(1,w)];          %Upsample & keep periodic
   Du(1:2:2*LL+n) = [D D(1,w)];          %Upsample & keep periodic
   C  = conv(Cu,H) + conv(Du,G);         %Convolve & add
   C  = C([n:n+2*LL-1]-1);               %Periodic part
   LL = 2*LL;                            %Double the size of level
end;
data = C;                                %The inverse DWT
```

8.5 Conclusion

In this chapter we gave an overview of several transforms useful in computational statistics. We emphasized frequency and scale domain transforms (Fourier and wavelet) since they provide an insight to the phenomena, not available in the

domain of untransformed data. Moreover, multiscale transforms are relatively new, and as such deserve more attention. It was pretentious to title this chapter *Transforms in Statistics,* since literally several dozens important transforms are not even mentioned. As it was hinted in the introduction, a just task of overviewing all important transformations used in statistical practice would take a space of a large monograph.

Acknowledgements Work on this chapter was supported by DOD/NSA Grant E-24-60R at Georgia Institute of Technology. Editor Jim Gentle read early versions of the chapter and gave many valuable comments. All `matlab` programs that produced figures and simulations are available from the author at request.

References

Anderson, T.W.: An Introduction to Multivariate Statistical Analysis, (2nd edn.), Wiley, New York (1984)

Antoniadis, A.: Wavelets in statistics: A review. J. Ital. Stat. Soc. **6**, 97–144 (1997)

Baraniuk, R.G.: Wigner–Ville spectrum estimation via wavelet soft–tresholding. In: *Proceedings of IEEE-SP International Symposium on Time-Frequency and Time-Scale Analysis*, Philadelphia, PA, USA (1994)

Box, G.E.P., Cox, D.R.: An analysis of transformations, J. Roy. Stat. Soc. **26**, 211–243 discussion 244–252 (1964)

Brigham, E.O.: The Fast Fourier Transform and Its Applications, Prentice-Hall, Englewood Cliffs, NJ (1988)

Carmona, R., Hwang, W-L., Torrésani, B.: Practical Time–Frequency Analysis, Wavelet Analysis and its Applications, vol. 9, Academic Press, San Diego (1998)

Cohen, A, Daubechies, I., Vial, P.: Wavelets on the interval and fast wavelet transforms. Appl. Comput. Harmon. Anal. **1**(1), 54–81 (1993)

Daubechies, I.: Ten Lectures on Wavelets, Number 61 in CBMS-NSF Series in Applied Mathematics, SIAM, Philadelphia (1992)

Feuerverger, A., Mureika, R.: The empirical characteristic function and its applications, Ann. Stat. **5**, 88–97 (1977)

Flandrin, P.: Time-scale analyses and self-similar stochastic processes. In: Byrnes et al. (eds.) Wavelets and Their Applications, vol. 442, pp. 121–142. NATO ASI Series (1992)

Flandrin, P.: Time-Frequency/Time-scale Analysis, p. 386. Academic Press, New York (Orlando.FL/London) (1999)

Gabor, D.: Theory of comunication. J. IEEE **93**, 429–457 (1946)

Grossmann, A., Morlet, J.: Decomposition of Hardy functions into square integrable wavelets of constant shape. SIAM J. Math. **15**, 723–736 (1984)

Grossmann, A., Morlet, J.: Decomposition of functions into wavelets of constant shape and related transforms. In: Streit, L. (eds.) Mathematics and physics, lectures on recent results, World Scientific, River Edge, NJ (1985)

Hï£¡rdle, W., Kerkyacharian, G., Pickard, D., Tsybakov, A.: Wavelets, Approximation, and Statistical Applications, Lecture Notes in Statistics 129. Springer, New York (1998)

Mallat, S.G.: Multiresolution approximations and wavelet orthonormal bases of $\mathbb{L}^2(\mathbb{R})$. Trans. Amer. Math. Soc. **315**, 69–87 (1989a)

Mallat, S.G.: A theory for multiresolution signal decomposition: The wavelet representation. IEEE Trans. on Patt. Anal. Mach. Intell. **11**(7), 674–693 (1989b)

Mallat, S.G.: A Wavelet Tour of Signal Processing, (2nd edn.), Academic Press, San Diego (1999)

Morlet, J., Arens, G., Fourgeau, E., Giard, D.: Wave propagation and sampling theory. Geophys. **47**, 203–236 (1982)

Murata, N.: Properties of the empirical characteristic function and its application to testing for independence. In: Lee, Jung, Makeig, Sejnowski (eds.) Proceedings ICA2001, 3rd International Conference on Independent Component Analysis, San Diego, CA, USA (2001)

Pensky, M., Vidakovic, B., De Canditiis, D.: Bayesian decision theoretic scale-adaptive estimation of log-spectral density. Statistica Sinica **17**, 635–666 (2007)

Tong, H.: Non-Linear Time Series, Clarendon Press, Oxford (1996)

Vidakovic, B.: Statistical Modeling by Wavelets, Wiley, NY (1999)

Ville, J.: Théorie et applications de la notion de signal analytique. Cables et Transm. **2A**, 61–74 (1948)

Walter, G.G., Shen, X.: Wavelets and Other Orthogonal Systems, (2nd edn.), CRC Press (2000)

Wickerhauser, M. V.: Adapted Wavelet Analysis from Theory to Software, A K Peters, Ltd., Wellesley, MA (1994)

Chapter 9
Parallel Computing Techniques

Junji Nakano

9.1 Introduction

Parallel computing means to divide a job into several tasks and use more than one processor simultaneously to perform these tasks. Assume you have developed a new estimation method for the parameters of a complicated statistical model. After you prove the asymptotic characteristics of the method (for instance, asymptotic distribution of the estimator), you wish to perform many simulations to assure the goodness of the method for reasonable numbers of data values and for different values of parameters. You must generate simulated data, for example, 100,000 times for each length and parameter value. The total simulation work requires a huge number of random number generations and takes a long time on your PC. If you use 100 PCs in your institute to run these simulations simultaneously, you may expect that the total execution time will be 1/100. This is the simple idea of parallel computing.

Computer scientists noticed the importance of parallel computing many years ago (Flynn 1966). It is true that the recent development of computer hardware has been very rapid. Over roughly 40 years from 1961, the so called "Moore's law" holds: the number of transistors per silicon chip has doubled approximately every 18 months (Tuomi 2002). This means that the capacity of memory chips and processor speeds have also increased roughly exponentially. In addition, hard disk capacity has increased dramatically. Consequently, modern personal computers are more powerful than "super computers" were a decade ago. Unfortunately, even such powerful personal computers are not sufficient for our requirements. In statistical analysis, for example, while computers are becoming more powerful,

J. Nakano (✉)
Department of Data Science, The Institute of Statistical Mathematics
Tachikawa, Tokyo, Japan
e-mail: nakanoj@ism.ac.jp

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_9,

data volumes are becoming larger and statistical techniques are becoming more computer intensive. We are continuously forced to realize more powerful computing environments for statistical analysis. Parallel computing is thought to be the most promising technique.

However, parallel computing has not been popular among statisticians until recently (Schervish 1988). One reason is that parallel computing was available only on very expensive computers, which were installed at some computer centers in universities or research institutes. Few statisticians could use these systems easily. Further, software for parallel computing was not well prepared for general use.

Recently, cheap and powerful personal computers changed this situation. The Beowulf project (Sterling et al. 1999), which realized a powerful computer system by using many PCs connected by a network, was a milestone in parallel computer development. Freely available software products for parallel computing have become more mature. Thus, parallel computing has now become easy for statisticians to access.

In this chapter, we describe an overview of available technologies for parallel computing and give examples of their use in statistics. The next section considers the basic ideas of parallel computing, including memory architectures. Section 9.3 introduces the available software technologies such as process forking, threading, OpenMP, PVM (Parallel Virtual Machine), MPI (Message Passing Interface) and HPF (High Performance Fortran). The last section describes some examples of parallel computing in statistics.

9.2 Basic Ideas

Two important parts of computer hardware are the processor, which performs computations, and memory, in which programs and data are stored. A processor is also often called a central processing unit (CPU). Modern computer systems adopt a stored programming architecture: all the program instructions are stored in memory together with processed data and are executed sequentially by a processor according to the instructions.

In a traditional single processor computer, a single stream of instructions is generated from the program, and these instructions operate on a single stream of data. Flynn (1966) called this arrangement a single instruction stream–single data stream (SISD) computer.

On the other hand, a parallel computer system uses several processors, and is realized as a single instruction stream–multiple data stream (SIMD) computer or a multiple instruction stream–multiple data stream (MIMD) computer. SIMD refers to a form of parallel execution in which all processors execute the same operation on different data at the same time, and is often associated with performing the same operation on every element of a vector or array. MIMD refers to parallel execution in which each processor works independently; for example, one processor might update a database file while another processor handles a graphic display.

The fundamental software of a modern computer system is an operating system such as UNIX or Microsoft Windows. They support multiple users and multiple tasks, even on single processor systems, by adopting time-slicing mechanisms, in which a processor executes tasks cyclically. In parallel computer systems, some tasks are executed on different processors simultaneously.

9.2.1 Memory Architectures of Parallel Computers

The traditional computer system has a single processor (or CPU) that can access all of the memory (Fig. 9.1). Parallel computers use more than one processor simultaneously for a single calculation task. There are two simple methods to increase the number of available processors in a single system. One method is to add processors to the traditional single processor system without changing other parts. Because all the memory is shared by all processors, such systems are called shared memory systems (Fig. 9.2). An example of a shared memory system is a dual processor personal computer, where the motherboard has two sockets for CPUs. When we mount one CPU, it works as a traditional single processor system. If we mount two CPUs, both processors can access all the memory in the PC, and it works as a shared memory system. A second method is to connect traditional single processor computers by a network. This is called a distributed memory system, because the memory is used by a single processor locally and is "distributed" over the whole system (Fig. 9.3). An example of a distributed memory system is a network of workstations, in which each node computer works independently and communicates with the others through a network to solve a single problem.

Integration of shared memory and distributed memory is possible (Fig. 9.4). Network-connected PCs that each have two processors can be considered a distributed shared memory system.

Fig. 9.1 Traditional system

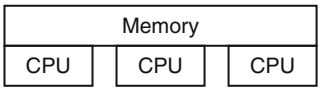


Fig. 9.2 Shared memory system

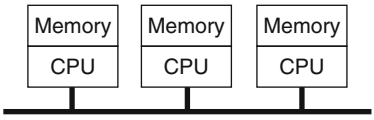


Fig. 9.3 Distributed memory system

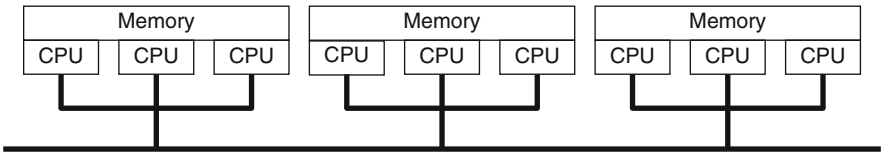


Fig. 9.4 Distributed shared memory system

Shared Memory Systems

In the simple shared memory realization, all the processors can access all the memory at the same speed by using a common memory bus. This is known as a uniform memory access (UMA) configuration. Performance in a UMA system is limited by the memory bus bandwidth; adding processors to the system beyond some point does not increase performance linearly, because signals from processors flow on the same memory bus and often cause collisions. Typically, UMA configurations do not scale well beyond 10–20 processors.

To improve communication between processors and memory, a non-uniform memory access (NUMA) configuration is used. In NUMA systems, all processors have access to all the memory, but the cost of accessing a specific location in memory is different for different processors, because different regions of memory are on physically different buses. Even if we adopt a NUMA configuration, it is not efficient to use more than 100 processors in a shared memory system.

A shared memory system is also a symmetric multiprocessor (SMP) system, in which any processor can do equally well any piece of work.

In a shared memory system, a single copy of an operating system is in charge of all the processors and the memory. It usually uses a programming model called "fork–join". Each program begins by executing just a single task, called the master. When the first parallel work is reached, the master spawns (or forks) additional tasks (called slaves or workers), which will "join" to the master when they finish their work (the middle figure in Fig. 9.5). Such activities can be programmed by using software technologies such as process, thread or OpenMP, which will be explained in the next section.

Distributed Memory Systems

In a distributed memory system, each node computer is an independent computer that has, at least, processor and memory, and the nodes are connected together by a network. This so called "network of workstations" (NOW) is the cheapest way to construct a distributed memory system, because we can utilize many different kinds of workstations available, connected by a network, without adding any new hardware. However, NOW is sometimes ineffective for heavy computation, because, for example, general purpose networks are slow, and nodes may be unexpectedly used for other work, so that it is difficult to schedule them efficiently.

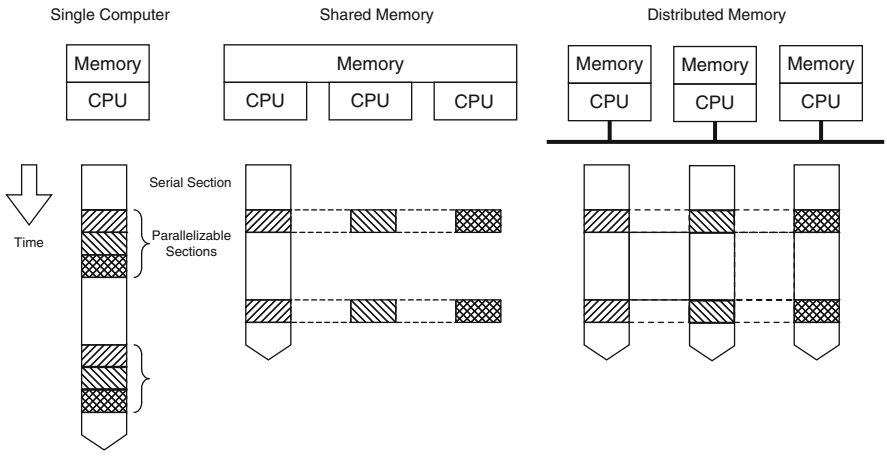


Fig. 9.5 Typical parallel computing execution

Nowadays, "Beowulf class cluster computers" are popular for distributed memory parallel computing (Sterling et al. 1999). These are a kind of NOW, but there are slight differences. First, the nodes in the cluster are the same kind of workstation or PC, and are dedicated to the cluster calculation tasks. Typically, node computers share the working directory on the hard disk and have no display or keyboard. The interconnection network is isolated from external networks and is also dedicated to the cluster, and communication among the nodes can be done without further authentication. Operating system parameters are tuned to improve the total performance for parallel computing. All these characteristics help the performance of the parallel computing on the cluster.

Distributed memory systems have no memory bus problem. Each processor can use the full bandwidth to its own local memory without interference from other processors. Thus, there is no inherent limit to the number of processors. The size of the system is constrained only by the network used to connect the node computers. Some distributed memory systems consist of several thousand processors.

As nodes in a distributed memory system share no memory at all, exchange of information among processors is more difficult than in a shared memory system. We usually adopt a message passing programming model on a distributed memory system; we organize a program as a set of independent tasks that communicate with each other via messages. This introduces two sources of overhead: it takes time to construct and send a message from one processor to another, and the receiving processor must be interrupted to deal with messages from other processors.

Available message passing libraries are PVM and MPI . The right figure in Fig. 9.5 shows an execution image of MPI. HPF is also mainly used in distributed memory systems. These libraries are illustrated in the next section.

9.2.2 Costs for Parallel Computing

We expect that the calculation speed increases n times if we use n processors instead of one. We also wish to use multiprocessor systems just like an ordinary single processor system. However, some costs are incurred in realizing parallel computing. They include the non-parallel characteristics of the problem, communication costs such as distributing and gathering data and/or programs, the difficulty of programming for synchronization among executions and unexpected influences of cache memory. All these factors reduce the effect of parallelization.

Amdahl's Law

All programming tasks include non-parallelizable or serial parts, which cannot be executed on several processors, for example, summarizing calculation results and writing them to the display or file. Assume the ratio of computing time for the serial parts to the whole task is f $(0 < f < 1)$. If a single processor requires t_s time to complete the task, $(1-f)t_s$ computation time is used for the parallelizable task and ft_s computation time is used for the serial task. If we use n processors, the elapsed time for execution of the parallelizable task will be at least $(1-f)t_s/n$, while the execution time of the serial task remains ft_s. Thus, the ratio of execution time for n processors to that for one processor, $S(n)$, which is called the speedup factor, is

$$S(n) = \frac{t_s}{ft_s + (1-f)t_s/n} = \frac{n}{1+(n-1)f}.$$

This equation is known as "Amdahl's law" (Amdahl 1967). When n is large, it converges to $1/f$, that is, the effect of parallel computing is limited. For example, if $f = 5\%$, the maximum possible speedup is 20, even if we use an infinite number of processors. This may discourage the use of parallel computing.

Of course, as f goes to zero, $S(n)$ converges to n, which is an ideal situation.

Gustafson's Law

Amdahl's law considers the situation where the task size is fixed and the number of processors increases. In real problems, however, we wish to perform larger tasks when the number of processors increases. For example, assume time s is required for preparing a task, and time p is required for the (moderate) simulation task. When a parallel computer is available, we wish to perform more simulations, typically, n times larger simulations than the original ones by n processors. To perform this simulation, a single processor system requires $s + np$ time, while the n-processor system requires $s + p$ time. The speedup factor is

$$S(n) = \frac{s + np}{s + p}.$$

This equation is called "Gustafson's law" (Gustafson 1988). Note that if we define $f = s/(s + np)$, this is as same as Amdahl's law. However, when n becomes large, $S(n)$ becomes large linearly. This means that parallel computing is useful for large-scale problems in which the serial part does not increase as the problem size increases. If s approaches zero, $S(n)$ converges to n, the ideal situation.

Other Costs

If we divide one task into several small tasks and execute them in parallel, we must wait until all the child tasks have been completed: we must synchronize executions. As the slowest child task determines the total execution time, child tasks should be designed to have almost the same execution times, otherwise some processors may be idle while others have tasks queuing for execution. Techniques that aim to spread tasks among the processors equally are called load balancing and are not easy.

In a shared memory system, exchange of information among processors is performed by variables stored in the shared memory. If several tasks use one variable almost simultaneously, it may cause trouble. Consider two tasks trying to decrease the value of variable x by one. Assume $x = 3$; task 1 obtains this value, decreases it and writes 2 into x. If task 2 tries to do the same task before task 1 finishes its work, task 2 also obtains the value 3, and writes 2 into x. Thus, the final result is 2, although x should have decreased twice. To avoid such a maloperation, task 2 must wait until task 1 finishes. All parallel computing software can handle this synchronization problem, typically by using a lock-unlock mechanism.

An important hardware aspect of shared memory systems is cache memory. As the advances in main memory technology do not keep up with processor innovations, memory access is very slow compared with processor speed. In order to solve this problem, another layer of memory has been added between a processor and main memory, called the cache. It is a small amount of very fast, expensive memory, that operates close to the speed of the processor. A separate cache controller monitors memory accesses and loads data and instructions in blocks of contiguous locations from memory into the cache. Once the content of memory is stored in the cache, the processor can operate at full speed by using them. Sometimes, the cache contents are different from the necessary ones. In these cases, the processor is stalled and has to wait while the necessary data is newly loaded from memory into the cache. This mechanism works well in a single processor system.

All processors in a shared memory system have their own caches. Suppose several processors access the same location of memory and copy them into their caches. If one processor changes the value of the memory in that location, other processors should not use the value in their caches. A cache coherence protocol is used to notify this information among caches. A common cache coherence protocol is an invalidate policy; when one copy of memory is altered, the same data in

any other cache is invalidated (by resetting a valid bit in the cache). In shared memory systems, cache coherence is done in the hardware and the programmer need not worry about cache coherence. However, it may cause the slowdown of the calculation speed. Note that caches handle blocks of memory. If one processor writes to one part of the block, copies of the whole block in other caches are invalidated though the actual data is not shared. This is known as false sharing and can damage the performance of the cache in a shared memory system. We are sometimes required to write programs considering the amount of the cache memory in a shared memory system to achieve enough performance.

Distributed memory systems require communication among node computers. Such communication is affected by several factors, including network bandwidth, network latency and communication latency. Network bandwidth is the number of bits that can be transmitted in unit time. Network latency is the time to prepare a message for sending it through the network. Communication latency is the total time to send the message, including software overhead and interface delays. Generally, communication is expensive compared with processor work.

If a problem can be divided into small tasks that are completely independent and require no or very little synchronization and communication, the problem is called "embarrassingly parallel". Clearly, embarrassingly parallel problems are particularly suitable for parallel computing.

9.3 Parallel Computing Software

Several well-designed software technologies are available for utilizing parallel computing hardware. Note that each of them is suitable for a specific hardware architecture.

In this section, we use as an example the calculation of the value of π by the approximation formula

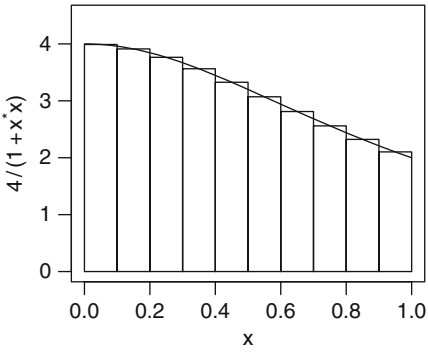


Fig. 9.6 Calculation of π

$$\pi = \int_0^1 \frac{4}{1+x^2} \mathrm{d}x \sim \frac{1}{n} \sum_{i=1}^{n} \frac{4}{1+\left(\frac{i-0.5}{n}\right)^2}.$$

The case $n = 10$ is illustrated in Fig. 9.6.

A C program to calculate the last term is given in Listing 9.3. The main calculation is performed in the `for` statement, which is easily divided into parallel-executed parts; this is an example of an embarrassingly parallel problem. We show several parallel computing techniques by using this example in this section. We choose this simple example to keep the length of following example source codes as small as possible and to give a rough idea of parallel computing techniques. Note that this example is so simple that only the most fundamental parts of each technique will be used and explained. Many important details of each technique are left to references.

```
#include <stdio.h>

main(int argc, char **argv)
{
  int n, i;
  double d, s, x, pi;
  n = atoi(argv[1]);
  d = 1.0/n;
  s = 0.0;
  for (i=1; i<=n; i++){
    x = (i-0.5)*d;
    s += 4.0/(1.0+x*x);
  }
  pi = d*s;
  printf("pi=%.15f\n", pi);
}
```

9.3.1 Process Forking

Modern operating systems have multi-user and multi-task features even on a single processor; many users can use a single processor system and can seemingly perform many tasks at the same time. This is usually realized by multi-process mechanisms (Tanenbaum 2001).

UNIX-like operating systems are based on the notion of a process. A process is an entity that executes a given piece of code, has its own execution stack, its own set of memory pages, its own file descriptors table and a unique process ID. Multiprocessing is realized by time-slicing the use of the processor. This technology repeatedly assigns the processor to each process for a short time. As the processor is very fast compared with human activities, it looks as though it is working

simultaneously for several users. In shared memory systems, multiprocessing may be performed simultaneously on several processors. Multiprocessing mechanisms are a simple tool for realizing parallel computing.

We can use two processes to calculate the `for` loop in Listing 9.3, by using the process-handling functions of UNIX operating systems: `fork()`, `wait()` and `exit()`. The function `fork()` creates a new copy process of an existing process. The new process is called the child process, and the original process is called the parent. The return value from `fork()` is used to distinguish the parent from the child; the parent receives the child's process id, but the child receives zero. By using this mechanism, an `if` statement, for example, can be used to prescribe different work for the parent and the child. The child process finishes by calling the `exit()` function, and the parent process waits for the end of the child process by using the `wait()` function. This fork–join mechanism is fundamental to the UNIX operating system, in which the first process to start invokes another process by forking. This procedure is repeated until enough processes are invoked. Although this mechanism was originally developed for one processor and a time-slicing system, UNIX operating systems that support shared memory can run processes on different processors simultaneously.

As processes are independent and share only a limited set of common resources automatically, we must write a program for information exchange among processes. In our example, we use functions to handle shared memory segments: `shmget()`, `shmat()` and `shmctl()`. `shmget()` allocates a shared memory segment, `shmat()` attaches the shared memory segment to the process, and `shmctl()` allows the user to set information such as the owner, group and permissions on the shared memory segment. When the parent process uses `fork()`, the shared memory segment is inherited by the child process and both processes can access it.

Listing 9.3.1 shows a two-process version of Listing 9.3. In the `for` statement, the parent process works for $i = 2, 4, 6, \ldots$, while the child process works for $i = 1, 3, 5, \ldots$ The child process stores its result to `*shared` and the parent process receives the value and adds it to its own result, then prints the final result.

```
#include <stdio.h>
#include <sys/types.h>
#include <unistd.h>
#include <sys/ipc.h>
#include <sys/shm.h>
main(int argc, char **argv)
{
  int n, i;
  double d, s, x, pi;
  int shmid, iproc;
  pid_t pid;
  double *shared;
  n = atoi(argv[1]);
  d = 1.0/n;
```

```
  shmid = shmget(IPC_PRIVATE,
                 sizeof(double), (IPC_CREAT | 0600));
  shared = shmat(shmid, 0, 0);
  shmctl(shmid, IPC_RMID, 0);
  iproc = 0;
  if ((pid = fork()) == -1) {
    fprintf(stderr, "The fork failed!\n");
    exit(0);
  } else {
    if (pid != 0) iproc = 1 ;
  }
  s = 0.0;
  for (i=iproc+1; i<=n; i+=2) {
    x = (i-0.5)*d;
    s += 4.0/(1.0+x*x);
  }
  pi = d*s;
  if (pid == 0) {
    *shared = pi;
    exit(0);
  } else {
    wait(0);
    pi = pi + *shared;
    printf("pi=%.15f\n", pi);
  }
}
```

Forking, however, is not appropriate for parallel computing. Much time and memory is required to duplicate everything in the parent process. Further, a complete copy is not always required, because, for example, the forked child process starts execution at the point of the fork.

9.3.2 Threading

As a process created using the UNIX `fork()` function is expensive in setup time and memory space, it is sometimes called a "heavyweight" process. Often a partial copy of the process is enough and other parts can be shared. Such copies can be realized by a thread or "lightweight" process. A thread is a stream of instructions that can be scheduled as an independent unit. It is important to understand the difference between a thread and a process. A process contains two kinds of information: resources that are available to the entire process such as program instructions, global data and working directory, and schedulable entities,

which include program counters and stacks. A thread is an entity within a process that consists of the schedulable part of the process.

In a single processor system, threads are executed by time-slicing, but shared memory parallel computers can assign threads to different processors.

Pthread Library

There were many thread libraries in the C language for specific shared memory systems. Now, however, the Pthread library is a standard thread library for many systems (Butenhof 1997). The Pthread API is defined in the ANSI/IEEE POSIX 1003.1-1995 standard, which can be purchased from IEEE.

Listing 9.3.2 is an example program to calculate π by using the Pthread library. The program creates a thread using the function `pthread_create()`, then assigns a unique identifier to a variable of type `pthread_t`. The caller provides a function that will be executed by the thread. The function `pthread_exit()` is used to terminate itself. The function `pthread_join()` is analogous to `wait()` for forking, but any thread may join any other thread in the process, that is, there is no parent–child relationship.

As multi-threaded applications execute instructions concurrently, access to process-wide (or interprocess) shared memory requires a mechanism for coordination or synchronization among threads. It is realized by mutual exclusion (mutex) locks. Mutexes furnish the means to guard data structures from concurrent modification. When one thread has locked the mutex, this mechanism precludes other threads from changing the contents of the protected structure until the locker performs the corresponding mutex unlock. Functions `pthread_mutex_init()`, `pthread_mutex_lock()` and `pthread_mutex_unlock()` are used for this purpose.

The compiled executable file is invoked from a command line with two arguments: `n` and the number of threads, which is copied to the global variable `num_threads`. The ith thread of the function `PIworker`, which receives the value i from the original process, calculates a summation for about `n/num_threads` times. Each thread adds its result to a global variable `pi`. As the variable `pi` should not be accessed by more than one thread simultaneously, this operation is locked and unlocked by the mutex mechanism.

```
#include <stdio.h>
#include <pthread.h>
int n, num_threads;
double d, pi;
pthread_mutex_t reduction_mutex;
pthread_t *tid;

void *PIworker(void *arg)
{
```

```
  int i, myid;
  double s, x, mypi;
  myid = *(int *)arg;
  s = 0.0;
  for (i=myid+1; i<=n; i+=num_threads) {
    x = (i-0.5)*d;
    s += 4.0/(1.0+x*x);
  }
  mypi = d*s;
  pthread_mutex_lock(&reduction_mutex);
  pi += mypi;
  pthread_mutex_unlock(&reduction_mutex);
  pthread_exit(0);
}

main(int argc, char **argv)
{
  int i;
  int *id;
  n = atoi(argv[1]);
  num_threads = atoi(argv[2]);
  d = 1.0/n;
  pi = 0.0;
  id = (int *) calloc(n,sizeof(int));
  tid = (pthread_t *) calloc(num_threads,
                             sizeof(pthread_t));
  if(pthread_mutex_init(&reduction_mutex,NULL)) {
    fprintf(stderr, "Cannot init lock\n");
    exit(0);
  };
  for (i=0; i<num_threads; i++) {
    id[i] = i;
    if(pthread_create(&tid[i],NULL,
                      PIworker,(void *)&id[i])) {
      exit(1);
    };
  };
  for (i=0; i<num_threads; i++)
    pthread_join(tid[i],NULL);
  printf("pi=%.15f\n", pi);
}
```

We note that it is not easy to write multi-threaded applications in the C language, even if we use the Pthread library. As the Pthread library was added to the C language later, there are no assurances that original basic libraries are “thread-safe”. The term thread-safe means that a given library function is implemented in such a manner that it can be executed correctly by multiple concurrent threads of execu-

tion. We must be careful to use thread-safe functions in multi-thread programming. The Pthread library is mainly used by professional system programmers to support advanced parallel computing technologies such as OpenMP.

Java Threads

The Java language supports threads as one of its essential features (Oaks and Wong 1999). The Java library provides a `Thread` class that supports a rich collection of methods: for example, the method `start()` causes the thread to execute the method `run()`, the method `join()` waits for the thread to finish execution. The lock–unlock mechanism can be easily realized by the `synchronized` declaration. All fundamental libraries are thread-safe. These features make Java suitable for thread programming.

```
public class PiJavaThread {
  int n, numThreads;
  double pi = 0.0;
  synchronized void addPi(double p) {
    pi += p;
  }
  public PiJavaThread(int nd, int nt) {
    n = nd;
    numThreads = nt;
    Thread threads[] = new Thread[numThreads];
    for (int i=0; i<numThreads; i++) {
      threads[i] = new Thread(new PIworker(i));
      threads[i].start();
    }
    for (int i=0; i<numThreads; i++) {
      try {
        threads[i].join();
      } catch (InterruptedException e) {
        e.printStackTrace();
      }
    }
  }
  class PIworker implements Runnable {
    int myid;
    public PIworker(int id) {
      myid = id;
    }
    public void run() {
      double d, s, x;
      d = 1.0/n;
```

```
        s = 0.0;
        for (int i=myid+1; i<=n; i+=numThreads) {
          x = (i-0.5)*d;
          s += 4.0/(1.0+x*x);
        }
        addPi(d*s);
      }
    }
    public static void main(String[] args) {
      PiJavaThread piJavaThread
         = new PiJavaThread(Integer.parseInt(args[0]),
                            Integer.parseInt(args[1]));
      System.out.println(" pi = " + piJavaThread.pi);
    }
}
```

Listing 9.3.2 is an example program to calculate the value of π using the Java language. This program is almost the same as the Pthread example. As the method declaration for `addPi()` contains the keyword `synchronized`, it can be performed by only one thread; other threads must wait until the `addPi()` method of the currently executing thread finishes.

Although the Java language is designed to be thread-safe and provides several means for thread programming, it is still difficult to write efficient application programs in Java. Java's tools are generally well suited to system programming applications, such as graphical user interfaces and distributed systems, because they provide synchronization operations that are detailed and powerful, but unstructured and complex. They can be considered an assembly language for thread programming. Thus, it is not easy to use them for statistical programming.

9.3.3 OpenMP

OpenMP is a directive-based parallelization technique (Chandra et al. 2001) that supports fork–join parallelism and is mainly for shared memory systems. The MP in OpenMP stands for "Multi Processing". It supports Fortran (77 and 90), C and C++, and is suitable for numerical calculation, including statistical computing. It is standardized for portability by the OpenMP Architecture Review Board (OpenMP Architecture Review Board 2004). The first Fortran specification 1.0 was released in 1997, and was updated as Fortran specification 1.1 in 1999. New features were added as Fortran specification 2.0 in 2000. Several commercial compilers support OpenMP.

We use the Fortran language for our examples in this section, because Fortran is still mainly used for high-performance computers focused on large numerical computation. Fortran is one of the oldest computer languages and has many reliable

and efficient numerical libraries and compilers. The Fortran program for the simple π computation is shown in Listing 9.3.3.

We note that C (and C++) are also used for large numerical computations and are now supported to the same extent as Fortran. The following examples can easily be replaced by C programs (except the HPF examples) but we omit them for space reasons.

```
      integer n, i
      double precision d, s, x, pi
      write(*,*) 'n?'
      read(*,*) n
      d = 1.0/n
      s = 0.0
      do i=1, n
        x = (i-0.5)*d
        s = s+4.0/(1.0+x*x)
      enddo
      pi = d*s
      write(*,100) pi
 100  format(' pi = ', f20.15)
      end
```

We can parallelize this program simply by using OpenMP directives (Listing 9.3.3).

```
      integer n, i
      double precision d, s, x, pi
      write(*,*) 'n?'
      read(*,*) n
      d = 1.0/n
      s = 0.0
!$OMP PARALLEL PRIVATE(x), SHARED(d)
!$OMP& REDUCTION(+: s)
!$OMP DO
      do i = 1, n
         x = (i-0.5)*d
         s = s+4.0/(1.0+x*x)
      end do
!$OMP END DO
!$OMP END PARALLEL
      pi = d*s
      write(*,100) pi
 100    format(' pi = ', f20.15)
      end
```

Lines started by `!$OMP` are OpenMP directives to specify parallel computing. Each OpenMP directive starts with `!$OMP`, followed by a directive and, optionally,

clauses. For example, "`!$OMP PARALLEL`" and "`!$OMP END PARALLEL`" encloses a parallel region and all code lexically enclosed is executed by all threads. The number of threads is usually specified by an environmental variable `OMP_NUM_THREADS` in the shell environment. We also require a process distribution directive "`!$OMP DO`" and "`!$OMP END DO`" to enclose a loop that is to be executed in parallel. Within a parallel region, data can either be private to each executing thread, or be shared among threads. By default, all data in static extents are shared (an exception is the loop variable of the parallel loop, which is always private). In the example, shared scope is not desirable for `x` and `s`, so we use a suitable clause to make them private: "`!$OMP PARALLEL PRIVATE (x, s)`". By default, data in dynamic extent (subroutine calls) are private (an exception is data with the `SAVE` attribute), and data in `COMMON` blocks are shared.

An OpenMP compiler will automatically translate this program into a Pthread program that can be executed by several processors on shared memory systems.

9.3.4 PVM

PVM (Parallel Virtual Machine) is one of the first widely used message passing programming systems. It was designed to link separate host machines to form a virtual machine, which is a single manageable computing resource (Geist et al. 1994). It is (mainly) suitable for heterogeneous distributed memory systems. The first version of PVM was written in 1989 at Oak Ridge National Laboratory, but was not released publicly. Version 2 was written at the University of Tennessee Knoxville and released in 1991. Version 3 was redesigned and released in 1993. Version 3.4 was released in 1997. The newest minor version, 3.3.4, was released in 2001 (PVM Project Members 2004).

PVM is freely available and portable (available on Windows and several UNIX systems). It is mainly used in Fortran, C and C++, and extended to be used in many other languages, such as Tcl/Tk, Perl and Python.

The PVM system is composed of two parts: a PVM daemon program (pvmd) and libraries of PVM interface routines. Pvmd provides communication and process control between computers. One pvmd runs on each host of a virtual machine. It serves as a message router and controller, and provides a point of contact, authentication, process control and fault detection. The first pvmd (which must be started by the user) is designated the master, while the others (started by the master) are called slaves or workers.

PVM libraries such as `libpvm3.a` and `libfpvm3.a` allow a task to interface with the pvmd and other tasks. They contain functions for packing and unpacking messages, and functions to perform PVM calls by using the message functions to send service requests to the pvmd.

Example Fortran programs are in Listings 9.3.4 and 9.3.4.

```
      program pimaster
      include '/usr/share/pvm3/include/fpvm3.h'
      integer n, i
      double precision d, s, pi
      integer mytid,numprocs,tids(0:32),status
      integer numt,msgtype,info
      character*8 arch
      write(*,*) 'n, numprocs?'
      read(*,*) n, numprocs
      call PVMFMYTID(mytid)
      arch = '*'
      call PVMFSPAWN('piworker',PVMDEFAULT,arch,
     $               numprocs,tids,numt)
      if( numt .lt. numprocs) then
         write(*,*) 'trouble spawning'
         call PVMFEXIT(info)
         stop
      endif
      d = 1.0/n
      msgtype = 0
      do 10 i=0, numprocs-1
        call PVMFINITSEND(PVMDEFAULT,info)
        call PVMFPACK(INTEGER4, numprocs, 1, 1, info)
        call PVMFPACK(INTEGER4, i,        1, 1, info)
        call PVMFPACK(INTEGER4, n,        1, 1, info)
        call PVMFPACK(REAL8,    d,        1, 1, info)
        call PVMFSEND(tids(i),msgtype,info)
10    continue
      s=0.0
      msgtype = 5
      do 20 i=0, numprocs-1
         call PVMFRECV(-1,msgtype,info)
         call PVMFUNPACK(REAL8,x,1,1,info)
         s = s+x
20    continue
      pi = d*s
      write(*,100) pi
100   format(' pi = ', f20.15)
      call PVMFEXIT(info)
      end

      program piworker
      include '/usr/share/pvm3/include/fpvm3.h'
      integer n, i
      double precision s, x, d
```

```
      integer mytid,myid,numprocs,msgtype,master,info
      call PVMFMYTID(mytid)
      msgtype = 0
      call PVMFRECV(-1,msgtype,info)
      call PVMFUNPACK(INTEGER4, numprocs, 1, 1, info)
      call PVMFUNPACK(INTEGER4, myid,     1, 1, info)
      call PVMFUNPACK(INTEGER4, n,        1, 1, info)
      call PVMFUNPACK(REAL8,    d,        1, 1, info)
      s = 0.0
      do 10 i = myid+1, n, numprocs
        x = (i-0.5)*d
        s = s+4.0/(1.0+x*x)
10    continue
      call PVMFINITSEND(PVMDEFAULT,info)
      call PVMFPACK(REAL8, s,1,1, info)
      call PVMFPARENT(master)
      msgtype = 5
      call PVMFSEND(master,msgtype,info)
      call PVMFEXIT(info)
      end
```

Listing 9.3.4 is the master program, and Listing 9.3.4 is the slave program, and its compiled executable file name should be `piworker`. Both programs include the Fortran PVM header file `fpvm3.h`.

The first PVM call `PVMFMYTID()` in the master program informs the pvmd of its existence and assigns a task id to the calling task.

After the program is enrolled in the virtual machine, the master program spawns slave processes by the routine `PVMFSPAWN()`. The first argument is a string containing the name of the executable file that is to be used. The fourth argument specifies the number of copies of the task to be spawned and the fifth argument is an integer array that is to contain the task ids of all tasks successfully spawned. The routine returns the number of tasks that were successfully created via the last argument.

To send a message from one task to another, a send buffer is created to hold the data. The routine `PVMFINITSEND()` creates and clears the buffer and returns a buffer identifier. The buffer must be packed with data to be sent by the routine `PVMFPACK()`. The first argument specifies the type of data to be packed. The second argument is the first item to be packed, the third is the total number of items to be packed and the fourth is the stride to use when packing. A single message can contain any number of different data types; however, we should ensure that the received message is unpacked in the same way it was originally packed by the routine `PVMFUNPACK()`. The routine `PVMFSEND()` attaches an integer label of `msgtype` and sends the contents of the send buffer to the task specified by the first argument.

After the required data have been distributed to each worker process, the master program must receive a partial sum from each of the worker processes by the `PVMFRECV()` routine. This receives a message from the task specified by the first argument with the label of the second argument and places it into the receive buffer. Note that a value of `-1` for an argument will match with any task id and/or label. The master program expects a label value of `5` on messages from the worker tasks.

The unpacking routine `PVMFUNPACK()` has the same arguments as `PVMFPACK()`. The second argument shows where the first item unpacked is to be stored.

After the sum has been computed and printed, the master task informs the PVM daemon that it is withdrawing from the virtual machine. This is done by calling the routine `PVMFEXIT()`.

The worker program uses the same PVM routines as the master program. It also uses `PVMFPARENT()` routine to find the task id of the master task that spawned the current task.

When we compile Fortran PVM codes, we must link in both the PVM Fortran library and the standard PVM library compiled for the target machine architecture. Before executing the program, the executables of the worker program should be available in a specific directory on all the slave nodes. The default authentication is performed by `rsh` call.

9.3.5 MPI

MPI (Message Passing Interface) is the most widely used parallel computing technique. It specifies a library for adding message passing mechanisms to existing languages such as Fortran or C. MPI is mainly used for homogeneous distributed memory systems.

MPI appeared after PVM. PVM was a research effort and did not address the full spectrum of issues: it lacked vendor support, and was not implemented at the most efficient level for a particular hardware. The MPI Forum (Message Passing Interface MPI) was organized in 1992 with broad participation by vendors (such as IBM, Intel, SGI), portability library writers (including PVM), and users such as application scientists and library writers. MPI-1.1 was released in 1995, MPI-1.2 was released in 1997, and MPI-2 was released in 1997.

MPI-1 has several functions that were not implemented in PVM. Communicators encapsulate communication spaces for library safety. Data types reduce copying costs and permit heterogeneity. Multiple communication modes allow precise buffer management. MPI-1 has extensive collective operations for scalable global communication, and supports process topologies that permit efficient process placement and user views of process layout (Gropp et al. 1999a).

In MPI-2, other functions were added: extensions to the message passing model, dynamic process management, one-sided operations (remote memory access), parallel I/O, thread support, C++ and Fortran 90 bindings, and extended collective operations (Gropp et al. 1999b).

MPI implementations are released from both vendors and research groups. MPICH (MPICH Team 2004) and LAM/MPI (LAM Team 2004) are widely used free implementations.

Although MPI has more than 150 routines, many parallel programs can be written using just six routines, only two of which are non-trivial: `MPI_INIT()`, `MPI_FINALIZE()`,`MPI_COMM_SIZE()`,`MPI_COMM_RANK()`,`MPI_SEND()` and `MPI_RECV()`. An example program is shown in Listing 9.3.5.

```
      include 'mpif.h'
      integer n, i
      double precision  d, s, x, pi, temp
      integer myid, numprocs, ierr, status(3)
      integer sumtag, sizetag, master
      call MPI_INIT(ierr)
      call MPI_COMM_SIZE(MPI_COMM_WORLD,numprocs,ierr)
      call MPI_COMM_RANK(MPI_COMM_WORLD,myid,ierr)
      sizetag = 10
      sumtag = 17
      master = 0
      if (myid .eq. master) then
        write(*,*) 'n?'
        read(*,*) n
        do i = 1, numprocs-1
          call MPI_SEND(n,1,MPI_INTEGER,i,sizetag,
     $          MPI_COMM_WORLD,ierr)
        enddo
      else
        call MPI_RECV(n,1,MPI_INTEGER,master,sizetag,
     $         MPI_COMM_WORLD,status,ierr)
      endif
      d = 1.0/n
      s = 0.0
      do i = myid+1, n, numprocs
         x = (i-0.5)*d
         s = s+4.0/(1.0+x*x)
      enddo
      pi = d*s
      if (myid .ne. master) then
        call MPI_SEND(pi,1,MPI_DOUBLE_PRECISION,
     $         master,sumtag,MPI_COMM_WORLD,ierr)
      else
        do i = 1, numprocs-1
          call MPI_RECV(temp,1,MPI_DOUBLE_PRECISION,
     $           i,sumtag,MPI_COMM_WORLD,status,ierr)
          pi = pi+temp
```

```
        enddo
      endif
      if (myid .eq. master) then
        write(*, 100) pi
100     format(' pi = ', f20.15)
      endif
      call MPI_FINALIZE(ierr)
      end
```

MPI follows the single program-multiple data (SPMD) parallel execution model. SPMD is a restricted version of MIMD in which all processors run the same programs, but unlike SIMD, each processor may take a different flow path in the common program.

If the example program is stored in file `prog8.f`, typical command lines for executing it are

```
f77 -o prog8 prog8.f -lmpi
mpirun -np 5 prog8
```

where the command `mpirun` starts five copies of process `prog8` simultaneously. All processes communicate via MPI routines.

The first MPI call must be `MPI_INIT()`, which initializes the message passing routines. In MPI, we can divide our tasks into groups, called communicators. `MPI_COMM_SIZE()` is used to find the number of tasks in a specified MPI communicator. In the example, we use the communicator `MPI_COMM_WORLD`, which includes all MPI processes. `MPI_COMM_RANK()` finds the rank (the name or identifier) of the tasks running the code. Each task in a communicator is assigned an identifying number from 0 to `numprocs-1`.

`MPI_SEND()` allows the passing of any kind of variable, even a large array, to any group of tasks. The first argument is the variable we want to send, the second argument is the number of elements passed. The third argument is the kind of variable, the fourth is the id number of the task to which we send the message, and the fifth is a message tag by which the receiver verifies that it receives the message it expects. Once a message is sent, we must receive it on another task. The arguments of the routine `MPI_RECV()` are similar to those of `MPI_SEND()`. When we finish with the message passing routines, we must close out the MPI routines by the call `MPI_FINALIZE()`.

In parallel computing, collective operations often appears. MPI supports useful routines for them. `MPI_BCAST` distributes data from one process to all others in a communicator. `MPI_REDUCE` combines data from all processes in a communicator and returns it to one process. In many numerical algorithms, `SEND/RECEIVE` can be replaced by `BCAST/REDUCE`, improving both simplicity and efficiency. Listing 9.3.5 can be replaced by Listing 9.3.5 (some parts of Listing 8 are omitted).

```
      ...
master = 0
if (myid .eq. master) then
```

```
          write(*,*) 'n?'
          read(*,*) n
        endif
        call MPI_BCAST(n,1,MPI_INTEGER,master,
       $      MPI_COMM_WORLD,ierr)
        d = 1.0/n
        s = 0.0
            ...
        enddo
        pi = d*s
        call MPI_REDUCE(pi,temp,1,MPI_DOUBLE_PRECISION,
       $      MPI_SUM,master,MPI_COMM_WORLD,ierr)
        pi = temp
        if (myid .eq. master) then
          write(*, 100) pi
            ...
```

In distributed shared memory systems, both OpenMP and MPI can be used together to use all the processors efficiently. Again, Listing 9.3.5 can be replaced by Listing 9.3.5 (the same parts of Listing 9.3.5 are omitted) to use OpenMP.

```
            ...
        d = 1.0/n
        s = 0.0
!$OMP PARALLEL PRIVATE(x), SHARED(d)
!$OMP& REDUCTION(+: s)
!$OMP DO
        do i = myid+1, n, numprocs
           x = (i-0.5)*d
           s = s+4.0/(1.0+x*x)
        enddo
!$OMP END DO
!$OMP END PARALLEL
        pi = d*s
        if (myid .ne. master) then
           ...
```

9.3.6 HPF

HPF (High Performance Fortran) is a Fortran 90 with further data parallel programming features (Koelbel et al. 1993). In data parallel programming, we specify which processor owns what data, and the owner of the data does the computation on the data (Owner-computes rule).

Fortran 90 provides many features that are well suited to data parallel programming, such as array processing syntax, new functions for array calculations, modular programming constructs and object-oriented programming features.

HPF adds additional features to enable data parallel programming. We use compiler directives to distribute data on the processors, to align arrays and to declare that a loop can be calculated in parallel without affecting the numerical results. HPF also has a loop control structure that is more flexible than DO, and new intrinsic functions for array calculations.

The High Performance Fortran Forum (HPFF) (High Performance Fortran Forum 2004) is a coalition of industry, academic and laboratory representatives, and defined HPF 1.0 in 1993. HPF 1.1 was released in 1994 and HPF 2.0 was released in 1997. Several commercial and free HPF compilers are now available.

Listing 9.3.6 is an example program for calculating π in HPF.

```
      integer n, i
      double precision  d, s, pi
      double precision, dimension (:),
     $           allocatable :: x, y
!HPF$ PROCESSORS procs(4)
!HPF$ DISTRIBUTE x(CYCLIC) ONTO procs
!HPF$ ALIGN y(i) WITH x(i)
      write(*,*) 'n?'
      read(*,*) n
      allocate(x(n))
      allocate(y(n))
      d = 1.0/n
!HPF$ INDEPENDENT
      FORALL (i = 1:n)
        x(i) = (i-0.5)*d
        y(i) = 4.0/(1.0 + x(i)*x(i))
      end FORALL
      pi = d*SUM(y)
      write (*, 100) pi
 100  format(' pi = ', f20.15)
      deallocate(x)
      deallocate(y)
      end
```

`!HPF$` is used for all HPF compiler directives. We note that this is a comment to non-HPF compilers and is ignored by them. The `PROCESSORS` directive specifies the shape of the grid of abstract processors. Another example "`!HPF$ PROCESSORS exprocs(6,2)`" specifies a 6×2 array of 12 abstract processors labelled `exprocs`.

The `DISTRIBUTE` directive partitions an array by specifying a regular distribution pattern for each dimension `ONTO` the arrangement of abstract processors. The `CYCLIC` pattern spreads the elements one per processor, wrapping around when it

runs out of processors, i.e., this pattern distributes the data in the same way that the program in Listing 9.3.5 performs. Another pattern is `BLOCK`, which breaks the array into equal-sized blocks, one per processor. The rank of the abstract processor grid must be equal to the number of distributed axes of the array.

The `ALIGN` directive is used to specify relationships between data objects. In the example program, elements of x and y that have the same index are placed on the same processor.

The `INDEPENDENT` directive informs the compiler that in the execution of the `FORALL` construct or the do loop, no iteration affects any other iteration in any way.

The `FORALL` statement is a data parallel construct that defines the assignment of multiple elements in an array but does not restrict the order of assignment to individual elements. Note that the do loop executes on each element in a rigidly defined order.

The `SUM` intrinsic function performs reduction on whole arrays.

We may compare HPF with OpenMP, because both systems use compiler directives in a standard language (Fortran) syntax. In OpenMP, the user specifies the distribution of iterations, while in HPF, the user specifies the distribution of data. In other words, OpenMP adopts the instruction parallel programming model while HPF adopts data parallel programming model. OpenMP is suitable for shared memory systems whereas HPF is suitable for distributed memory systems.

9.4 Parallel Computing in Statistics

9.4.1 Parallel Applications in Statistical Computing

The most important thing in parallel computing is to divide a job into small tasks for parallel execution. We call the amount of independent parallel processing that can occur before requiring some sort of communication or synchronization the "granularity". Fine granularity may allow only a few arithmetic operations between processing one message and the next, whereas coarse granularity may allow millions. Although the parallel computing techniques described above can support programming of any granularity, coarse granularity is preferable for many statistical tasks. Fine granularity requires much information exchange among processors and it is difficult to write the required programs. Fortunately, many statistical tasks are easily divided into coarse granular tasks. Some of them are embarrassingly parallel.

In data analysis, we often wish to perform the same statistical calculations on many data sets. Each calculation for a data set is performed independently from other data sets, so the calculations can be performed simultaneously. For example, Hegland et al. (1999) implemented the backfitting algorithm to estimate a generalized additive model for a large data set by dividing it into small data sets, fitting a function in parallel and merging them together. Beddo (2002) performed parallel multiple correspondence analysis by dividing an original data set and merging their calculation results.

Another embarrassingly parallel example is a simulation or a resampling computation, which generates new data sets by using a random number generating mechanism based on a given data set or parameters. We calculate some statistics for those data sets, repeat such operations many times and summarize their results to show empirical distribution characteristics of the statistics. In this case, all calculations are performed simultaneously except the last part. Beddo (2002) provided an example of bootstrapping from parallel multiple correspondence analysis.

We must be careful that random numbers are appropriately generated in parallel execution. For example, random seeds for each process should all be different values, at least. SPRNG (Mascagni 1999) is a useful random number generator for parallel programming. It allows for the dynamic creation of independent random number streams on parallel machines without interprocessor communication. It is available in the MPI environment and the macro `SIMPLE_SPRNG` should be defined to invoke the simple interface. Then the macro `USE_MPI` is defined to instruct SPRNG to make MPI calls during initialization. Fortran users should include the header file `sprng_f.h` and call `sprng()` to obtain a double precision random number in $(0, 1)$. In compiling, the libraries `liblcg.a` and the MPI library should be linked.

The maximum likelihood method requires much computation and can be parallelized. Jones et al. (1999) describes a parallel implementation of the maximum likelihood estimation using the EM algorithm for positron emission tomography image reconstruction. Swann (2002) showed maximum likelihood estimation for a simple econometric problem with Fortran code and a full explanation of MPI. Malard (2002) solved a restricted maximum likelihood estimation of variance-covariance matrix by using freely available toolkits: the portable extensible toolkit for scientific computation (PETSc) and the toolkit for advanced optimazation (TAO) (Balay et al. 2001) which are built on MPI.

Optimization with dynamic programming requires much computation and is suitable for parallel computing. Hardwick et al. (1999) used this technique to solve sequential allocation problems involving three Bernoulli populations. Christofides et al. (1999) applied it to the problem of discretizing multidimensional probability functions.

Racine (2002) demonstrated that kernel density estimation is also calculated efficiently in parallel.

9.4.2 Parallel Software for Statistics

Several commercial and non-commercial parallel linear algebra packages that are useful for statistical computation are available for Fortran and/or C. We mention two non-commercial packages with freely available source codes: ScaLAPACK (Blackford et al. 1997) supports MPI and PVM, and PLAPACK (van de Geijin 1997) supports MPI. Murphy et al. (1999) described the work to transfer sequential

libraries (Gram-Schmidt orthogonalization and linear least squares with equally constraints) to parallel systems by using Fortran with MPI.

Although we have many statistical software products, few of them have parallel features. Parallel statistical systems are still at the research stage. Bull et al. (1999) ported a multilevel modeling package MLn into a shared memory system by using C++ with threads. Yamamoto and Nakano (2002) explained a system for time series analysis that has functions to use several computers via Tkpvm, an implementation of PVM in the Tcl/Tk language.

The statistical systems R (The R Development Core Team 2004) and S (Chambers 1998) have some projects to add parallel computing features. Temple Lang (1997) added thread functions to S. PVM and MPI are directly available from R via the rpvm (Li and Rossini 2001) and Rmpi (Yu 2002) packages. They are used to realize the package "snow" (Rossini et al. 2003), which implements simple commands for using a workstation cluster for embarrassingly parallel computations in R. A simple example session is:

```
> cl <- makeCluster(2, type = "PVM")
> clusterSetupSPRNG(cl)
> clusterCall(cl, runif, 3)
[[1]]
[1] 0.749391854 0.007316102 0.152742874

[[2]]
[1] 0.8424790 0.8896625 0.2256776
```

where a PVM cluster of two computers is started by the first command and the SPRNG library is prepared by the second command. Three uniform random numbers are generated on each computer and the results are printed by the third command.

The statistical system "Jasp" (Nakano et al. 2000) is implementing experimental parallel computing functions via network functions of the Java language (see also http://jasp.ism.ac.jp/).

References

Amdahl, G.M.: Validity of the single-processor approach to achieving large scale computing capabilities. In AFIPS Conference Proceedings, vol. 30, pp. 483–485 (1967)

Balay, S., Buschelman, K., Gropp, W.D., Kaushik, D., Knepley, M., McInnes, L.C., Smith, B.F., Zhang, H.: PETSc home page (2001); http://www.mcs.anl.gov/petsc

Beddo, V.: Applications of parallel programming in Statistics. Ph.D. dissertation, University of California, Los Angeles (2002); http://theses.stat.ucla.edu/19/parallel_programming_beddo.pdf

Blackford, L., Choi, J., Cleary, A., D'Azevedo, E., Demmel, J., Dhillon, I., Dongarra, J., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D., Whaley, R.C.: ScaLAPACK Users' Guide. SIAM Press (1997)

Bull, J.M., Riley, G.D., Rasbash, J., Goldstein, H.: Parallel implementation of a multilevel modelling package. Comput. Stat. Data Anal. **31**(4), 457–474 (1999)

Butenhof, D.R.: Programming with POSIX Threads. Addison Wesley, Reading, MA, USA (1997)
Chambers, J.M.: Programming with Data: A Guide to the S Language. Springer, Berlin (1998)
Chandra, R., Dagum, L., Kohr, D., Maydan, D., McDonald, J., Menon, R. Parallel Programming in OpenMP. Morgan Kaufman, Los Altos, CA (2001)
Christofides, A., Tanyi, B., Christofides, D., Whobrey, D., Christofides, N.: The optimal discretization of probability density functions. Comput. Stat. Data Anal. **31**(4), 475–486 (1999)
Flynn, M.: Very high-speed computing systems. Proc. IEEE **54**(12), 1901–1909 (1966)
Geist, A., Beguelin, A., Dongarra, J., Jiang, W., Manchek, R., Sunderam, V.S.: PVM: Parallel Virtual Machine: A Users' Guide and Tutorial for Networked Parallel Computing. MIT Press, Cambridge, MA (1994)
Gropp, W., Lusk, E., Skjellum, A.: Using MPI: Portable Parallel Programming with the Message-Passing Interface, 2nd Edition. MIT Press, Cambridge, MA (1999a)
Gropp, W., Lusk, E., Thakur, R.: Using MPI-2: Advanced Features of the Message-Passing Interface. MIT Press, Cambridge, MA (1999b)
Gustafson, J.L.: Reevaluating amdahl's law. Comm. ACM **31**(5), 532–533 (1988)
Hardwick, J., Oehmke, R., Stout, Q.F.: A program for sequential allocation of three bermoulli populations. Comput. Stat. Data Anal. **31**(4), 397–416 (1999)
Hegland, M., McIntosh, I., Turlach, B.A.: A parallel solver for generalized additive models. Comput. Stat. Data Anal. **31**(4), 377–396 (1999)
High Performance Fortran Forum: HPF: The high performance fortran home page (2004); http://www.crpc.rice.edu/HPFF/.
Jones, H., Mitra, G., Parkinson, D., Spinks, T.: A parallel implementation of the maximum likelihood method in positron emission tomography image reconstruction. Comput. Stat. Data Anal. **31**(4), 417–439 (1999)
Koelbel, C.H., Loveman, D.B., Schreiber, R.S., Steele, J. G.L., Zosel, M.E.: The High Performance Fortran Handbook. MIT Press, Cambridge, MA (1993)
LAM Team: LAM/MPI parallel computing (2004); http://www.lam-mpi.org/.
Li, N., Rossini, A.: RPVM: Cluster statistical computing in R. R News **1**(3), 4–7 (2001); http://CRAN.R-project.org/doc/Rnews/.
Malard, J.M.: Parallel restricted maximum likelihood estimation for linear models with a dense exogenous matrix. Parall. Comput. **28**(2), 343–353 (2002)
Mascagni, M.: SPRNG: A scalable library for pseudorandom number generation. In: Spanier, J. et al. (eds.) Proceedings of the Third International Conference on Monte Carlo and Quasi Monte Carlo Methods in Scientific Computing. Springer, Berlin (1999)
Message Passing Interface (MPI) Forum: Message passing interface (MPI) forum home page (2004); http://www.mpi-forum.org/.
MPICH Team: MPICH – A portable mpi implementation (2004); http://www-unix.mcs.anl.gov/mpi/mpich/.
Murphy, K., Clint, M., Perrott, R.H.: Re-engineering statistical software for efficient parallel execution. Comput. Stat. Data Anal. **31**(4), 441–456 (1999)
Nakano, J., Fujiwara, T., Yamamoto, Y., Kobayashi, I.: A statistical package based on Pnuts. In: Bethlehem, J.G., van der Heijden, P.G.M. (eds.) COMPSTAT 2000 Proceedings in Computational Statistics, pp. 361–366. Physica, Wurzburg (Wien) (2000)
Oaks, S., Wong, H.: Java Threads. (2nd edn.), O'Reilly (1999)
OpenMP Architecture Review Board: OpenMP: Simple, portable, scalable SMP programming (2004); http://www.openmp.org/.
PVM Project Members: PVM: Parallel virtual machine (2004); http://www.csm.ornl.gov/pvm/pvm_home.html.
Racine, J.: Parallel distributed kernel estimation. Comput. Stat. Data Anal. **40**(2), 293–302 (2002)
Rossini, A., Tierney, L., Li, N.: Simple parallel statistical computing in R. UW Biostatistics working paper series, Paper 193, University of Washington, USA (2003); http://www.bepress.com/uwbiostat/paper193.
Schervish, M.J.: Applications of parallel computation to statistical inference. J. Aner. Statist. Assoc. **83**, 976–983 (1988)

Sterling, T., Salmon, J., Becker, D.J., Savarese, D.F.: How to Build a Beowulf : A Guide to the Implementation and Application of PC Clusters. MIT Press, Cambridge, MA (1999)

Swann, C.A.: Maximum likelihood estimation using parallel computing: An introduction to MPI. Comput. Econ. **19**, 145–178 (2002)

Tanenbaum, A.S.: Modern Operating Systems. (2nd edn.), Prentice Hall, Netherland (2001)

Temple Lang, D.: A multi-threaded extension to a high level interactive statistical computing environment. Ph.D. dissertation, University of California, Berkeley (1997); http://cm.bell-labs.com/stat/doc/multi-threaded-S.ps.

The R Development Core Team: The R project for statistical computing (2004); http://www.r-project.org/.

Tuomi, I.: The lives and death of moore's law. First Monday **7**(11), (2002); http://firstmonday.org/issues/issue7_11/tuomi/index.html.

van de Geijin, R.A.: Using PLAPACK. MIT Press, Cambridge, MA (1997)

Yamamoto, Y., Nakano, J.: Distributed processing functions of a time series analysis system. J. Jpn. Soc. Comput. Stat. **15**(1), 65–77 (2002)

Yu, H.: Rmpi: Parallel statistical computing in R. R News **2**(2), 10–14 (2002); http://CRAN.R-project.org/doc/Rnews/.

Chapter 10
Statistical Databases

Claus Boyens, Oliver Günther, and Hans-J. Lenz

10.1 Introduction

Most data collected in technology, statistics and science is still stored in simple *flat files*, usually as data matrices with rows identified by a case identifier (case_id), columns corresponding to attributes (variables), and numerical data types for the elements of each data matrix due to a universal numeric coding of all attributes. Each row (tuple) carries the (coded) values of the attributes, besides the case_id. Due to an encoding that maps even a symbolic domain to a numerical one, all matrix entries have a numeric data type. The scales of the attributes – nominal, ordinal and cardinal – may of course be quite different.

A simple example is given by census data stored at statistical offices in files according to a schema like `census_questionnaire (case_id, age-group, gender, profession,...)`. While science gains their data from observational sampling or experiments, statistical agencies collect their data still mostly off-line from surveys, reports or census. Industry and services get their data on-line from their business processes, i.e., from their logistical, production and administrative transactions. A typical example is sales data, which may be represented by a schema like

`sales (transaction_id, customer_id, date, product_name, quantity, price, amount).`

C. Boyens
1 & 1 Internet AG Karlsruhe, Germany
e-mail: claus.boyens@web.de

O. Günther
Universität Potsdam, Präsidialamt, Potsdam, Germany
e-mail: guenther@wiwi.hu-berlin.de

H.-J. Lenz (✉)
Institut für Statistik und Ökonometrie, Freie Universität Berlin, Berlin, Germany
e-mail: Hans-J.Lenz@fu-berlin.de

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_10,

Such data is called *microdata*, since it is kept in its original form and is not divisible but atomic. In the business area such data is labeled as *on-line transaction data* because it is subject to frequent updates and is the basis for the bulk of continuous business transactions. The use of a simple file system to store microdata is rarely a good choice because of a lack of independence between applications and data, safety and integrity, and, consequently, retrieval problems with especially ad hoc queries. Such data should rather be stored as tables of a relational database. A *database management system* (*DBMS*) asserts safety, integrity and retrieval flexibility. For instance, a query like "Find prices and amount of all sales since year 2001 where customer equals 007 and product 4711" can be simply stated in SQL (*structured query language*) as

```
SELECT price, amount FROM sales
WHERE year $\ge $ 2001
AND customer_id = 007
AND product_name = 4711;
```

It is interesting to note that SQL provides for a set of query operators that is relationally complete. One may thus process any reasonable query as long as it does not involve "transitive closure", i.e. a potentially infinite loop based on some logical inference (such as a part-of hierarchy).

Macrodata is derived from microdata by applying statistical functions, aggregation and grouping, and consequently has a larger granularity. For example, a business analyst might be interested in a *three-way table* (*data cube*) of total sales classified by `month` and `year, customer_id` and `product_name`. Such retrieval can be achieved on sales by the command:

```
SELECT SUM(sales), date.month, date.year,
customer_id, product_name
FROM sales Group BY date.month, date.year,
customer_id, product_name;
```

This type of activities is coined *on-line analytical operations* (*OLAP*), which expresses clearly its objective, i.e. a statistical analysis of data for planning, decision support, and controlling.

As we shall see later there does not exist a clear boundary between retrieval and statistical modeling. However, a statistical function like sum (or average) must be selected for a specific query, which does imply some basic statistical modeling. Consequently, a closed set of operators does not exist for such multi-way tables, cf. consider as an example average (avg), median and geometric or harmonic mean. Moreover, two further problems are involved. First of all, the question arises which data structure is efficient, and secondly, what kind of background information is needed to efficiently access data, to assist the database administrator and the interpretation of real data by the end user? This leads to discuss *metadata* as data about real data and functions involved. Modern database management systems encapsulate metadata in a *repository* (integrated metadata database).

In the following we are first concerned with some fundamentals of database management. Then we turn to the architecture of a statistical database or a data

warehouse and some concepts related to it. We pay special attention to conceptual data structures and their related operators, the summarizability problem, and the specifics of hierarchically structured attributes like time and space. We discuss metadata, access methods and ETL ("extraction, transformation, and loading"). We close with metadata and XML, and privacy.

10.2 Fundamentals of Data Management

We start our discussion with file systems, have a look at database systems useful to store transactions or microdata, and finally turn to data warehouses which host micro and macro data either in a real (materialized) or virtual form.

10.2.1 File Systems

Data is classically stored in *files*. All statistical packages offer this facility. Files can be viewed as a conceptually related set R of records, which are represented by a given record type, see Wirth (1986), and an access mode (direct, indexed or sequential). If the records have a numeric type for each of its fields and the access mode is sequential, then a data matrix can be stored in a sequential file. A collection of such files is called a file system (FS), if there exist logical relations between files $f \in FS$, a set of constraints on FS and application software. Typical applications in statistics are simple surveys like price surveys, where in most cases only one file is needed. A more complex file system is compulsory if, for instance, stratified or panel sampling designs are considered, where various sampling periods, areas, objects and units (carriers of interest) are involved. Note, that relational data mining as described by Dzeroski and Lavrac (2001) and Wrobel (2001), is devoted to such data structures.

File systems are appropriate if only single user-access and weakly logically connected files with simple constraints are effective. Note, that application programs must be specially tailored to execute queries, and to achieve data safety and security. This implies data dependence between the application software and the files referenced, which reduces the program's flexibility with respect to structural changes of the data structure and the ease of "on-the-fly" querying. These pitfalls can be overcome by database systems.

10.2.2 Relational Database Systems

Multi-user access, complex data structures, data independence, disclosure control and logical constraints ask for a *relational database system* (*RDBS*). It consists of

a set T of relations (flat tables) together with a set S of corresponding schemas and a set C of constraints, a database management system (DBMS) and application software. A database schema describes the attributes (variables) of a specific table, its data types and roles. It defines further the underlying constraints which represent either schema based or semantic restrictions. For instance, there are entity or key and referential integrity constraints as well as business rules to be considered, see Elmasri and Navathe (2006). To avoid redundancy and anomalies during insert, delete or update transactions, those tables should be transformed into a "normal form", see Elmasri and Navathe (2006). As an example, we take a Census. When we look at the RDBS 'Census' from a conceptual point of view, there are four table schemas involved: Census-questionnaire, household, dwelling, and employment. We shall consider only the first two in some detail, and select only some few attributes for the sake of brevity. The first schema is

`census_questionnaire(case_id, household_id, age-group, gender, profession,...).`

Its first four attributes are numeric and the fourth one is of type "string". The attribute `case_id` acts as a primary key. Therefore it is underlined. The remaining attributes of this table are functionally dependent on it. Because a key attribute uniquely identifies any tuple (record) of the corresponding table (set of tuples), there is one (entity integrity) constraint among others saying that duplicates in a given table are not allowed. Note, that attribute `household_id` acts as a foreign key because it is defined as a primary key in the relation `household`. In order to mention just one semantic constraint, the domain of the identifier `case_id` may be restricted to the set of non-negative integers, i.e. cardinals.

The other schema is

`household (household_id, case-id, role,...).`

The first two attributes have a numeric domain, while `role` is of type "string" with the value set }"member", "owner"}. Of course, we have again the constraint on `household_id` that duplicates are not allowed, but we need at least one further restriction to ensure reference integrity, i.e., whenever there exist entries of people grouped together in a household, each of their corresponding records in `census_questionnaire` has to exist.

Last but not least, we reconsider our sales example from the introduction. The schema is

`sales (transaction_id, customer_id, date, product_name, quantity, price, amount)`

The primary key is `transaction_id`, which implies that only one product can be part of any sales transaction. Evidently, this scheme is not normalized, because `price` depends on `product_name` besides of `transaction_id`, and we have the balance equation `amount=quantity` $\times$ `price`. The relation itself is of degree (number of attributes) seven. The six attributes `(customer_id, date, ..., amount)` span a six-dimensional data space, where each tuple has six data items and is identified by its corresponding `transaction_id`. We represent four tuples only in the table below to illustrate the difference between

a schema and its corresponding relation (Table 10.1). We use abbreviations in the header of the table sales.

The need of various users for different data can be satisfied by the concept of virtual relations (views), which can be created by an appropriate external schema of a given user coherently related to the underlying conceptual schema of the database system.

Note that the term "table" used in a relational database to store such information is quite different from the tables statisticians use for the same purpose. Table 10.2 shows the representation of the same information in a different table structure that allows the natural computation of aggregates along rows and columns ("marginal sums" etc.). Note that this table structure cannot be mapped directly into a relational database context due to the margins (total or ALL), see Gray et al. (1996).

Let us close this example with a discussion of the background information needed. We mentioned above metadata like schema names, attribute names, data types, roles (key vs. non key, null values allowed vs. not allowed) of attributes, constraints etc. All this can be considered as technical metadata. Moreover, we need further metadata of a semantic and statistical type. Take for instance the attributes `quantity`, `price`, and `amount`. What is their definition? As far as amount is concerned we have "`amount=quantity × price`". Furthermore, we need the corresponding *measurement units* which may be units, *e*/unit, and *e*. As far as data collection at Statistical Offices is concerned, we may need information about the *periodicity* of data surveys like "annual", "quarterly" or "monthly". With respect to data analysis we may be interested in the *measurement scale*. While

Table 10.1 Relational table `sales` of degree 7 and cardinality 4

Transaction_id	Customer_id	Date	Product_name	Quantity	Price	Amount
015	A	4 Jan 97	Tennis shoes	200	95	19,000.00
018	A	4 Jan 97	Tennis balls	300	1.50	450.00
004	A	3 Jan 97	Tennis nets	350	27	9,450.00
009	C	3 Jan 97	Tennis shoes	100	95	9,500.00
…	…	…	…	…	…	…

Table 10.2 `Sales` data in the form of the three-way statistical table `total_sales` with dimensions (`date, customer_id, product_name`)

	Tennis shoes	Tennis balls	Tennis nets
3 Jan 1997			
Customer A	0	0	350
Customer B	0	0	0
Customer C	100	0	0
Total	100	0	350
4 Jan 1997			
Customer A	300	400	450
Customer B	1, 100	1, 100	800
Customer C	600	1, 600	350
Total	2, 350	3, 400	1, 900

product_name has a nominal scale allowing only operations like "equal" and "not equal", the attributes quantity, price, and amount have a metric scale allowing for all basic numerical operations. There exist further ambiguities. For example, the *generation mode* of the attribute sales may have the categories "real", "simulated" or "forecasted". There may further vagueness exist about sales of category "real" unless its *update state* is set to "final", and not to "provisional".

10.2.3 Data Warehouse Systems (DWS)

A *data warehouse system* consists of a (replicated) micro database DB, a set of materialized or virtual multi-way tables (*data cubes*) needed to represent macro (pre-aggregated and grouped) data, a *data warehouse management system* (*DWMS*), and a repository, which stores all required technical, statistical and semantic metadata.

As an example of a data cube, we remind the reader of the three-way table presented above:

total_sales (date.month, date.year, customer_id, product_name, sum (sales)).

This table is represented in a relational form, where date, customer_id, and product_name are concatenated as a primary key. These attributes are called *dimensions*. Evidently, the non-key attribute sum(sales) is fully dependent upon this key, i.e. given the values of date.month, date.year, customer_id, product_name there exist one and only one value of sum(sales) if missing values (null values) are excluded.

Views are useful again and can be provided by joining cubes or sub-cubes in combination with table projections to lower dimensions. It is worthwhile considering separately the attributes sum (sales), date and pro-duct_name. The first attribute is sometimes called summary attribute and is composed of the statistical sum applied to the attribute sales, see Shoshani (1997). This operation is feasible because the function sum and the attribute sales have an identical statistical data type, i.e., a metric or cardinal scale. Moreover, the attribute sales is of attribute type flow, but not stock. While summarizing over flows (rates) is reasonable, such an operation over stocks like "monthly number of customers" is nonsense. Evidently, such and further integrity constraints of semantic type must be effective for a *DWS*, in order to protect the naive user from nonsense queries. This is extremely important for data warehousing, because contrary to database queries, the application of statistical functions is an inherent part of any query.

Furthermore, there exists a specific problem related to date. This attribute can be decomposed into month and year but these components are functionally dependent, i.e., for a given month of a calendar year the year is fixed. We thus have (month, year) → year as a functional dependency. Therefore only one dimension called date is used for the two attributes month and year in the data cube above. There may be further temporal levels of the hierarchy *time* like hour, day, month, quarter, and year. Such hierarchical attributes are called taxonomies and need special attention; see Lehner et al. (1998), Lenz and Thalheim (2009). It

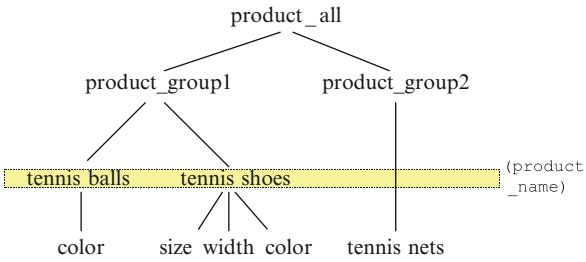


Fig. 10.1 The product taxonomy with a weak functional dependency

is quite remarkable that all dimensions can be allocated to three main groups only: time, location and subject. This is called the 3D-principle, see Lenz (1994).

Let us have a further look at *taxonomie*s that are unbalanced and asymmetric. This may happen in case of a product or regional hierarchy. In our running example the subgroups tennis shoes and balls may be grouped together as product-group 1, while tennis nets build-up product-group 2, but are free of sub-grouping. Both groups 1 and 2 build the root group product-all. As sub-groups exist only for shoes and balls, sub-groups are no longer functionally dependent on `product_name`, but only weakly functionally dependent, see Lehner et al. (1998), Fig. 10.1. This implies that queries, which involve sub-grouping over products, are not feasible and must be refused. Further pitfalls of operations on a data-cube are given in Lenz and Shoshani (1997) and Lenz and Thalheim (2001).

It becomes evident that real data without metadata is more or less useless especially for on-line analytical processing (OLAP). A repository with metadata is a prerequisite of any OLTP or OLAP DBS engineering and any sound data analysis or data mining.

10.3 Architectures, Concepts, and Operators

We first consider the architecture of micro or operational data used for on-line transaction processing (OLTP), and then illustrate the different architecture of macro or analytical data used for decision support and its relation to operational data, see OLAP. We note that the key features of a DBS for OLTP data are: transaction-oriented, measurement- or record-based, real time processing of inserts, deletes and updates of records. In contrary, a DWS for OLAP data is characterized by the features: subject-oriented, integrated and aggregated, calendar or fiscal period related, and non-volatile, see Inmon (1992).

10.3.1 Architecture of a Database System for OLTP

The architecture of database system (DBS) can be represented by the quintuple (data sources, application server, DB server with a DBMS, application server, DB

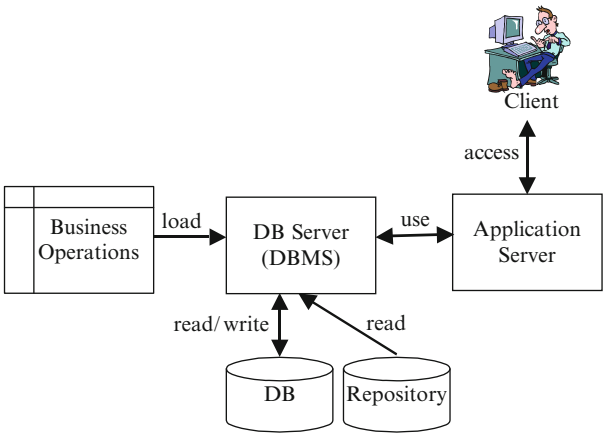


Fig. 10.2 Architecture of a DBS used to manage and query operational data

and repository); see also Fig. 10.2. As mentioned above, business processes act as data sources in commercial systems, while at statistical offices data is supplied by surveys, periodic reports or a census. Similarly, in science the data is generated by observations or measurements collected by operating sensors, field or simulation experiments. We represent the architecture in Fig. 10.2.

As an example from business we consider a company, which manages wages and salaries of its employees. The data is generated by bookkeeping, the DBMS administers the real and metadata, processes queries, and controls transactions. The application server is responsible for running the software for wage and salary computation, while the client is used as a presentation layer for the employees according to their access rights.

10.3.2 Architecture of a Data Warehouse

The main components of the architecture of any OLAP application are heterogeneous data sources S like internal or external databases or files, an OLAP server with DWMS, DW, Repository and Data Marts, and OLAP clients. The DWMS is responsible for the load management, query management and warehouse management.

The data warehouse (see Fig. 10.3) incorporates data replications, archived data and aggregated data stored as data cubes. The departmental view on the whole data is given by subsets of the data cube, called data marts.

As can be seen from Fig. 10.3, analytical processing is concerned with data from various data sources, i.e., external or internal (operational) data. These sources are integrated by ETL in data marts in a unified manner. The data marts can be viewed as collections of data cubes.

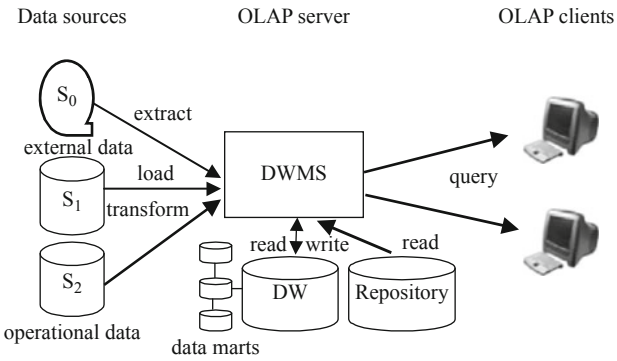


Fig. 10.3 DW architecture

There exist two types of OLAP clients:

(i) stand-alone applications like spreadsheets with a DW interface, and
(ii) Web clients that use an Internet browser and some applets.

10.3.3 Concepts (ROLAP, MOLAP, HOLAP, Cube Operators)

As we have seen above, the schema of a data cube consists of a cube identifier (name), a list of identifying attributes called dimensions and -optional- a statistical function like min, max, count (frequency), sum, avg (arithmetic mean) applied to a summary attribute. Furthermore, the data types and roles of the attributes and integrity constraints must be given. As an example we take from above the data cube "sales cross-classified by month and year, customer and product":

`total_sales (date.month, date.year, customer_id, product_name, sum(sales)).`

Evidently, the dimensions span a three-dimensional space on which the statistical function sum `(sales)` is defined. The corresponding data types are date (mm, yyyy), integer, string and decimal.

Relational OLAP (ROLAP)

In the following we map the conceptual schema of a data cube into a relational database schema. This approach is called *ROLAP* for Relational OLAP, see Raden (1996). There exist two schemas, star and snowflake schemas. As illustrated in Fig. 10.4, the star schema refers to two types of corresponding tables:

1. A *fact table* with a primary key reference to each dimension and the facts which are composed of at least one statistical function and –optional- a summary attribute.

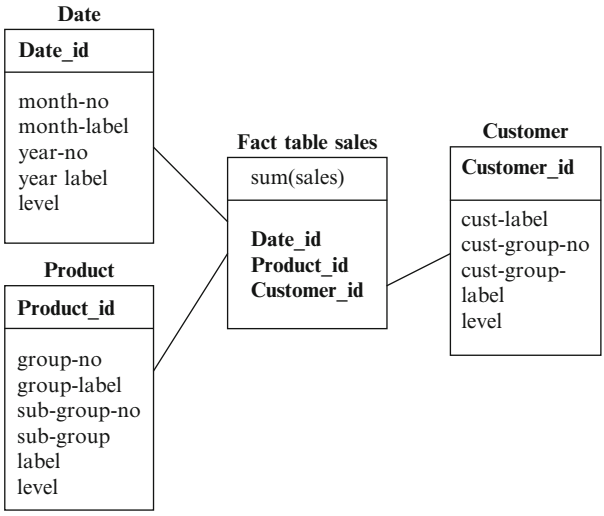


Fig. 10.4 Star schema of a three-dimensional data cube
(one fact table, three dimension tables; the product hierarchy is assumed to have two levels)

2. A *dimension table* for each dimension with a primary key and a level indicator for each entry of a hierarchical attribute.

The star schema models all kind of hierarchical attributes including parallel hierarchies see Lehner et al. (1998). The schema is not normalized as becomes obvious, for example, from the dimension table `Date`. The attributes `month` and `year` are nested, which implies some redundancy. For small or medium-sized data volumes, such schemas have a sufficient performance because join operations are only necessary between the fact table and the related dimension tables.

In order to normalize tables by level attributes, the snowflake schema was introduced. Instead of modelling each dimension by one table, a table is created for each *level* of a hierarchical attribute. The schemas involved are related by identifiers, which play the role either of a primary or a foreign key. In Fig. 10.5 we display only the normalized dimension tables `Month` and `Year` and the fact table `Sales`. The identifiers are `month-no` in the fact table and dimension table Month and `year-no` in the dimension table `Year`.

It can be shown that the normalization is lossless by applying an inner join to the tables of a snowflake schema.

Other Storage Modes (MOLAP, HOLAP)

The above conceptual model of a star or snowflake schema may lead to the wrong conclusion that data cubes are exclusively represented by a relational data model approach. There exist further storage modes, which are in use.

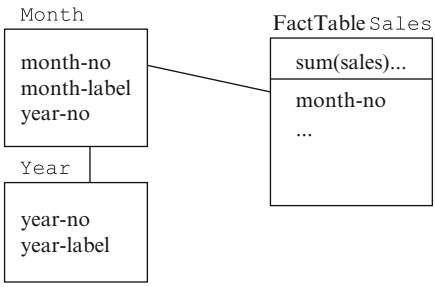


Fig. 10.5 Data cube sales represented (fractionally) as a snowflake schema

The main advantage of ROLAP lies in the reliability, security and ease of loading of the data warehouse based on *Relational DBMS (RDBMS)* technology. As was mentioned above, this is achieved due to the mapping of facts into a normalized relation and dimension into a mostly non-normalized relation of a relational database. As the set of statistical functions in SQL is too restrictive, some of the functionality of OLAP must be added to the application server. An example is to find the *top-ten* among all products sold in a given period.

Multi-dimensional OLAP (MOLAP) makes use of specially tailored data structures like arrays and associated dimension lists or bitmaps. The operational data is extracted and stored as aggregates in those structures. The performance is acceptable for up to medium-sized data sets (<1 Gbyte). There exists a multi-dimensional query language called MDX (multidimensional expressions), see Microsoft 1998. "XML for Analysis" defines a standardized programming interface for an OLAP server, see http://www.xmla.org. An OLAP client encodes a query of a data cube and inserts it into a XML document, which specifies the method "execute" and the accompanying parameters according to the "Simple Object Access Protocol" (SOAP). This document is transmitted over the Internet based on the "Hypertext Transfer Protocol" (HTTP). After decoding the OLAP server executes the query, and sends the data back in a XML document to the client according to SOAP. For further details see Messerschmidt and Schweinsberg (2003). MOLAP has the disadvantage of "miss hits" if a data cube cannot be stored fully in-core and an access to a second storage device is necessary. Moreover, array compression or sparse array handling is needed because mostly the data cube or, equivalently, the arrays are sparse.

Hybrid OLAP (HOLAP) tries to combine the advantages of relational and multi-dimensional database technology. The relational model is used to store replicated and low-level aggregates, while the multi-dimensional model is responsible for high-level aggregates.

Data Cube Operators

Data cubes are used for analytical purposes and not for (simple) transaction processing. Therefore a clear boundary does not exist between data extraction or retrieval and data analysis. This implies that there does not exist a minimal, closed

and complete set of OLAP operators. The mostly built-in operators on data cubes in commercial DWs are the following; see Shoshani (1997), Jarke et al. (2000), and Lenz and Thalheim (2009).

Slicing $\sigma_c(T)$ is to select data from a cube T according to a fixed condition c. This operation is called in Statistics conditioning if only frequencies (counts) applied to multi-way tables are considered. For example, we can retrieve data from `total_sales` according to $\sigma_{\texttt{product\_id}}$, `customer_id, month, year=97(total_sales)`.

Dicing $\pi_c(T)$ is table projection on T by selecting a sub-cube T' of some lower dimension c than the original cube T has. This operation is equivalent to marginalization in Statistics, i.e. projection of a data space into a lower dimension. For instance $\pi_{\texttt{date, customer\_id}}$ `(total_sales)` retrieves a sub-cube of total sales cross-classified by `date` and `customer`.

*Table aggregation (*roll-up) and *disaggregation (*drill-down) are meaningful operations on data cubes if at least on dimension is hierarchical. For example $\rho_{\text{year,customer\_}id\text{,product\_}id}$ `(total_sales)` is a query for less fine-grained data, i.e. for `years` and summarizing over all `months` per year. This specific operation is called *temporal aggregation*. We observe that such an operation is not allowable if a type conflict happens with respect to the summary attribute. This is the case if the attribute "sales" is substituted by "no of employees", see Lenz and Shoshani (1997).

Drill-across $\delta_{\text{level, node, attribute}}(T)$ is a navigation on the same level through the various sub trees of a hierarchical attribute starting at a given node. For example, retrieving products from level 1 (product–group) with start at product group 1 (shoes and balls) of the taxonomy "Product" delivers data about tennis nets.

In order to compute ratios, products etc. of data cubes the *join operator* $\gamma_{\otimes}(T_1, T_2)$ is needed. For instance, as `sales=turnover` × `price` we have `sales=`γ`(turnover, price)`.

We note that there exist further operators like `pivot` (rotation of a cube), see Jarke et al. (2000), or `cube`, which was introduced by Gray et al. (1996). It delivers the margins *ALL* for any subset of dimensions.

10.3.4 Summarizability and Normal Forms

The main objective of *summarizability* is to guarantee correct results of the cube operation `roll-up` and the utilization of statistical (aggregation) functions like `min, max, avg, sum` and `count` under all circumstances, see Lenz and Shoshani (1997). The corresponding integrity constraints are based on *non-overlapping levels of dimensions*, *completeness* and *type compatibility*. The first condition assures that each node of taxonomy has at most one preceding node except for the root node. The second one ascertains that any node on a low level granularity is directly linked to at least one node of a higher granularity. Type compatibility guarantees that the application of any statistical function to a summary attribute is feasible from a statistical point of view. In a preceding section we mentioned the

unfeasibility of aggregation of stocks over time. Another example is the misuse of the sum operator applied to numerically coded professions.

As Lehner et al. (1998) pointed out, the integrity constraint of completeness may turn out to be too restrictive. This happens if structural missing values (null values) in taxonomies exist. For example, the German state *Bavaria* is divided into regions called "Kreise". *Berlin* is a city as well as an autonomous German state. It is not divided into regions, but into suburbs called "Bezirke". In such cases a context sensitive summarizability constraint is appropriate. The authors consequently proposed three multi-dimensional normal forms for fact tables. Lechtenbörger and Vossen (2001) improved the design of such normal forms.

10.3.5 Comparison of Terminologies

To sum up this chapter, the following tables compare the terminology of statistical databases and OLAP, see Shoshani (1997) (Tables 10.3 and 10.4).

10.4 Access Methods

10.4.1 Views (Virtual Tables)

Statistical databases are often accessed by different users with different intentions and different access rights. As already indicated in Sect. 10.2.2, these different requirements can be accounted for by using *views*. These views are derived *virtual*

Table 10.3 Comparison of concepts

Statistical databases	OLAP
Categorical attribute	Dimension
Structural attribute	Dimension hierarchy
Category value	Dimension value
Summary attribute	Fact
Multi-way table	Data cube
Cross product	Multidimensionality

Table 10.4 Comparison of operators

Statistical databases	OLAP
Table projection	Dice
Table selection	Slice
Table aggregation	Roll-up
Table disaggregation	Drill-down
Table join	term missing
Term missing	Drill across
Viewing	Pivoting

tables, which are computed from (actually stored) base tables; see Elmasri and Navathe (2006). There are two main purposes for the use of views.

1. It makes the use of the DBS or DW more convenient for the user by providing only customized parts of the whole data cube.
2. It enforces security constraints by restricting operations on the base tables and by granting users access to their specific views only.

The following SQL statement creates a view for the manager of the product "Tennis Nets" from our example in Table 10.1. It only permits to look up the revenues for Tennis Nets while for all other products, viewing the sales and modifying the corresponding base tables is not possible.

```
CREATE VIEW tennis_nets_manager AS
SELECT date.month, date.year, customer_id,sum(sales)
FROM total_sales WHERE product_name=''Tennis Nets'';
```

Views can never contain information that is not present in the base tables. This is true because the database system translates all view queries into equivalent queries that refer only to the given base tables.

Base tables of a data warehouse may contain billions of tuples. Scanning these tables can be time-consuming and may slow down the interaction between the decision support system and the user significantly. One strategy to speed up the access to aggregated data is to pre-compute a range of probable queries and to store the results in *materialized* views, see Gupta et al. (1997). The access to these materialized views is then much faster than computing data on demand. Yet there are drawbacks to this strategy. The pre-computed data need space, the prediction of the users' queries may be difficult, and each change in the base table requires an update of the materialized view, too. Furthermore, real-time data warehousing as being mandatory in RFID technology increases the existence of computing power for synchronized updates. This is known as the *view maintenance problem*, see Huyn (1997).

10.4.2 Tree-based Indexing

The tables of a DW can physically be accessed either by a sequential scan or by random access. With today's hard disk technology, a sequential scan is 10 to 20 times faster than random access, see Jürgens (2002). That means if more than approximately 5% to 10% of the data has to be accessed it is faster to scan the entire table than addressing specific tuples via random access. In order to avoid full table scans, the number of tuples involved in the result computation has to be reduced. This can be achieved via *index structures,* which permit a fast look-up of specific tuples.

The best-known index structure for one-dimensional data (i.e. data with just one key such as `product_name`) is the *B-tree*; see Bayer and McCreight (1972),

Comer (1979). Pointers to the data items are stored in the leaf nodes of a balanced tree. The B-tree is a very general and flexible index structure, yet in some specific cases it may be outperformed by different kinds of hashing, see Gaede and Günther (1998).

The *universal B-tree* (*UB-tree*, see Bayer 1997) is an extension of the B-tree for indexing multidimensional data such as `total_sales (date.month, date.year, customer_id, product_name, sum(sales))`. The approach partitions the multidimensional data space into squares each of which is captured by a space-filling Z-curve, see Fig. 10.6. For each record, the Z-address of the square, which contains the key values is computed. These Z-addresses are one-dimensional and serve as the new primary keys for the records, which can then be indexed with a standard B-tree.

Another approach for indexing multidimensional data is the *R-tree*, see Guttman (1984). It uses rectangles to represent multidimensional intervals. The leaf rectangles correspond to entries in the database. The parent nodes contain all child nodes and the minimal bounding rectangle. The root rectangle covers the entire query space. An example of how to store sales indexes in an R-tree when `product_name` and `customer_id` build the concatenated primary key is shown in Fig. 10.7. The minimal bounding rectangle of the dashed-line rectangles A, B, and C constitutes the entire search space.

Refinements are the R^+-tree of Sellis et al (1985), the R^*-tree of Beckmann et al. (1990) and a slightly improved version called R_a^*-tree of Jürgens (2002).

10.4.3 Bitmap Index Structures

An important alternative to tree index structures is *bitmap indexing*. For each value of an attribute, a bitmap vector indicates whether or not it is assumed in the records of the table, see Chan and Ionanidis (1998), O'Neil and Quass (1997),

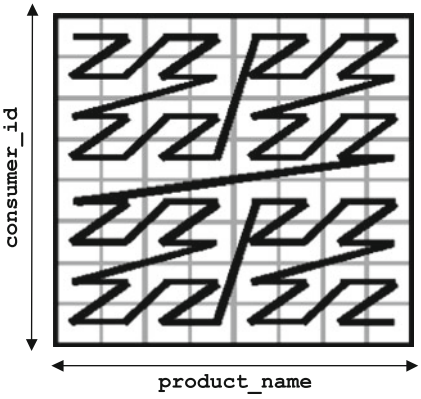


Fig. 10.6 The UB-tree: partition and capture of multidimensional space with the Z-curve

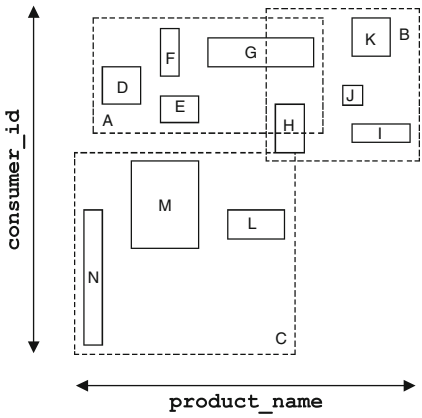


Fig. 10.7 An exemplary R-tree

Table 10.5 Bitmap index for the attribute `product_name`

Transaction_id	Tennis balls	Tennis nets	Tennis shoes
015	0	0	1
018	1	0	0
004	0	1	0
009	0	0	1

Wu and Buchmann (1998). Table 10.5 shows a bitmap index for the attribute `product_name` corresponding to the example presented in Table 10.1. There are three bitmaps needed to represent each value of attribute `product_name`.

The bitmap vector for the attribute value `Tennis Balls` is $(0, 1, 0, 0)^T$. Such a set of bitmap vectors is created for all dimensions. In our `total_sales` example, bitmap indexes have to be created further for `(date.month, date.year)` and `customer_id`.

The size of the bitmap index depends on the number of tuples and on the cardinality of the attribute domain. The required operations on bitmaps are simple and therefore very fast. Thus loading blocks from disc and performing the basic Boolean operations is efficient, especially if the number of dimensions is high, see Jürgens (2002). As bitmaps are often sparse, they are well suited for compression techniques. This is the reason why many commercial database systems are implemented using bitmaps. However, standard bitmaps indexes become space consuming for high attribute's domain cardinality, and they are not very efficient for (low dimensional) range queries, which are typical for DW systems.

Several approaches have been proposed to overcome these drawbacks like the multi-component equality encoded bitmap index, see Chan and Ionanidis (1998). The basic idea is to compress bitmap indexes by encoding all values into a smaller number system by applying modular multiplication. This significantly reduces the space requirements for attributes of high cardinality.

To summarize, bitmaps are more suited for high-dimensional queries with low attribute cardinality. Tree index structures are better for low-dimensional range queries with attributes of high cardinality.

10.5 Extraction, Transformation and Loading (ETL)

ETL is a shorthand notation for a workflow of the initial popularization or a follow-up update of a DW, a data mart or an OLAP application. In the first step data must be extracted from various autonomous, often heterogeneous data sources and temporarily stored in a so-called *staging area* of a DWS. Transformation means to modify data, schema and data quality according to requirement specifications of the DWS. Loading is the integration of replicated and aggregated data in the DW. As the data volume may be huge, incremental loading within pre-selected time slots by means of a bulk loader is appropriate.

10.5.1 Extraction

Extraction can be triggered by events linked to time and state of a running DBS or can be executed under human control. Mostly extraction is deferred according to an extraction schedule supplied by monitoring of the DWS. However, changes of data in the source system are tracked in real time, if the actuality of data is mandatory for some decision makers, see Kimball (1996) or real-time processing based on sensors is involved.

As the data sources are generally heterogeneous, the efforts to wrap single data sources can be enormous. Therefore software companies defined standard interfaces, which are supported by almost all DBMS and ETL tools. For example, the OLE DB provider for ODBC, see Microsoft (1998, 2003), Oracle (2003), and IBM (2003).

10.5.2 Transformation

Transformations are needed to resolve conflicts of schema and data integration and to improve data quality, see Davis and Gather, **Chapter\ref{III.9}**.

We first turn to the first type of conflicts. Spaccapietra et al (1992) consider four classes of conflicts of schema integration, which are to be resolved in each case.

(i) *Semantic conflicts* exist, if two source schemas refer to the same object, but the corresponding set of attributes is not identical, i.e. the class extensions are

different. As an example take two customer files. One record structure includes the attribute name `gender`, while it is missing in the other one.

(ii) A second kind of conflict of integration happens if *synonyms*, *homonyms*, different *data types*, *domains* or *measurements units* exist. For instance, think of the synonym part/article, a homonym like water/money pool, string/date as a domain, and Euro/USD. The ambiguity of our natural language becomes clear when one thinks of the meaning of "name" – family name, nickname, clerical name, former family name, artist name, friar name etc.

(iii) *Schema heterogeneity conflicts* appear if the source schemas differ from the target schema of the DW. For example, sales and departments can be modeled as two relations `Sales` and `Department` of a relational data model or as a nested relation `Department\Sales` as part of an object oriented model. Another kind of conflict corresponds to the mapping of local source keys to global surrogates; see Bauer and Günzel (2001). This problem gets tightened if entity identification is necessary in order decide whether a pair of records from two data sources refer to same entity or not. Fellegi and Sunter (1969) were the first to solve this problem by the record-linkage technique, which is now considered as a special classification method; see Neiling (2003).

(iv) *Structural conflicts* are present if the representation of an object is different in two schemas. There may only one customer schema exist with the attribute `gender` in order to discriminate between "males" and "females". Alternatively, there may be two schemas in use, one linked to "females", the other one to "males".

The second type of conflicts, i.e., conflicts of data integration, happens, if false or differently represented data are to be integrated. False data are generated by erroneous or obsolete entries. Differences in representation are caused by non-identical coding like male/female versus 0/1 or by different sizes of rounding-off errors.

10.6 Metadata and XML

McCarthy (1982) described *metadata* as data about data. However, the technical progress of OLTP and OLAP database systems, workflow techniques and information dissemination has made it necessary, to use a more general definition of metadata.

Metadata is now interpreted as any kind of integrated data used for the design, implementation and usage of an information system. This implies that metadata not only describes real data, but functions or methods, workflows, data suppliers or sources and data receivers or sinks, too. It does not only give background information about the technology of a DBS or DWS, but about its semantic, structure, statistics and functionality. Especially, the semantic metadata enable the

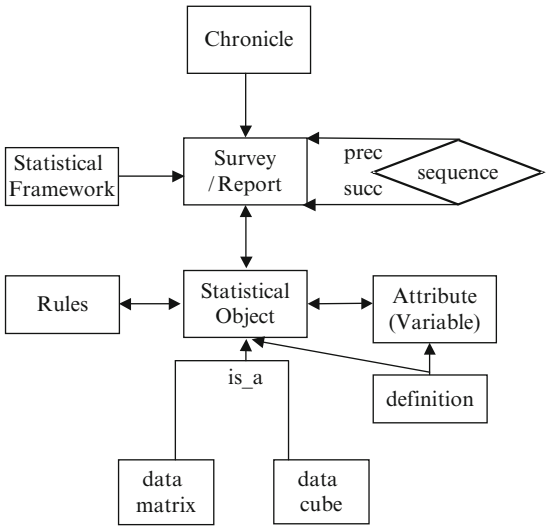


Fig. 10.8 Statistical view of metadata

common user to retrieve definitions of an attribute, to select and filter values of meta attributes, and to navigate through taxonomies.

In Fig. 10.8 we present a view of a conceptually designed metadata. Its core is given by a statistical object, which is either a specialisation of a data matrix or a data cube. It is uniquely described by a definition, and is related in a many to many way to validation and processing rules, surveys or reports and attributes. As we present only a view, no further refinement is given with respect to attributes like role (measure, key, property), scale (nominal, ordinal, cardinal), ontology or even domain (natural, coded) etc. Each statistical object is linked to at least one survey or report. Surveys or reports can be related to a preceding or succeeding one, are related to a statistical framework ("statistical documentation") giving details about the sampling scheme and frame, the corresponding population and statistical estimation methods, and are associated to a chronicle as a calendar of events. Furthermore references to the specific literature and law are allowed. The corresponding substructure is not displayed in Fig. 10.8. For further information about the metadata structure from the user's point of view, see Lenz (1994).

As metadata is stored and can be retrieved similar to real data, it is captured in a *repository* and is managed by a *metadata manager*. A repository can be accessed by users, administrators and software engineers according to their privileges and read/write rights.

Repositories are offered from all vendors. Microsoft (2001) labelled its repository "metadata services", and it is integrated in its SQL server. Alliances were founded to harmonize the metadata models and to standardize the exchange formats. Leading examples are the "Open Information Model" of the "Metadata Coalition (MDC"), see http://www.mdcinfo.com, and the "Common Warehouse Metamodel (CWM)", which was developed by the "Object Management Group" (OMG), see

http://www.omg.org. Since the year 2000 both groups were fused and try to merge their models. Due to the increasing importance of XML and XML databases, import and export format of metadata based on XML is becoming an industrial standard. This happened to OLAP client-server architectures, see "XML for Analysis" as referred in Sect. 10.3.3.

10.7 Privacy and Security

10.7.1 Preventing Disclosure of Confidential Information

The statistical databases that are built by government agencies and non-profit organizations often contain confidential information such as income, credit ratings, type of disease or test scores of individuals. In corporate data warehouses, some strategic figures that are not related to individuals like sales for recently launched products may also be confidential. Whenever sensitive data is exchanged, it must be transmitted over a secure channel like the *Secure Socket Layer (SSL)*, see Netscape (1996) in order to prevent unauthorized use of the system. For the purposes of this chapter, we assume that adequate measures for security and access control are in place, see Stallings (1999).

However, even if the information in the statistical database safely reaches the correct address, the system has to ensure that the released information does not compromise the privacy of individuals or other confidential information. Privacy breaches do not only occur as obvious disclosures of individual values in single queries. Often, the combination of multiple non-confidential query results may allow for the *inference* of new confidential facts that were formerly unknown.

We give an example. From Table 10.1, we take the total sales for Tennis Shoes (28,500), Tennis Balls (450), Tennis Nets (9450) and a fourth, new product (Tennis Socks, 500). We assume that sum queries for groups of products are allowed but that single, product-specific sales values are confidential. After querying the sum for balls and shoes (28,950) and for balls and socks (950), the user can infer an interval of [28,000; 28,950] for the sales of shoes, as sales cannot be negative. The length of the interval, which is the maximum error of the user's estimation of the confidential shoe sales, is only 3.3% of the actual value. This particular case of disclosure is called *interval inference*; see Li et al. (2002). Other types of inference include *exact inference* (concluding the exact value of 28,500 for shoes sales) and *statistical inference* (inferring estimates like mean

$\bar{x}_{\text{Tennis Shoes}} = 30,000$ and standard deviation $s_{\text{Tennis Shoes}} = 5,000$).

If a researcher is granted ad-hoc access to a statistical database, there are basically two different approaches to protect information that is private and confidential from being revealed by a malevolent snooper; cf. Fig. 10.9, and see Adam and Wortmann (1989), Willenborg and de Waal (1996) and Agrawal and Srikant (2000). In the first approach, the kind and number of queries that a researcher poses

to the statistical database (SDB) is restricted (*query restriction)*. In the second approach, the entire database is subject to a manipulation that protects single values but preserves the statistical properties which are of interest to the user. Then the perturbed database can be accessed by a researcher without restrictions (*data perturbation*). In the following, we give an overview of disclosure protection techniques of this kind.

10.7.2 Query Set Restriction

With this approach a query is either denied or responded with an exact answer as the upper sketch in Fig. 10.9 indicates.

Query set size control, Fellegi (1972) works by setting lower and upper bounds for the size of the query answer set based on the properties of the database and on the preferences fixed by the database administrator. If the number of returned records did not lie within these bounds, the information request would have to be rejected and the query answer is denied. As queries that are issued sequentially by one user often have a large numbers of entities in common, an improvement is the restriction of these entities to a maximum number, see Dobkin et al. (1979). Although popular, this method is not robust enough as a stand-alone solution, see Denning (1982).

Auditing involves keeping up-to-date logs of all queries made by each user and constantly checking for possible disclosures whenever a new query is issued. One major drawback of this method is that it requires huge amounts of storage and CPU time to keep these logs updated. A well-known implementation of such an audit system is *Audit Expert* by Chin and Özsoyoglu (1982). It uses binary matrices; see

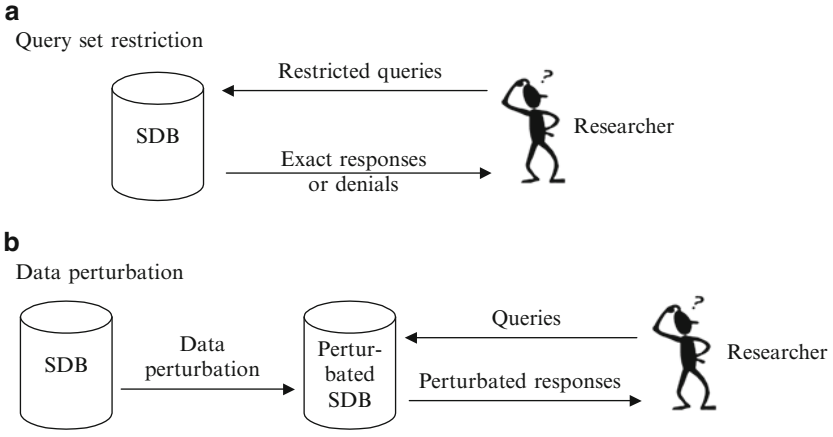


Fig. 10.9 (**a**) Query set restriction and (**b**) data perturbation
Adam and Wortmann (1989)

bitmap indexes in Sect. 10.4.3, to indicate whether or not a record was involved in a query.

Cell suppression, see Cox (1980) is an important method for categorical databases when information is published in tabular form. Especially Census Bureaus often make use of tabular data and publish counts of individuals based on different categories. One of the main privacy objectives is to avoid answers of small size. For example, if a snooper knows somebody's residence, age and employer, he can issue a query for `(ZIP=10,178, age=57, employer="ABC")`. If the answer is one entity, the snooper could go on and query for `(ZIP=10,178, age=57, employer=" ABC", diagnosis="depression")`. If the answer is one again, the database is compromised and the person with the diagnosis identified. The cells should have to be suppressed. A common criterion to decide whether or not to suppress a cell is the *N–k rule* where a cell is suppressed if the top N respondents contribute at least $k\%$ of the cell total. N and k are parameters that are fixed by the database administrator, i.e. the Census Bureau. In the exemplary case of $N = 2$ and $k = 10\%$, a cell which indicates aggregated income ($10M) of 100 individuals would have to be suppressed if the top two earners' aggregate income exceeded $1M.

10.7.3 Data Perturbation

In the query restriction approach, either exact data is delivered from the original database or the query is denied. As depicted in the lower part of Fig. 10.9, an alternative is to perturb the original values such that confidential, individual data become useless for a snooper while the statistical properties of the attribute are preserved. The manipulated data is stored in a second database and is then freely accessible for the users.

If in Table 10.1, we permute the sales of tennis balls, tennis nets, and tennis shoes, individual sales data is not correct anymore. But the arithmetic average and the standard deviation of the attribute sales stay the same. This procedure is called *data swapping*, see Denning (1982).

Noise addition for numerical attributes, see Traub et al. (1984), means adding a disturbing term to each value: $Y_k = X_k + e_k$, where X_k is the original value and e_k adheres to a given probability distribution with mean zero. As for every value X_k value, the perturbation e_k is fixed, conducting multiple queries does not refine the snooper's search for confidential single values.

A hybrid approach are *random-sample queries,* Denning (1982), where a sample is drawn from the query set in such a way that each entity of the complete set is included in the sample with probability P. If, for example, the sample of a `count` query has n entities, then the size of the not perturbed query set can be estimated as n/P. If P is large, there should be a set-size restriction to avoid small query sets where all entities are included.

10.7.4 Disclosure Risk vs. Data Utility

All methods presented in the preceding sections aim at lowering the disclosure risk for data that is private and confidential. But at the same time, each of these methods reduces, in some way, the utility of the data for the legitimate data user. Duncan and Keller-McNulty (2001) present a formal framework to measure this trade-off between disclosure risk and data utility, the Risk-Utility ($R - U$) map. There are numberless measures for disclosure risk, see Domingo-Ferrer et al. (2002) for an excellent overview. We already gave an intuitive measure for interval inference. The sales for tennis shoes were predicted with an error of only 3.3%, see Sect. 10.7.1.

However, it is far more difficult to measure data utility because it strongly depends on the varying preferences of the data user. Especially for this reason, classifying statistical disclosure control methods as presented here on an absolute scale is almost an impossible task.

References

Adam, N., Wortmann, J.: Security-control methods for statistical databases: a comparative study. ACM Comput. Surv. **21**(4), 515–556 (1989)

Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (2000)

Bauer, A., Günzel, H. (eds.): Data warehouse systeme. dpunkt, Heidelberg (2001)

Bayer, R., McCreight, E.: Organization and maintenance of large ordered indexes. Acta Inform. **1**(3), 173–189 (1972)

Bayer, R.: The universal B-tree for multidimensional indexing: general concepts. In: World-wide computing and its applications '97 (WWCA '97). Lecture Notes on Computer Science. vol. 10–11, Springer, Tsukuba (1997)

Beckmann, N., Kriegel, H.P., Schneider, R., Seeger, B.: The R*-tree: an efficient and robust access method for points and rectangles. In: Proceedings of the ACM SIGMOD International Conference on Management of Data, New York (1990)

Chan, C.Y., Ionanidis, Y.E.: Bitmap index design and evaluation. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (1998)

Chin, F.Y., Özsoyoglu, G.: Auditing and inference control in statistical databases. IEEE Trans. Softw. Eng. **8**(6), 574–582 (1982)

Comer, D.: The ubiquitous B-tree. ACM Comput. Surv. **11**(2), 121–138 (1979)

Cox, L.H.: Suppression methodology and statistical disclosure control. J. Am. Stat. Assoc. **75**(370) (1980)

Denning, D.E.: Cryptography and data security. Addison-Wesley (1982)

Dobkin, D., Jones, A.K., Lipton, R.J.: Secure databases: protection against user influence. ACM Transactions on Database Systems. **4**(1), 97–106 (1979)

Domingo-Ferrer, J., Oganian, A., Torra, V.: Information-theoretic disclosure risk measures in statistical disclosure control of tabular data. In: Proceedings of the 14th International Conference on Scientific and Statistical Database Management (SSDBM '02) (2002)

Duncan, G., Keller-McNulty, S.: Disclosure risk vs. data utility: the R-U confidentiality map. Technical Report. Statistical Sciences Group. Los Alamos National Laboratory (2001)

Dzeroski, S., Lavrac N. (eds.): Relational data mining. Springer, Heidelberg (2001)

Elmasri, R., Navathe, S.B.: Fundamentals of database systems. Addison-Wesley (2006)

Fellegi, I.P., Sunter, A.B.: A theory of record linkage. J. Am. Stat. Assoc. **40**, 1183–1210 (1969)
Fellegi, I.P.: On the question of statistical confidentiality. J. Am. Stat. Assoc. **67**(337), 7–18 (1972)
Gaede, V., Günther, O.: Multidimensional access methods. ACM Comput. Surv. **30**(2), 170–231 (1998)
Gray, J., Bosworth, A., Layman, A., Pirahesh, H.: Data cube: a relational aggregation operator generalizing group-by, cross-tab, and sub-total. In: Proceedings of the 12th International Conference on Data Engineering (ICDE'96), pp. 29–53. IEEE Computer Society, New Orleans (1996)
Gupta, H., Harinarayan, V., Rajaraman, A., Ullman, J.D.: Index selection for OLAP. In: Proceedings of the Thirteenth International Conference on Data Engineering (ICDE '97). IEEE Computer Society, Birmingham (1997)
Guttman, A.: R-trees: a dynamic index structure for spatial searching. In: Proceedings of the ACM SIGMOD International Conference on Management of Data (1984)
Huyn, N.: Multiple-view self-maintenance in data warehousing environments. In: Proceedings of the 23rd Conference on Very Large Databases (VLDB), (1997)
IBM: DB2 OLAP Server. http://www-3.ibm.com/software/data/db2/db2olap/library.html (2003)
Inmon, W.H.: Building the data warehouse. Wiley, New York (1992)
Jarke, M., Lenzerini, M., Vassiliou, Y., Vassiliadis, P.: Fundamentals of data warehouses. Springer, Berlin (2003)
Jürgens, M.: Index structures for data warehouses.In: Springer lecture notes in computer science, Berlin (2002)
Kimball, R.: The data warehouse toolkit. Wiley, New York (1996)
Li, Y., Wang, L., Wang, X., Jajodia, S.: Auditing interval-based inference. In: Proceedings of the 14th Conference on Advanced Information Systems Engineering (CAiSE'02), Toronto (2002)
Lehner, W., Albrecht, J., Wedekind, H.: Normal forms for multidimensional databases. In: Proceedings of the 10th International Conference on Scientific and Statistical Data Management (SSDBM'98), Capri (1998)
Lenz, H.J.: A rigorous treatment of microdata, macrodata and metadata. In: Dutter, R. (ed.) Proceedings in computational statistics. Physica, Heidelberg (1994)
Lenz, H.J., Shoshani, A.: Summarizability in OLAP and statistical databases, In: Proceedings of the 10th International Conference on Scientific and Statistical Data Management (SSDBM '97), Washington (1997)
Lenz, H.J., Thalheim, B.: OLAP databases and aggregation functions. In: Proceedings of the 14th International Conference on Scientific and Statistical Data Management (SSDBM '01), Washington (2001)
Lenz, H.J., Thalheim, B.: A formal framework of aggregation for the OLAP-OLTP model, J. Univers. Comp. Sci. **15**(1) (2009)
Lechtenbörger, J., Vossen, G.: Quality-oriented data warehouse schema design. In: Information technology. **45**, 190–195 (2001)
McCarthy, J.: Metadata management for large statistical databases. In: Proceedings of the Eight International Conference on Very Large Data Bases, Mexico City (1982)
Messerschmidt, H., Schweinsberg, K.: OLAP mit dem SQL-Server. dpunkt, Heidelberg (2003)
Microsoft: OLE DB for OLAP programmer's reference (1998)
Microsoft: Microsoft SQL server: data transformation services (DTS). http://www.microsoft.com/sql/evaluation/features/datatran.asp (2003)
O'Neil, P., Quass, D.: Improved query performance with variant indexes. SIGMOD Rec. **26**(2), 38–49 (1997)
Oracle: Oracle warehouse builder – product information. http://otn.oracle.com/products/warehouse/index.html (2003)
Neiling, M.: Identifizierung von Realwelt-Objekten in multiplen Datenbanken. Ph.D. dissertation, University of Cottbus, Germany (2003)
Netscape: Secure Socket Layer 3.0 Specification. http://wp.netscape.com/eng/ssl3/ (1996)
Raden, N.: Star Schema 101. Archer Decision Sciences, Santa Barbara, http://members.aol.com/nraden/str101_e.htm(2000.12.12) (1996)

Shoshani, A.: OLAP and statistical databases: similarities and differences. In: Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGART Symposium on Principles and Database Systems (PODS'97), Tucson (1997)

Spaccapietra, S., Parent, C., Dupont, Y.: Model independent assertions for integration of heterogeneous schemas. VLDB J. **1**(1), 81–126 (1992)

Stallings, W.: Cryptography and network security, principles and practice. Addison-Wesley (1999)

Traub, J.F., Yemini, Y., Wozniakowski, H.: The statistical security of a statistical database. ACM Trans. Database Syst. **9**(4), 672–679 (1984)

Willenborg, L., de Waal, T.: Statistical disclosure control in practice. Springer, New York (1996)

Wirth, N.: Algorithms and data structures. Englewood Cliffs, Prentice Hall (1986)

Wrobel, S.: Inductive logic programming for knowledge discovery in databases. In: Dzeroski, S., Lavrac, N. (eds.) Relational data mining. Springer, Heidelberg (2001)

Wu, M.C., Buchmann, A.P.: Encoded bitmap indexing for data warehouses. In: Proceedings of the 14th International Conference on Data Engineering (ICDE), 220–230 (1998)

Chapter 11
Discovering and Visualizing Relations in High Dimensional Data

Alfred Inselberg

11.1 Introduction

*V*isualization flourished in Geometry. Legend has it that Archimedes was absorbed in a diagram when he was killed by a Roman soldier. "Do not disturb my circles" he pleaded as he was being struck by the sword ... the first reported death in defense of visualization. Visual *interaction* with diagrams is interwoven with the testing of conjectures and construction of proofs. Our tremendous *pattern recognition* enables us to extract insight from images. This essence of visualization is abstracted and adapted in the more general problem-solving process to the extent that we form a mental image of a problem we are trying to solve and at times we say *see* when we mean understand. My interest in visualization was sparked and nourished while learning geometry. Later, while studying multi-dimensional geometry I struggled with the thought of displaying multidimensional geometry and multivariate problems. What emerged is Parallel Coordinates (Inselberg 1985 and earlier).

In the Euclidean plane $\mathbb{R}^2$ with xy-Cartesian coordinates, N copies of the real line $\mathbb{R}$ labeled $\bar{X}_1, \bar{X}_2, \ldots, \bar{X}_N$ are placed equidistant and perpendicular to the x-axis as shown in Fig. 11.1. They are the axes of the **Parallel Coordinates** system for the Euclidean N-dimensional space $\mathbb{R}^N$, all having the same positive orientation as the y-axis. A point C with coordinates $(c_1, c_2, \ldots, c_N)$ is represented by the complete polygonal line $\bar{C}$ (i.e. the *lines* containing the segments between the axes) whose N vertices are at the c_i value on the $\bar{X}_i$-axis for $i = 1, \ldots, N$. In this way, a 1-1 correspondence between points in $\mathbb{R}^N$ and planar polygonal lines with vertices on the parallel axes is established. In principle, a large number of axes can be placed and be seen parallel to each other. The representation of points is deceptively simple

A. Inselberg (✉)
School of Mathematical Sciences, Tel Aviv University, Tel Aviv, Israel
e-mail: aiisreal@post.tau.ac.il

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_11,

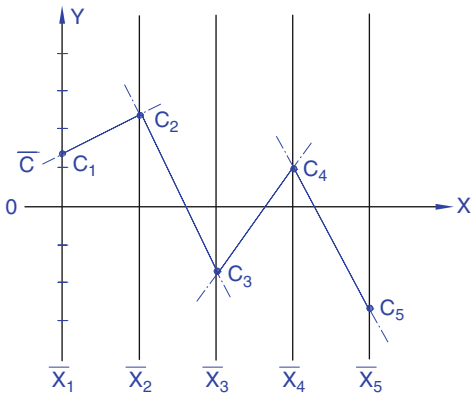


Fig. 11.1 The polygonal line $\bar{C}$ represents the point $C = (c_1, c_2, c_3, c_4, c_5)$

and much development with additional ideas is needed to enable the visualization of multivariate relations or equivalently multidimensional objects.

Many contributed to the development, colleagues and students: B. Dimsdale Inselberg and Dimsdale (1990), A. Hurwitz, T. Chomut Chomut (1987), M. Boz Inselberg et al. (1991), M. Reif Inselberg et al. (1987), P.Fiorini Fiorini and Inselberg (1989), J. Eickemeyer Eickemeyer (1992), C.K.Hung Hung and Inselberg (1992), A. Chatterjee Chatterjee (1995) and T. Mastkewich Matskewich et al. (2000) and others S. Cohan & Yang Cohan and Yang (1986), H.Hinterberger Schmid and Hinterberger (1994), P.Fiorini Fiorini and Inselberg (1989), C.Gennings et al. Gennings et al. (1990), E. Wegman Wegman (1990), A. Desai & L. Walters Desai and Walters (1991) and more. Progress continued A.Chatterjee et al. Chatterjee et al. (1993), M.Ward et al Ward (1994), C.Jones Jones (1996), Inselberg (1997), Inselberg and Avidan (1999) to the more recent work of H. Hauser Hauser (2005), H. Choi and Heejo Lee Choi and Lee (2005), G. Conti Conti (2007), M. Theus and S. Urbanek Theus and Urbanek (2009) and others increased the body of knowledge and versatility of ∥-cs. The last chapter in Inselberg (2009) contains the exciting recent contributions of S. Cohen-Ganor – Displaying Several Lines Efficiently, N. Shahaf – Separating Point Clusters on Different Planes, C. K. Hung – Surface Representation and Developable Quadrics, Y. Singer & O. Greenshpan – Network Visualization and Analysis, and Y. Yaari – To See $\mathbb{C}^2$, The Visualization of Complex Valued Functions. And this list is by no means exhaustive. As of this writing a query for "parallel coordinates" on Google returned about 140,000 "hits".

11.2 Visual Data Mining

The first, and still more widespread, application of parallel coordinates is for exploratory data analysis (EDA). That is, the discovery of data subsets (relations) fulfilling given objectives. A dataset with M items has 2^M subsets anyone of which

may be the one we really want. Our fantastic pattern-recognition ability and a good data display can penetrate this combinatorial explosion by recognizing patterns and the multivariate relations they represent. Extracting *insight from images* is the crux of data visualization.

For the visualization of multivariate problems numerous mappings encoding multidimensional information visually into 2-D or 3-D (see Friendly and al (2005) and Tufte (1996) have been invented to augment our perception, which is limited by our 3-dimensional habitation. Wonderful successes like Minard's "Napoleon's March to Moscow", Snow's "dot map" and others are *ad hoc* (i.e. one-of-a-kind) and exceptional. Succinct multivariate relations are rarely apparent from **static** displays; **interactivity** is essential. Searching a dataset with M items for interesting, depending on the objectives, properties is inherently hard. The *visual cues*, our eyes can pick from a good data display, navigate the knowledge discovery process. Clearly, if the transformation : $data \rightarrow picture$ clobbers information a great deal is lost right at the start. Good displays of datasets with N variables should preserve information and work for any number of variables N. Also they need to be computationally and space efficienct. These considerations limit the use of the scatterplot matrix (abbr. *SM*) and other methods. For our purposes, the crucial value and role of visualization is not seeing "zillions of objects" but rather recognizing **relations** among them.

11.2.1 Exploratory Data Analysis with Parallel Coordinates

Parallel coordinates transform multivariate relations into 2-D patterns suitable for exploration and analysis. The exploration[1] paradigm is that of a *detective*, starting from the data, searching for clues leading to conjectures, testing, backtracking until *voila* ... the "culprit" is discovered. The task is especially intricate when there are many variables (i.e. dimensions).

During the ensuing interaction think, dear reader, how similar queries can be done using other exploration methodologies including the ubiquitous spread-sheets. More important, what visual clues are available that would **prompt** the use of such queries. Recall that in $\|$-cs due to the $point \leftrightarrow line$ and other dualities, some but not all actions are best performed in the dual. The queries, which are the "cutting tools", operate on the display i.e. the *dual*. Their design should exploit the methodology's strengths and avoid its weaknesses; rather than mimic the action of queries operating on standard "non-dual" displays. As a surgeon's many specialized cutting tools, one of our early software versions had lots of specialized queries. Not only was it hard to classify and remember them but they still could not handle all situations encountered. After experimentation, few (3) intuitive **atomic** queries were chosen

[1]The venerable name "Exploratory Data Analysis" *EDA* is used interchangeably with the currently more fashionable "Visual Data Mining".

which can be combined via *boolean* operations to form complex intricate cuts. Even for relatively small datasets the ‖-cs display can look uninformative and intimidating. Lack of understanding the basic underlying geometry and poor choice of queries limits the use of ‖-cs to unrealistically small datasets. Summarizing, the requirements for successful exploratory data analysis are:

- An informative display *without loss of information* of the data,
- Good choice of queries, and
- Skillful interaction with the display.

Aside from starting the exploration without biases it is essential to understand the objectives. The task in the first example is the detection and location of various ground features (i.e. built-up areas, vegetation, water etc) on the map. There is a prominent lake, on the lower-left corner with an unusual shape like an upward pointing “finger”.

11.2.2 An Easy First Study: Satellite Data

The first advice is not to let the picture intimidate you as can easily happen by taking an uninformed look at Fig. 11.4(left) showing the dataset to be explored. It consists of over 9,000 measurements with 9 variables, the first two (X, Y) specify the location on the map in Fig. 11.2(left), a portion of Slovenia, where 7 types of ground emissions are measured by satellite. The ground location, (X, Y), of one data item is shown in Fig. 11.2 (right), which corresponds to the map’s region and remains open during the exploration. The query, shown in Fig. 11.3(left), used to select the data item is called *Pinch*. It is activated by the button **P** on the tool bar. By means of this query, a bunch of polygonal lines (i.e. data items) can be chosen by being “pinched” *in-between* the axes. The cursor’s movement changes the position of the *selected* arrow-head which is the larger of the two shown. In due course various parts of the *GUI* are explained(*Parallax*).[2]

Follow up on anything that catches the eyes, gaps, regularities, holes, twists, peaks & valleys, density contrasts like the one at the lower values of $B3$ through $B7$. Using the *Interval* query, activated by the **I** button, starting at the minimum we grab the low range of $B4$ (between the arrowheads) stopping at the dense part as shown in Fig. 11.3 (right). The result, on the left of Fig. 11.4, is amazing. Voila we found the water[3] the lake is clearly visible together with two other regions which in the map turn up to be small streams. Our scrutiny having been rewarded we recall the adage that a good thing may be worth repeating. Examining for density variations now *within the selected lower interval of* $B4$ we notice another. The lowest part is much denser. Experimenting a bit, appreciating the importance of interactivity, we

[2]MDG’s Ltd proprietary software – All Rights Reserved, is used by permission.

[3]Suggesting that the Landsat Thematic mapper band 4 filters out water though unknown to me.

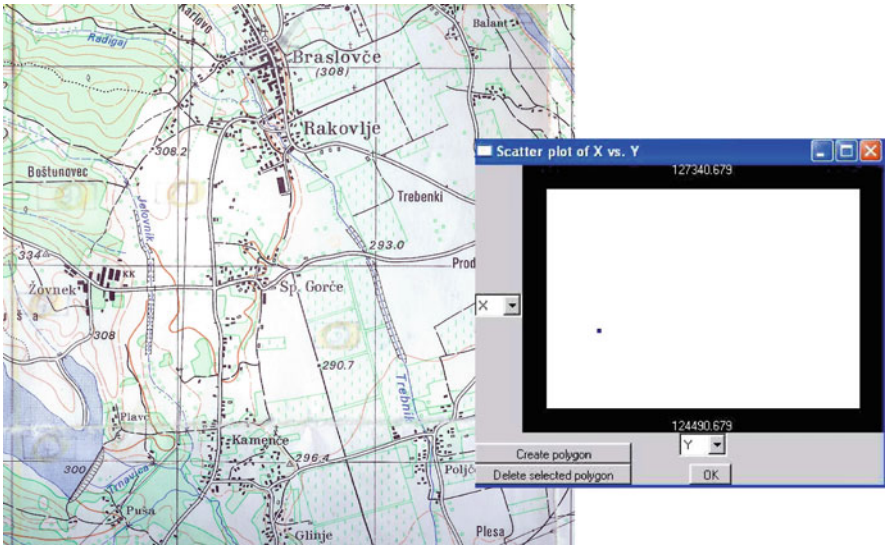


Fig. 11.2 Seven types of ground emissions were measured on this region of Slovenia. Measurements made by the LandSat Thematic Mapper. Thanks and acknowledgement to Dr. Ana Tretjak and Dr. Niko Schlamberger, Statistics Office of Slovenia, for providing the data. (*Right*) The display is the map's rectangular region. The dot marks the position where the 7-tuple shown in the next figure was measured

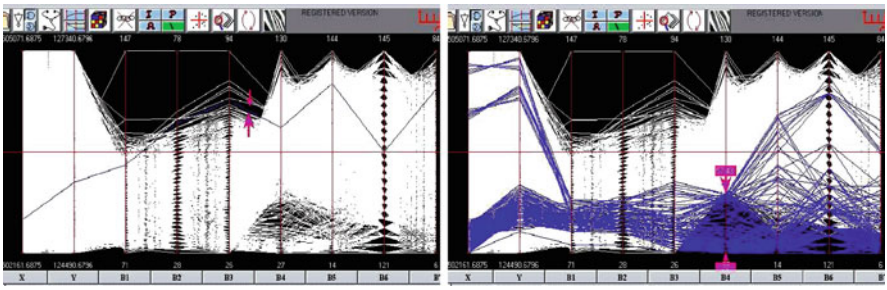


Fig. 11.3 (*Left*) Query selecting a single data item. (*Right*) Finding water regions. (*Left*)The X, Y (position, also shown on the right of Fig. 11.2), and values of the 7-tuple ($B1$, $B2$, $B3$, $B4$, $B5$, $B6$, $B7$) at that point. (*Right*)The contrast due to density differences around the lower values of $B4$ is the *visual cue* prompting this query

select the sparse portion, Fig. 11.5, which defines the water's edge (right) 11.4 and in fact more. By dropping the lower arrow we see the lake filling up starting from the edge i.e. shallow water first. So the lower values of $B4$ reveal the water and the lowest "measure" the water's depth; not bad for few minutes of playing around.

But all this pertains to a single variable when we are supposed to be demonstrating *multivariate* exploration. This is a valid point but we did *pick* $B4$ among

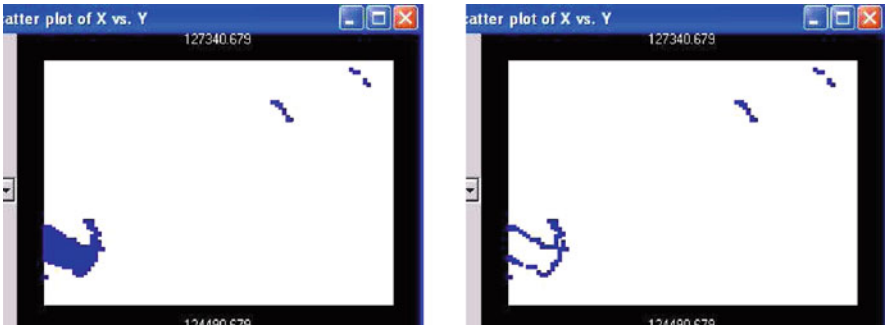


Fig. 11.4 (*Left*)The lake – result of query shown in Fig. 11.4 (*Right*).
On the right is just the lake's edge. It is the result of query shown in Fig. 11.5

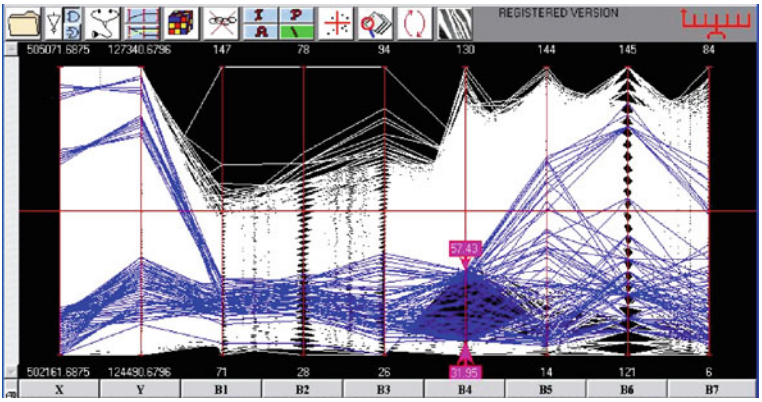


Fig. 11.5 Query finding the water's edge

several variables. Further, this is a nice "warm-up" for the subsequent more involved examples enabling us to show two of the queries. The astute observer must have already noticed the regularity, the vertical bands, between the $B1$, $B2$ and $B3$ axes. This is where the *angle* query, activated by the **A** button, comes into play. As the name implies it selects groups of lines within a user-specified angle range. A data subset is selected between the $B2, B3$ axes as shown, with enlarged inter-axes distance better showing the vertical bands, in Fig. 11.6 (left) to select a data subset which corresponds on the map to regions with high vegetation. Clicking the **A** button and placing the cursor on the middle of one axis opens an angle, with vertex on the mid-range of the previous(left) axis, whose range is controlled by the arrow movements on the right axis. Actually this "rule" (i.e. relation among some parameters) for finding vegetation can be refined by twicking a couple of more parameters. This raises the topic of rule finding in general, *Classification*, which is taken up in Sect. 11.3.

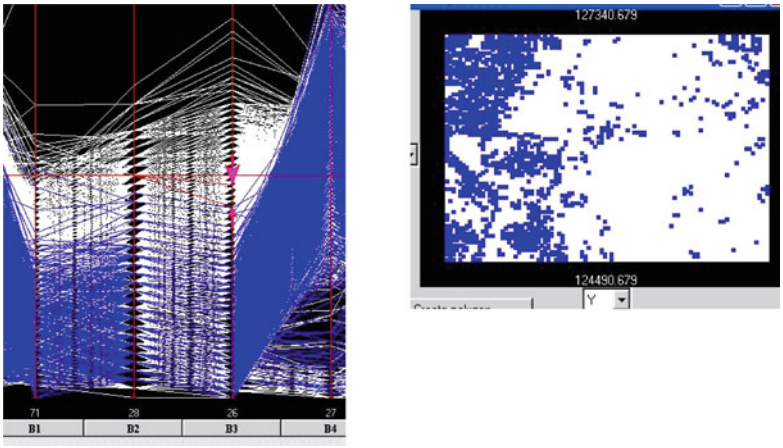


Fig. 11.6 Finding regions with vegetation

The *angle* and *pinch* queries are motivated by the ℓ *line* $\rightarrow$ *point* $\bar{\ell}$ duality

$$\ell : x_2 = mx_1 + b \leftrightarrow \bar{\ell} = (\frac{d}{1-m}, \frac{b}{1-m}) \tag{11.1}$$

in $\|$-coords illustrated in Fig. 11.7 where the inter-axes distance is d. As seen from its x-coordinate, the point $\bar{\ell}$ lies between the parallel axes when the line's slope $m < 0$, to the right of the $\bar{X}_2$ axis for $0 < m < 1$ and left of $\bar{X}_1$ for $m > 1$. Lines with $m = 1$ are mapped to the *direction* with slope b/d in the on the xy-plane; with d the inter-axes distance and b the constant (intercept) in the equation of ℓ. This points out that dualities properly reside in the *Projective*, the *directions* being the *ideal points*, rather than the Euclidean plane. For sets of points having a "general" direction with negative slope, i.e. are "negatively correlated", the lines representing them in $\|$-cs cross each other in between the axes and they can be *selected with the pinch query*. For positively correlated sets of points their corresponding lines cross outside the axes and can be *selected with the angle query*. All this exemplifies the need to understand some of the basic geometry so as to work effectively with the queries and of course, at first, design them well. The three atomic queries having been introduced there remains to learn how they can be combined to construct complex queries.

Prior to that, Fig. 11.6 (left) begs the question: "what if the $B2$ and $B3$ axes were *not* adjacent"? Then the pattern and hence their pairwise relation would be missed. Clearly the axes-permutation used for the exploration is important. In particular what is the minimum number of permutations among N-axes containing the *adjacencies* for all pairs of axes? It turns out that M permutations are needed for even $N = 2M$ and $M + 1$ for odd $N = 2M + 1$. It is fun to see why. Label the N vertices of a graph with the index of the variables X_i, $i = 1, \ldots, N$

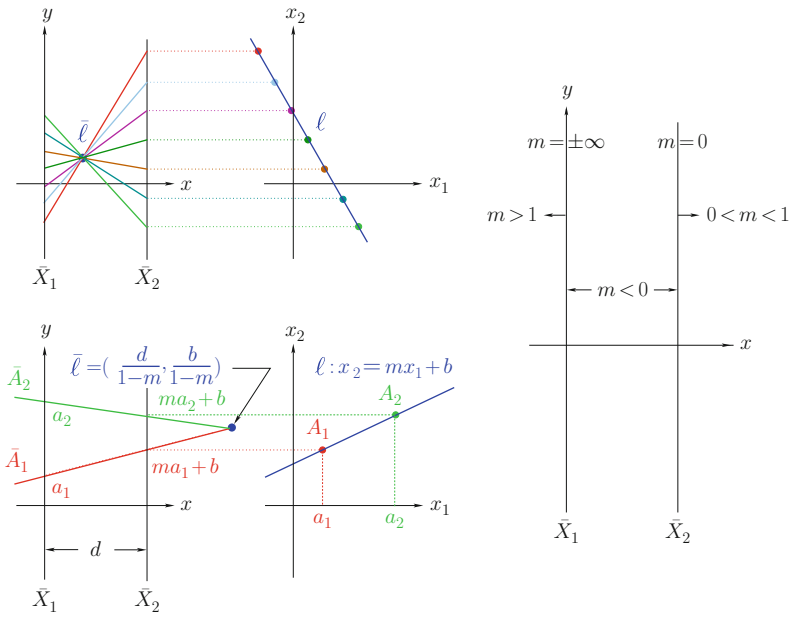


Fig. 11.7 Parallel coordinates induce a *point* $\bar{\ell} \leftrightarrow \ell$ *line* duality (*left*). (*Right*) The horizontal position of the point $\bar{\ell}$ representing the line ℓ is determined only by the line's slope m. The vertical line $\ell : x_1 = a_1$ is represented by the point $\bar{\ell}$ at the value a_1 on the $\bar{X}_1$ axis

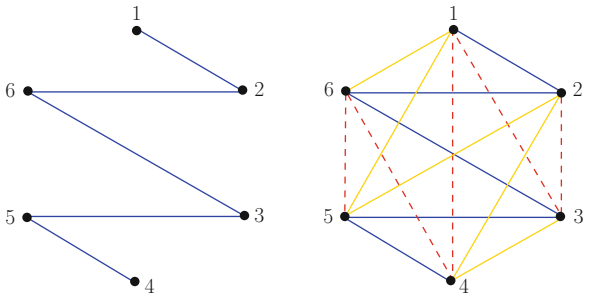


Fig. 11.8 (Left) Graph corresponds to the (*axes*) index permutation **126354**. (*Right*) The complete graph as the union of the 3 distinct Hamiltonian paths starting successively at the vertices **1, 2, 3**

as shown in Fig. 11.8 for $N = 6$. An edge joining vertex **i** with **j** signifies that the axes indexed by **i , j** are adjacent. The graph on the left is a *Hamilton path* for it contains all the vertices. Such paths have been studied starting with Euler in the eighteenth century with modern applications to the "travelling salesman" problem and elsewhere (Harary 1969 pp. 66, Bollobas 1979 pp. 12). The graph corresponds to the axes index permutation **126354**. On the right, the union with the additional two Hamiltonian paths, starting at vertices **2** and **3**, forms the complete graph which contains all possible edges. Hence the 3 permutations **126354, 231465, 342516** contain all possible adjacent pairs; just try it. The remaining permutations

are obtained from the first by successively adding *1 mod 6* to each digit and this works for general N. This early result and recent ones appears in the comprehensive paper by C. Hurley and W. Olford Hurley and Olford (2010). An important more general question, let's call it the *the triad problem*, is "what is the minimum number of permutations needed containing all adjacent triple variables displays?" Clearly this is relevant for in ‖-cs adjacent order-independent triples appear, revealing their interelations, and can lead to the faster discovery of more global multivariate relations. This problem may be of interest in *social networks* in finding common friends between pairs of members. Apparently it is an open question and hopefully this discussion may motivate and inspire readers to contribute a solution.

Returning to EDA, the icon with the *Rubik's Cube* on *Parallax*'s toolbar activates a *permutation editor* which automatically generates the Hamiltonian permutations (abbr. *HP*). After scrutinizing the dataset display the recommended next step is to run through the $O(N/2)$ *HP*. This is how all nice adjacencies such as the one in Fig. 11.6 are discovered. Then using the editor, patch your own custom-made permutation containing all the parts you like in the *HP*. With this preprocessing cost the user sets her own best permutation to work with. Of course, there is nothing to prevent the inclusion of axes several times in different positions as well as experimenting with different permutations in the course of the exploration.

11.2.3 Compound Querries: Financial Data

$\mathcal{T}$o be explored next is the financial dataset shown in Fig. 11.9, the goal being to discover relations useful for investments and trading. The data for the years 1986 (second tick on the 3rd axes) and 1992 are selected and compared. In 1986 the **Yen** had the greater volatility among the 3 currencies, interests varied in the mid-range, gold had a price gap while **SP**500 was uniformly low. By comparison in 1992, the **Y**en was stable while the **S**terling was very volatile (possibly due to Soros' speculation that year), interests and gold price were low and the **SP**500 was uniformly high. Two **I**nterval queries are combined with the *OR* boolean operator (i.e. Union) to obtain this picture.

We continue "looking for the gold" by checking out patterns that caught our attention. The data for 1986 is isolated in Fig. 11.10 and the lower range in the gold price gap is selected. Gold prices were low until the 2nd week in August when they jumped and stayed higher. The exploration was carried out in the presence of four financial experts who carefully recorded the relation between low **Yen**, high **3M**TB rates and low **G**old prices. By the way, *low* **Yen** rate of exchange means the Yen has high value relative to the US $.

There are two bunches of crossing lines between 6th and 7th axes in Fig. 11.9 which together comprise more than 80% of the dataset. This and recalling the previous discussion on the *line* ← *point* mapping in Fig. 11.7 points out the strong negative correlation between **Yen** and **3M**TB rates. The smaller cluster in Fig. 11.11 (left) is selected. Moving from the top range of any of the two axes, with the **I** query,

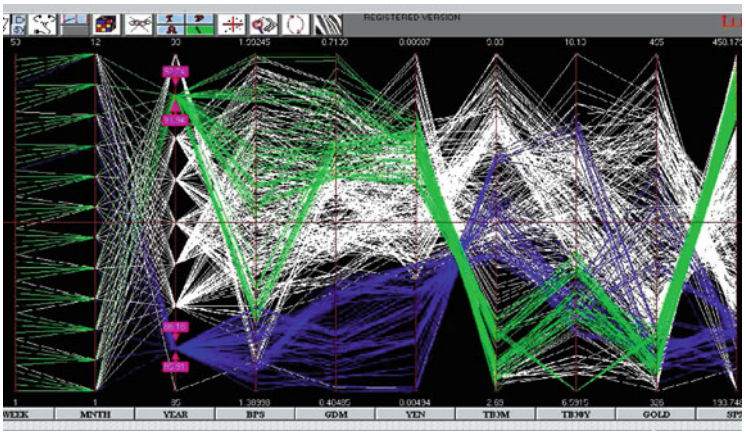


Fig. 11.9 Financial data.
Quotes by **W**eek-on Mondays, **M**onth, **Y**ear – the first 3 axes fix the date; **S**terling, **D**mark, **Yen** rates per $ 4th, 5th, 6th axes; **3M**TB, **30Y**TB interest rates in %, 7th, 8th axes; **G**old in $/ounce, 9th, **SP**500 index values on 10th axes

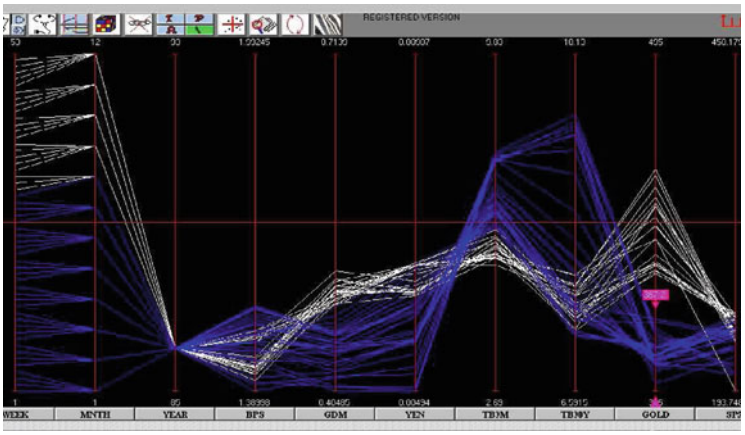


Fig. 11.10 Gold prices In 1986.
Gold prices jumped in the 2nd week of August. Note the correlation between the low **Yen**, high **3M**TB rates and low **G**old price range

and lowering the range causes the other variable's range to rise and is a nice way to show negative correlation *interactively*.

For the contrarians among us, we check also for positive correlation Fig. 11.11 (right). We find that it exists when **G**old prices are low to mid-range as happened for a period in the 1990s. This is a free investment tip for bucking the main trend shown in Fig. 11.11 (left). It is also a nice opportunity for showing the *inversion* feature activated by the icon with 2 cyclical arrows. A variable is selected and the

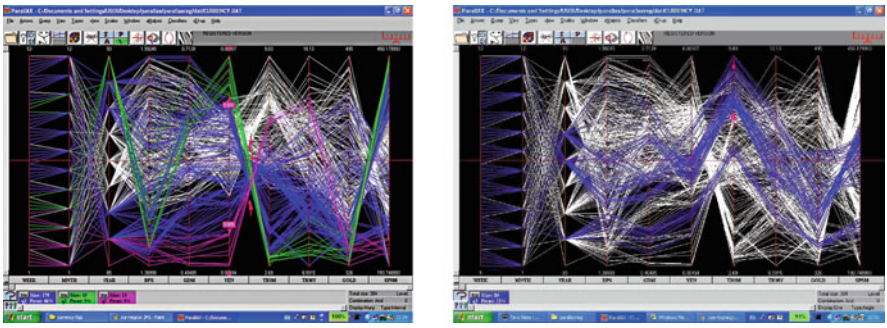


Fig. 11.11 (*Left*) Negative correlation. (*Right*) Positive correlation.
(*Left*)The crossing lines between the 6th and 7th axes in Fig. 11.9 show strong negative correlation between **Yen** and **3M**TB rates. One cluster is selected with the **P**inch query and combined with the high and low ranges on the **Yen** axis.(Right) A positively correlated cluster where the **Yen** and **3M**TB rates move *together when* **G***old prices are low to mid-range*

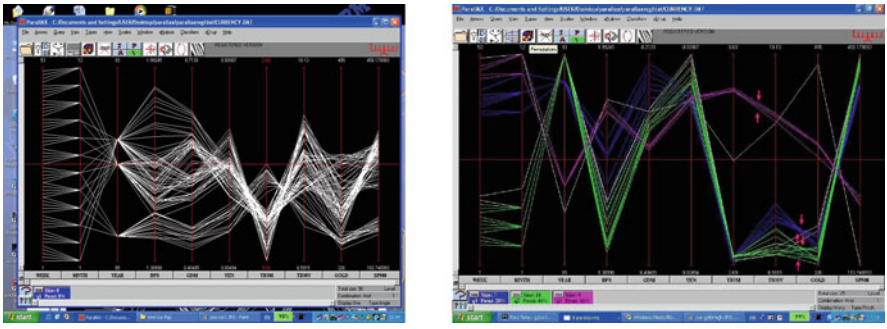


Fig. 11.12 (*Left*)Inverting the **3M**TB axis. (*Right*) Variations in exchange rates.
Now the lines between the **Y**en-**3M**TB and **3M**TB-**30M**TB axes in Fig. 11.11 (*right*) cross. Variations in the rate of exchange of the currencies correlate with movements in the price of **G**old

min/max values on that axes are inverted. Diverging lines (as for + correlation) now intersect Fig. 11.12 (left) making it easier visually to spot the crossing and hence the correlation. Actually, the recommendation is to work with the **A** query experimenting with various angle ranges using the inversion to check out or confirm special clusters.

When stuck don't just stand there but vary one of the variables watching for interesting variations in the other variables. Doing this on the **Yen** axis, Fig. 11.12(left) we strike another gold connection. The (rough) intersection of a bunch of lines joining **Yen** to the **D**mark corresponds, by the duality, to their rate of exchange. When the rate of exchange changes so does the intersection and the price of Gold! That is movements in currency exchange rates and the price range of **G**old go together. Are there any indications that are associated with the high range of **G**old? The top price range is selected, Fig. 11.13 (left), and prompted by the result

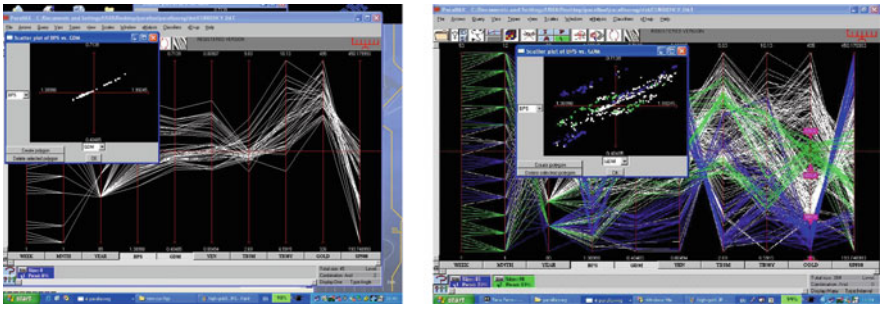


Fig. 11.13 (*Left*) High **G**old. (*Right*) Two price ranges of **G**old.
(*Left*) Note the perfect straight line in the **S**terling vs. **D**mark plot. The slope is the *rate of exchange* between them and which remains constant when **G**old prices peak. (*Right*) The associated **S**terling vs. **D**mark plots show no regularity

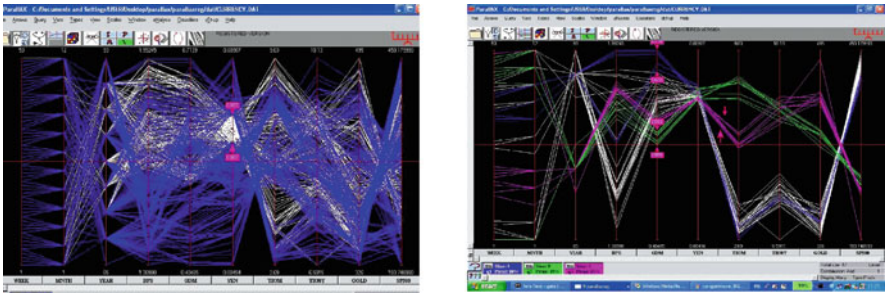


Fig. 11.14 (*Left*)The complement of an **I** query. (*Right*) **Yen** stable.
(*Right*) For the **Yen** trading in a narrow range, high **D**mark goes with low **3M**TB rates, low **D**mark goes with high **3M**TB rates, while mid **3M**TB rates go with high **G**old

of the previous query we check out the exchange rate between **S**terling and **D**mark (or **Yen**) and the resul is stunning: a perfect straight line. The slope *is* the rate of exchange which is constant when **G**old tops out. The relation between **S**terling and **D**mark is checked for different price ranges of **G**old, Fig. 11.13 (right), and the only regularity found is the one straight-line above. Aside from the trading guideline it establishes, it suggests "behind-the-scenes manipulation of the **G**old market" … we could have said that but we won't. We perish this thought and proceed with the boolean complement, Fig. 11.14(left) of an **I** (or any other) query. Not finding anything we select a narrow but dense range on the **Yen**, Fig. 11.14 (right) and notice an interesting relation between **D**mark, interest rates and **G**old.

There is an exploratory step akin to "multidimensional contouring" which we fondly call **Zebra** activated by the last icon button on the right with the appropriate skin-color. A variable axis is selected, the **SP**500 axis in Fig. 11.15 (left), and divided into a number (user specified) intervals (here it is 4) and colored differently. This shows the connections (influence) of the intervals with the remaining variables

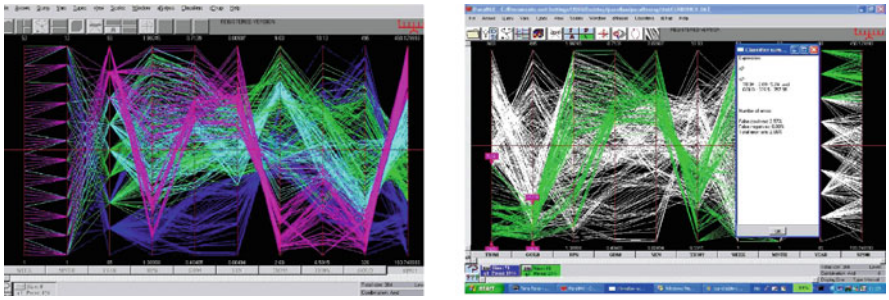


Fig. 11.15 (*Left*) The *zebra* query. (*Right*) The rule for high **SP**500.
(*Left*)It partitions and colors the segments of values differently. A variable, here the **SP**500 axis, is divided it into equal (here 4) intervals. This quickly reveals interelationships. Note especially those for the highest **SP**500 range. (*Right*) Both **3M**TB (the "short-bond" as it is called) and **G**old are low and in this order of importance

which here is richly structured especially for the highest range. So what does it take for the **SP**500 to rise? This is a good question and helps introduce Parallax's classifier. The result, shown in Fig. 11.15 (right) confirms the investment community's experience that low **3M**TB and **G**old correlate with high **SP**500. A comparison with the results obtained on this dataset with other visualization tools would be instructive though unfortunately not available. Still let us consider such an analysis done by the scatterplot matrix. There are 10 variables (axes) which requires 45 pairwise scatterplots. Let us assume that each is no larger than 5×5 cm square and a large screen monitor is available. Varying 1, 2 or more variables in tandem and observing the effects ***simultaneously*** over **all** the variables in the 45 squares may be possible but quite challenging. By contrast, the effects of varying **D**mark, *conditionally* for stable **Y**en, are easily seen on the two interest rates, **G**old as well as the remaining variables in *one* Fig. 11.14 (right). This example illustrates the difficulties due to high representational complexity which is $O(N^2)$ for the scatterplot matrix but $O(N)$ for ‖-coords.

11.3 Classification

Though it is fun to do this data exploration, the level of skill and patience required tends to discourage some users. So there have been persistent requests and admonitions for tools which at least partially automate the knowledge discovery. Here the **Nested Cavities**[4] (abbr. **NC**) classifier Inselberg and Avidan (2000) is revisited and substantially improved. For a dataset P and a subset S the goal is to construct a rule distinguishing the elements of S from those in $P - S$. **NC** is a geometrical algorithm which builds a sequence of nested unbounded parallelopipeds

[4]My dentist really liked this name!

of minimal dimensionality containing subsets of P, from which a hypersurface (the rule) containing the subset S emerges. The partitioning of $P - S$ and S into disjoint subsets is very useful when the original rule obtained is either too complex or imprecise. By applying **NC** to the partitions of S a simpler and more precise classification may be obtained. This process is illustrated on a (sonar) dataset with 60 variables and two categories ("mines" and "rocks") resulting in significant improvements of the original rule. Such a situation is generic and occurs with other datasets as illustrated with a similar decompositions of a financial dataset producing two sets of conditions determing gold prices. Yet another example is a dataset pertaining to ovarian cancer which is decomposed by **NC** to distinct cases of this cancer. We propose including such partitioning for the classification of datasets. The process may be automated and also allows the classification of the sub-categories to be done in parallel.

The classifier's output may also be that there is insufficient information to obtain the desired distinction. What can be done when the classifier either fails to converge or the rule it yielded is very complex or not accurate? It turns out that the classifier reveals the dataset's structure pointing out how it can partitioned into sub-categories which can be more simply and accurately classified.

To understand the key idea and make this section reasonably self-sufficient an overview of the classification algorithm is presented. It is illustrated on a dataset with 32 variables and 2 categories obtaining an accurate rule using the classifier as originally proposed. The motivation for the extension is described next with a dataset having 60 variables and two categories. Though the resulting rule is not accurate the dataset's structure is revealed yielding a partition which substantially improves the classification. The presentation is intuitive and technical details for the implementation are not elaborated.

11.3.1 Classification Algorithm

*W*ith parallel coordinates Inselberg (2009) a dataset P with N variables is transformed into a set of points in N-dimensional space. In this setting, the designated subset S can be described by means of a hypersurface which encloses just the points of S. In practical situations the strict enclosure requirement is dropped and some points of S may be omitted ("false negatives"), while some points of $P - S$ are allowed ("false positives") in the hypersurface. The description of such a hypersurface provides a rule for identifying, within an acceptable error, the elements of S. It turns out that using Parallel Coordinates not only enables the efficient construction of the hypersurface but also the **visualization of the rule**.

At first the algorithm determines a tight upper bound for the *dimension R of S*. For example, P may be a 3-dimensional set of points but all point of S may be on a plane; in which case S has dimension 2. Once R is determined R variables out of the N are chosen according to their predictive value and the construction process, schematically shown in Fig. 11.16, operates only on these R selected variables. It is accomplished by:

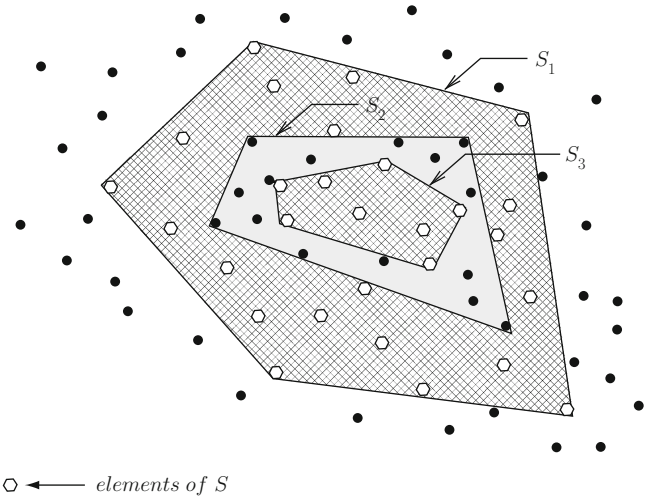


Fig. 11.16 Construction of enclosure for the **Nested Cavities** algorithm. The first "wrapping" S_1 is the convex hull of the points of S which also includes some points of $P - S$. The second wrapping S_2 is the convex hull of these points and it includes some points of S which are enclosed with the third wrapping S_3. To simplify the wrappings are shown as convex hulls rather than as approximations. Here the selected set is $S = (S_1 - S_2) \cup (S_3 - S_4)$ where $S_4 = \emptyset$

- Use of a "wrapping" algorithm to enclose the points of S in a hypersurface S_1 containing S and typically also some points of $P - S$; so $S \subset S_1$.[5]
- The points in $(P - S) \cap S_1$ are isolated and the wrapping algorithm is applied to enclose them, and usually also some points of S_1, producing a new hypersurface S_2 with $S \supset (S_1 - S_2)$,
- The points in S not included in $S_1 - S_2$ are next marked for input to the wrapping algorithm, a new hypersurface S_3 is produced containing these points as well as some other points in $P - (S_1 - S_2)$ resulting in $S \subset (S_1 - S_2) \cup S_3$,
- The process is repeated alternatively producing upper and lower containment bounds for S; termination occurs when an error criterion is satisfied or when convergence is not achieved.

The algorithm decomposes P into nested subsets, hence the name **Nested Cavities** (abbr. **NC**) for the classifier. The nested subsets are disjoint so they are *partitions* of P. Let us illustrate all this with an example on a real dataset with 2 categories and 32 variables $x_1, x_2, \ldots, x_{32}$. The rule found by the **NC** classifier is shown in Fig. 11.18 and Fig. 11.19.

[5]By $S_j \subset S_k$ it is meant that the set of points enclosed in the hypersurface S_j is contained in the set of points enclosed by the hypersurface S_k.

Basically, the "wrapping" algorithm produces a convex-hull approximation; the technical details are not needed here. The efficiency of the version implemented here is due to the use of the ‖-cs representations of N-dimensional objects Inselberg (2009) applied in the description of the resulting hypersurface. It can and does happen that the process does not converge when P does not contain sufficient information to characterize S. It may also happen that S is so "porous" (i.e. sponge-like) that an inordinate number of iterations are required.

At step r the output is the description of the set S_r which consists of:

- A list of the minimum number of variables needed to describe S *without loss of information*. Unlike other methods, like the Principal Component Analysis (PCA), the classifier discards only the redundant variables. It is important to clarify this point. A subset S of a multidimensional set P is not necessarily of the same dimensionality as P. So the classifier finds the dimensionality of S in terms of the original variables and retains only those needed for describing S. That is, it finds the *basis* in the mathematical sense of the smallest subspace containing S, or more precisely the current approximation for it. This basis is the minimal set M_r of variables needed to describe S; alternatively the *"features"* of S. We call this process dimensionality **selection** to distinguish it from dimensionality *reduction* which is usually done *with* loss of information. Retaining the original variables is important in the applications where the domain experts have developed intuition about the variables they measure. The classifier presents M_r *ordered according to a criterion which optimizes the clarity of separation*. In addition the classifier's output describes:
- The current approximation of the rule stated in terms of conditions on the variables M_r, which constitutes the description of the current hypersurface.

So on convergence, say at step $2n$, the description of S provided is :

$$S \approx (S_1 - S_2) \cup (S_3 - S_4) \cup \ldots \cup (S_{2n-1} - S_{2n}),$$

this being the terminating expression resulting from the algorithm. The implementation allows the user to select a subset of the available variables and restrict the rule generation to these variables. In certain applications, such as process control, not all variables can be controlled and hence it would be useful to have a rule involving such variables that are "accessible" in a meaningful way.

The results (precision of rule) obtained by the **NC** classifier applied to benchmark datasets were the most accurate when compared to those obtained by 22 other well-known classifiers (see Inselberg and Avidan 2000).

11.3.2 Tracing some Steps

A dataset with 32 variables $x_1, x_2, \ldots, x_{32}$ having 2 categories each having 300 points in Fig. 11.17 is chosen to exemplify the process. The **NC** classifier

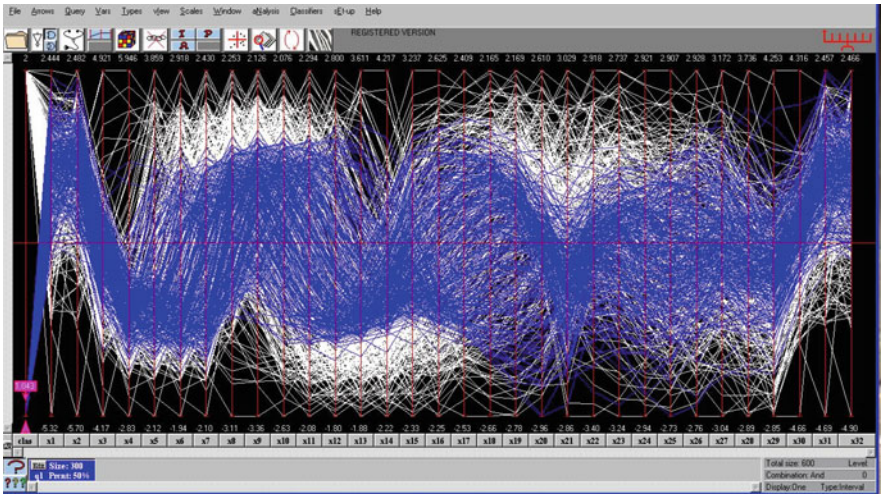


Fig. 11.17 Two categories colored differently each have 300 data points. Category 1 is selected by the query indicated with the downward and upward arrowheads at the bottom of the first axis

applied to category 1 found that only the variables $x_{11}, x_{14}, x_8, x_{10}, x_{12}, x_9, x_7, x_{23}, x_{13}$ are needed to specify the classification rule in only one iteration and about 6% error. The order of the variables is significant and is discussed shortly. The second iteration involves additionally x_2, x_5, x_6 reducing the error to 4%. The result is shown in Fig. 11.18; the separation achieved is striking. See also further cross-sections in Fig. 11.19 which reveal two tight-fiting "pretsels" winding in 9-dimensions. As recommended by H. Hinterberger Schmid and Hinterberger (1994) more than one display, here parallel coordinate and scaterplots, are used for supporting comparisons, confirming and clarifying the results.

For the example it suffices to use unbounded parallelopipeds for the wrapping. Let I_j be the range of the variable x_j within the set of points C in S (here category 1); that is from the minimum to the maximum values of x_j. Further, let C_j be the number of data points in the **global** set P with range I_j. Specifically, as shown in Fig. 11.20, C_{11} has 484 points and, Fig. 11.21, C_{14} has 510 points. So the algorithms first step is to find the C_j and put them in *ascending order* i.e. here $C_{11} < C_{14}$ and in general $C_{j_1} < C_{j_2} \ldots < C_{j_m}$ where m is the number of variables (here $m = 32$). Next starting with j_1 the range of each variable is restricted in the order $j_1, j_2, \ldots$ observing the number of points C_{j_k} in the P with these restrictions. At some stage either $C_{j_k} = C_{j_{k+1}}$ for $k < m$ and procecure stops or C_{j_m} is reached and then terminates. The k at termination is the *dimensionality* of S. Here, Fig. 11.22, $C_{11} \cap C_{14} = 400$. This is what determines the order of the 9 variables chosen here by **NC** i.e. $j_1 = 11, j_2 = 14, \ldots, j_8 = 23, j_9 = 13$. In turn, restricting the ranges of variables on P in this order sequentially reduces *most rapidly* the number of points approximating S with one iteration. That is,

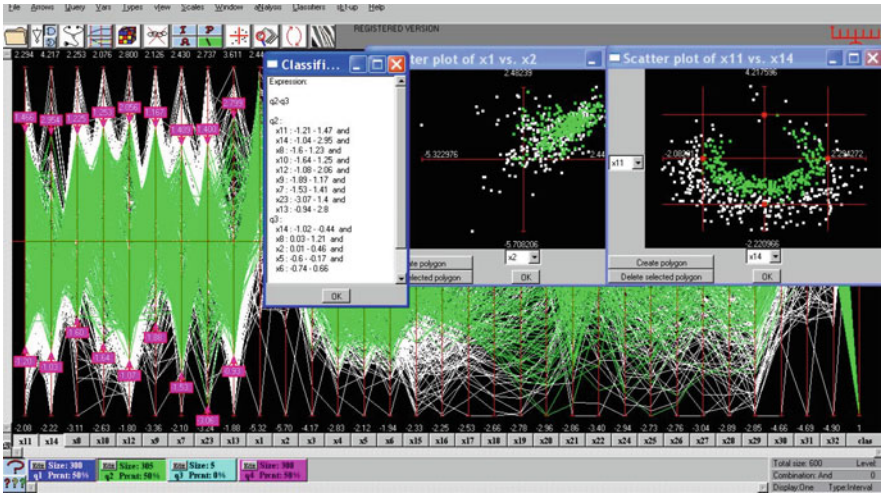


Fig. 11.18 The dataset with 32 variables is shown in the background. It has 2 categories whose points are differencly colored. The table contains the explicit rule. The left scatterplot shows the first two consecutive variables. The classifier found that only 9 variables, whose ranges are indicated by the downward and upward arrowheads on their axis, are needed to describe the rule with a precision of 4%. The plot of the right shows the two best predictors and the separation achieved between the two categories

$$C_{11} = 484 > 400 > 370 > 350 > \ldots 308 > 305,$$

terminates with $j_9 = 13$. Restricting the range of **any** other variable x_t to I_t *does not reduce* the number of points further in the resulting set which *at all times must contain the selected subset* S. Precisely for this reason we consider the dimension of S in this case to be 9 as it is completely contained withing the unbounded parallelopiped determined by the restricted ranges of the **9** aforementioned variables. This is also the reason for considering x_{11} as the *best* predictor, x_{14} as the *second* best predictor and so on. This is a measure of the variables' *relative importance* and has considerable practical significance when there are *missing values* whose influence diminishes in the same order.

With one more iteration the number of points selected by the rule is exactly those contained in S (i.e. 300). For this iteration another parallelopiped is constructed *within* the one obtained in the first iteration and *deleted* creating a *cavity*. In this way the process carves out unwanted parts and provides the separation shown. Parallel coordinates are used internally in the implementation taking advantage of the efficient intersection and containment algorithms Inselberg (2009). The overall computational complexity is $O(N^2|P|)$ where N is the number of variables and $|P|$ is the number of points in P.

Two error estimates are used: Train & Test and Cross-correlation. When the rule involves several iterations an additional criterion is employed to avoid *overfiting*.

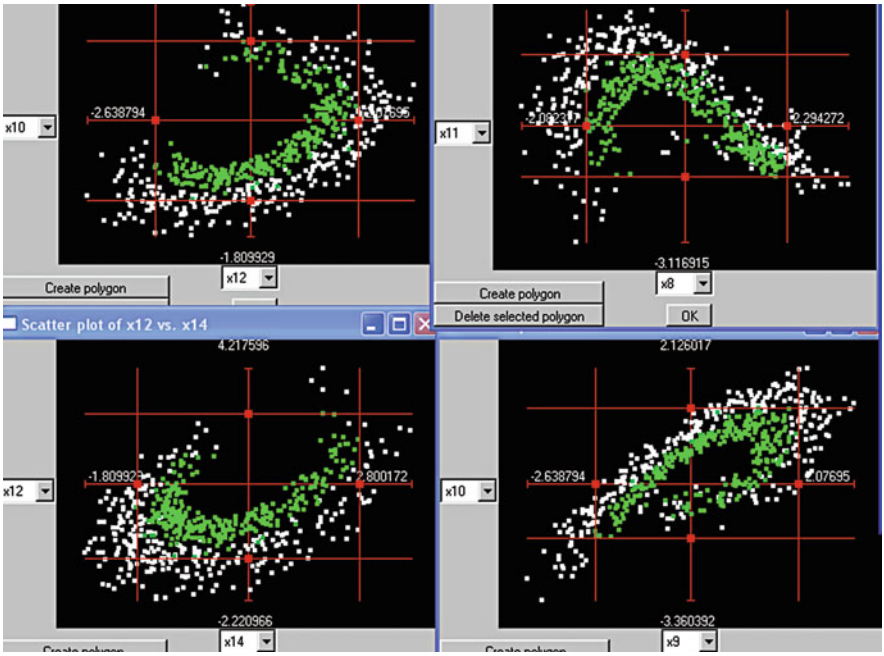


Fig. 11.19 Various cross-sections of the hypersurface corresponding to the classification rule for category 1 above

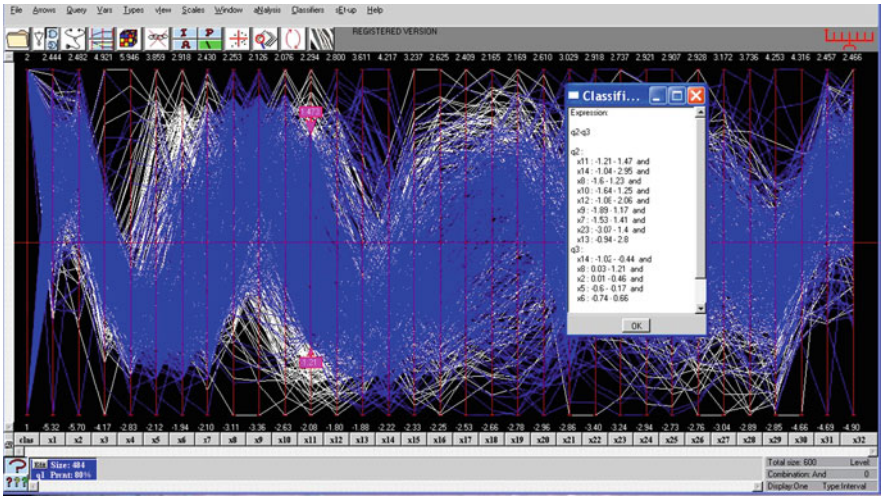


Fig. 11.20 Restricting the range of the first variable x_{11} chosen by the classifier eliminates 116 of the points in category 2

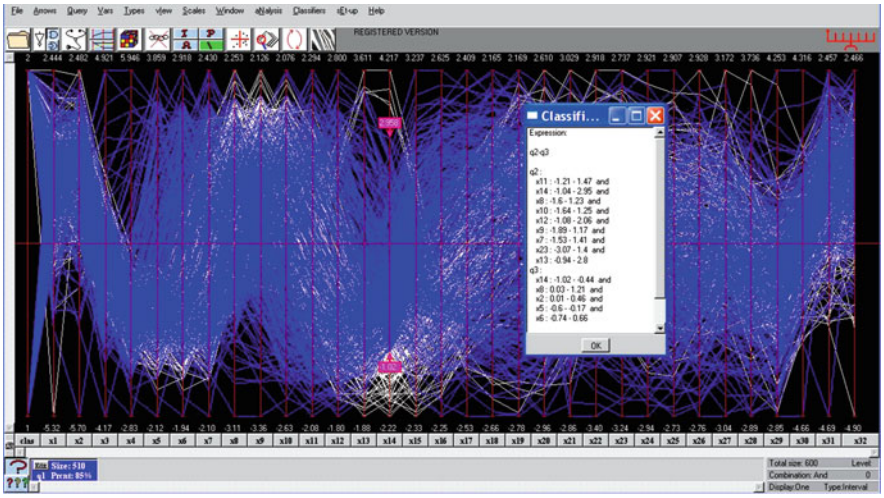


Fig. 11.21 Restricting the range of the second variable x_{14} eliminates only 90 of the points in category 2 showing why x_{11} (above) is a better predictor for category 1 than x_{14}

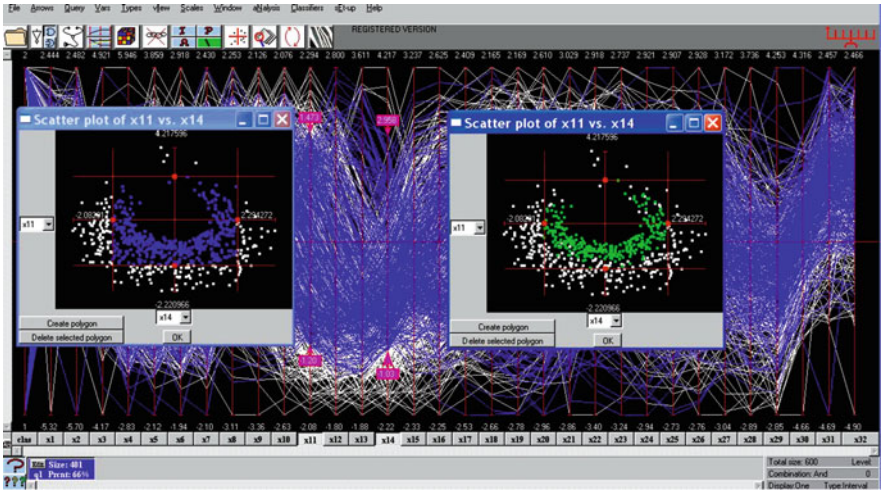


Fig. 11.22 Restricting both x_{14} and x_{11} eliminates 200 points of category 2. The remaining 100 points of category 2 are eliminated by applying the subsequent conditions specified by the rule

Namely, the rule error is traced iteration by iterations and the process is stopped when the error *increases* compares to the previous. As pointed out in Inselberg and Avidan (2000), the rule obtained by the **NC** classifier were applied to 4 bench-mark datasets and were the most accurate compared to those obtained by 22 other well known classifiers.

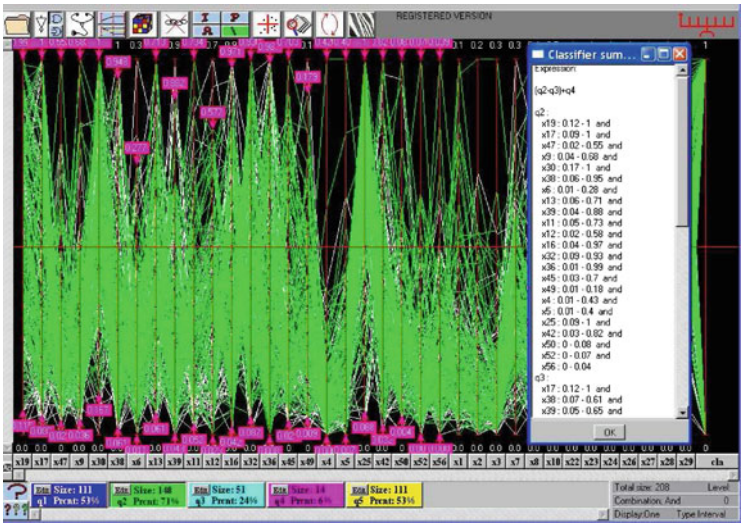


Fig. 11.23 Sonar dataset with 60 variables and 2 categories. The **NC** classifier partitions the dataset into 3 nested subsets indicated by the 3 rectangles, in middle of the lower row, with 148, 51 and 14 items each. To improve the visual clarity some of the variables (axes) not needed in the rule were removed

Due to the short exposition in Inselberg and Avidan (2000) questions were raised by a number number of users on detailed aspects of the **NC** algorithm. This detailed explanation is intended to demistify some of the nuances, present the updates introduced in the meantime and provide foundational understanding for the new idea introduced next.

11.3.3 Partioning into Sub-categories

As one might expect things do not always work out as nicely as for the example. The sonar dataset from UCI (0000) has been a real classification challenge. It has 60 variables, 208 observations and 2 categories 1 for *Mines* with 111 observations and 0 for *Rocks* with 97 data points. Applying the **NC** classifier partitions the dataset into 3 nested subsets S_1, S_2, S_3, with 148, 51 and 14 items respectively, The rule obtained involves about 35 variables and an unacceptable high error of about 45%. The result, demarcating the nesting (by the rectangles in the lower row) and showing some of the variables used in the rule is shown in Fig. 11.23.

The schematic in Fig. 11.24 clarifies the partition of the dataset into 4 disjoint sets, $M_1 =, M_2$ for the mines and R_1, R_2 for the "rocks". These are obtained by $S_3 = M_2$, $R_2 = S_2 - S_3$, $M_1 = S_1 - S_2$ and $R_1 = All - S_1$ where All stands

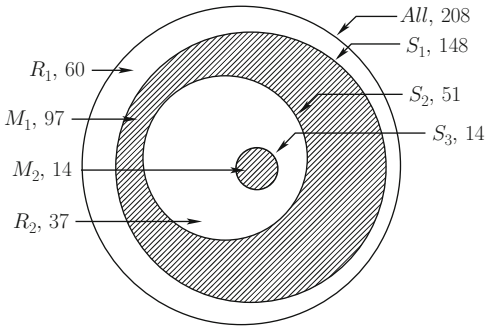


Fig. 11.24 Schematic of the sonar dataset partition. The S_i are the nested subsets, $R = R_1 \cup R_2$ and the mines $M = M_1 \cup M_2$. Together with the notation is the number of items contained in each subset

for the full dataset. This is a very useful insight into the structure of the dataset and motivates the idea. The bulk of the mines are in M_1 which has the higher values of the variables needed to specify the rule. By contrast, the subset $M_2 = S_3$ is a small "island", having the smaller variable values, surrounded by R_2 differs markedly from M_1. Why not split M into two classes as suggested by the picture, finding the rules separately and use them if they are more precise than the one found at first, and it works!

Consider $R \cup M_1$ and apply the **NC** classifier. A rule distinguishing M_1 from R is found needing only 4 variables. Due to small size of M_1 the error estimates, with either *cross-correlation* or *train-and-test* the number of "false-negatives" were high, about 30%, though the "false-positives" were about 5% yielding a weighted average error of about 15%. For another interesting comparison distinguishing M_1 from M, **NC** yields a rule with 5 variables and an 8% average error. It is clear that M_1 is easily distinguished both from the "rocks" and the larger class of mines M_1. This strongly suggests that there are two very different types of mines included in this dataset. To summarize part of **NC**'s output, indicated by the rectangles in the lower row, gives the decomposition of the dataset into nested subsets. From these one or more of the categories can be partitioned to obtain a more accurate and simpler rule. While this has been observed for some time it was only investigated recently. Of course, the idea of partitioning is inherent in classification which after all pertains to the division of a dataset and differentiating between the parts. While there is a lot of literature on partitions in *data mining* i.e. Han and Kamber (2001) and Agarwal et al. (1999) this specific method has apparently not been suggested. Such a decomposition can clearly be automated and also the classification of the new categories can be *done in parallel*.

We have encountered similar situations with other datasets. From the 1986-year subset of the financial dataset in the previous section, classification with **NC** showed that there are two different sets of conditions which cause the price of gold to rise. These are better characterized separately as for the sonar dataset. Interestingly, the price of Yen is involved in one of the conditions but not the other. Another such example is shown in Fig. 11.25 for a dataset with measurements on ovarian cancer having 50 variables and 3 of categories (types of cancer). Classification

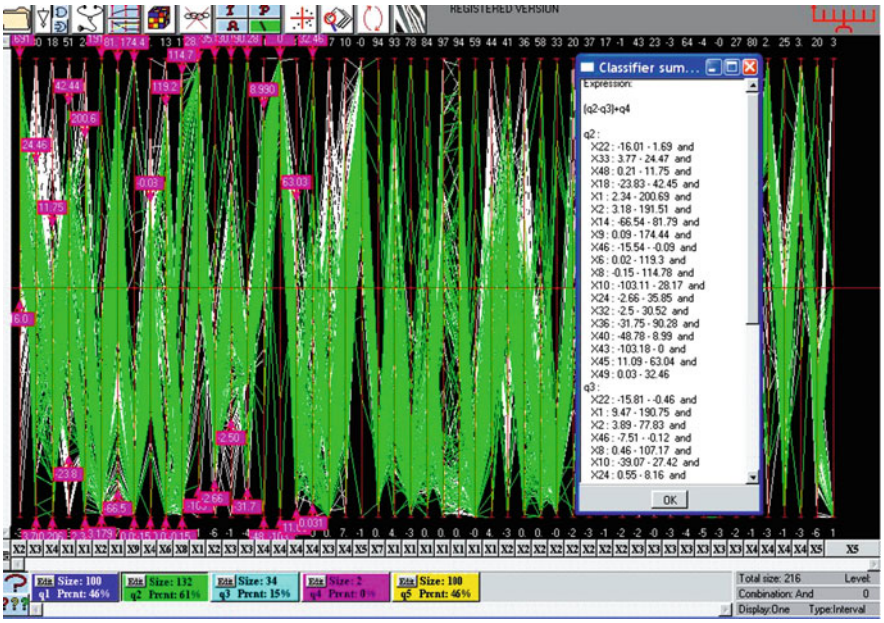


Fig. 11.25 This a dataset with measurements pertaining to ovarian cancer having 50 variables and 3 categories. Classification by **NC** of one category yields a complex and inaccurate rule. It also partitions it into 2 sub-categories yielding simpler and more precise rule. This may suggest that this type of cancer has two different descriptions (morphologies)

of one category yielded a complex and imprecise rule. However, it also showed a decomposition into two sub-classes for which good rules were obtained. Since different descriptors were involved for each sub-class the thought arises that the cancer types are really different. These examples are *generic* of a common problem in classification, and for these we offer a time-honored solution: **divide and conquer**.

11.4 Visual & Computational Models

*F*inally we illustrate the methodology's ability to model multivariate relations in terms of hypersurfaces – just as we model a relation between two variables as a region in a 2-D plane. Then by using an interior point algorithm, as shown in Fig. 11.32 of the next section, with the model we can do trade-off analyses, discover sensitivities, understand the impact of constraints, and in some cases do optimization. For this purpose we shall use a dataset consisting of the outputs of various economic sectors and other expenditures of a particular (and real) country. It consists of the monetary values over several years for the **Ag**ricultural, **Fish**ing,

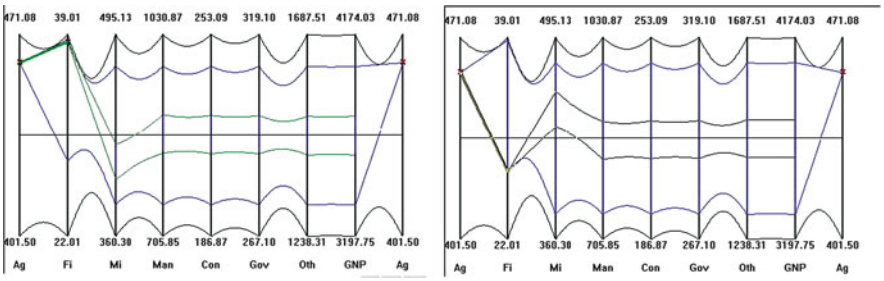


Fig. 11.26 (*Left*) Model of a country's economy. (*Right*) Competition for labor between the **Fish**ing & **Min**ing sectors.
Choosing high **Ag**ricultural and high **Fish**ing output **forces** low **Min**ing output

and **Min**ing sector outputs, **Man**ufacturing and **Con**struction industries, together with **Gov**ernment, Miscellaneous spending and resulting GNP; eight variables altogether. We will not take up the full ramifications of constructing a model from data. Rather, we want to illustrate how ∥-coords may be used as a modeling tool. Using the Least Squares technique we "fit" a function to this dataset and are not concerned at this stage whether the choice is "good" or not. The function obtained bounds a region in $\mathbb{R}^8$ and is represented by the upper and lower curves shown in Fig. 11.26.

The picture is in effect a simple visual model of the country's economy, incorporating its capabilities, limitations and interelationships among the sectors. A point interior to the region, satisfies all the constraints simultaneously, and therefore represents (i.e. the 8-tuple of values) a feasible economic policy for that country. Using the interior point algorithm we can construct such points. It can be done *interactively* by sequentially choosing values of the variables and we see the result of one such choice in Fig. 11.26(left). Once a value of the first variable is chosen (in this case the **Agr**icultural output) within its range, the dimensionality of the region is reduced by one. In fact, the upper and lower curves between the 2nd and 3rd axes correspond to the resulting 7-dimensional hypersurface and show the available range of the second variable **Fish**ing reduced by the constraint. This can be seen (but not shown here) for the rest of the variables. That is, due to the relationship between the 8 variables, a constraint on one of them impacts all the remaining ones and restricts their range. The display allows us to experiment and actually see the impact of such decisions downstream. By interactively varying the chosen value for the first variable we found, that it not possible to have a policy that favors **Agr**iculture without also favoring **Fish**ing and vice versa.

Proceeding, a very high value from the available range of **Fish**ing is chosen and it corresponds to very low values of the **Min**ing sector. By contrast in Fig. 11.26(right) we see that a low value in **Fish**ing yields high values for the **Min**ing sector. This inverse correlation was examined and it was found that the country in question has a large number of migrating semi-skilled workers. When the fishing industry is doing well most of them are attracted to it leaving few available to work in the mines and

vice versa. The comparison between the two figures shows the competition for the same resource between **Min**ing and **Fish**ing. It is especially instructive to discover this *interactively*. The construction of the interior point proceeds in the same way. In the next section in the discussion on surfaces this construction is shown for higher dimensional hypersurfaces.

11.5 Parallel Coordinates: Overview of the Fundamentals

This section is for readers interested in the foundational understanding of the methodology. The overview covers recent results and future prospects. For deeper excursions to ‖-cs readers are referred to the textbook Inselberg (2009).

11.5.1 Lines

An N-dimensional line ℓ can be described by the $(N-1)$ linear equations:

$$\ell \quad : \quad \begin{cases} \ell_{1,2} & : \; x_2 = m_2 x_1 + b_2 \\ \ell_{2,3} & : \; x_3 = m_3 x_2 + b_3 \\ & \cdots \\ \ell_{i-1,i} & : \; x_i = m_i x_{i-1} + b_i \\ & \cdots \\ \ell_{N-1,N} & : \; x_N = m_N x_{N-1} + b_N \end{cases} , \tag{11.2}$$

each with a pair of adjacently indexed variables. In the $x_{i-1}x_i$-plane the relation labeled $\ell_{i-1,i}$, $N = 2, \ldots, N$ is a line, and by the *line* $\leftrightarrow$ *point* duality, (11.1), it can be represented by the point

$$\bar{\ell}_{i-1,i} = \left(\frac{1}{(1-m_i)} + (i-2) \; , \; \frac{b_i}{(1-m_i)}\right) \tag{11.3}$$

Here the inter-axes distance is 1 so that $i-2$ is distance between the y (or $\bar{X}_1$) and $\bar{X}_{i-1}$ axes. Actually any $N-1$ independent equations like

$$\ell_{i,j} : x_i = m_{i,j} x_j + b_{i,j} \; , \tag{11.4}$$

can equivalently specify the line ℓ, for (11.4) is the projection of ℓ on the $x_i x_j$ 2-D plane and $N-1$ such independent projections completely describe ℓ. There is a beautiful and very important relationship illustrated in (left) Fig. 11.27.

For a line ℓ in 3-D the three points $\bar{\ell}_{12}, \bar{\ell}_{13}, \bar{\ell}_{23}$ are collinear; the line so determined is denoted by $\bar{L}$. It is easy to see that a polygonal line on all the $N-1$

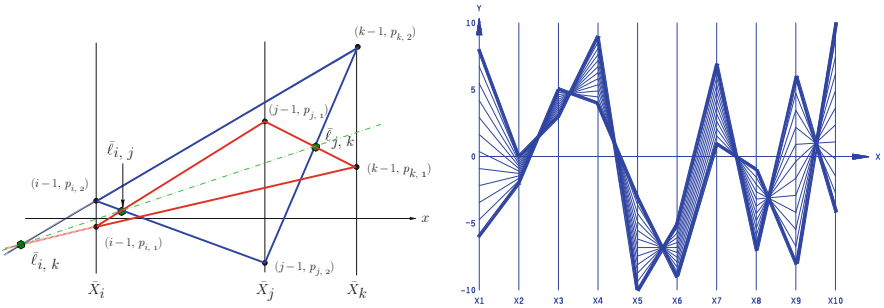


Fig. 11.27 Properties of multidimensional lines.
(*Left*)The 3 points $\bar{\ell}_{i,j}, \bar{\ell}_{j,k}, \bar{\ell}_{i,k}$ are collinear for $i \neq j \neq k$. (Right) A line interval in 10-D

points, given by (11.3) or their equivalent, represents a point on the line ℓ. Two points in $\mathbb{R}^N$ determine a line ℓ, so starting with the two polygonal lines the $N-1$ intersections of their $\bar{X}_{i-1}, \bar{X}_i$ portions are the $\bar{\ell}_{i-1,i}$ representing points for ℓ. A line interval in 10-D and several of its points is seen on the (right) Fig. 11.27. By the way, the indexing of the points $\bar{\ell}$, usually not shown to conserve display space, is essential and must be accessible when needed.

11.5.2 Planes & Hyperplanes

*W*hile a line can be determined from its projections, a plane even in 3-D can not. A new approach is called for Eickemeyer (1992). Rather than discerning a p-dimensional object from its points, it is described in terms of its (p-1)-dimensional subsets constructed from the points. Let's see how this works. In Fig. 11.28 (left) polygonal lines representing a set of coplanar points in 3-D are seen. From this picture even the most persistent pattern-seeker can **not** detect any clues hinting at a relation among the three variables much less a linear one. The plane has dimension $p = 1$ so we look at *lines* (having dimension $p - 1 = 1$) on the plane constructed so that each pair of polygonal lines the lines $\bar{L}$ of the 3 point collinearity shown in Fig. 11.27 (left) are obtained. The result, shown on the right, is stunning. All the $\bar{L}$ lines intersect at a point which turns out to be characteristic of coplanarity but not enough to specify the plane. Translating the first axis $\bar{X}_1$ to the position $\bar{X}_{1'}$, one unit to the right of the $\bar{X}_3$ axis and repeating the construction, based on the axes triple $\bar{X}_2, \bar{X}_3, \bar{X}_{1'}$, yields a second point shown in Fig. 11.29(left). For a plane described by:

$$\pi : c_1 x_1 + c_2 x_2 + c_3 x_3 = c_0 \,, \tag{11.5}$$

the Cartesian coordinates of the two points, in the order they are constructed, are respectively

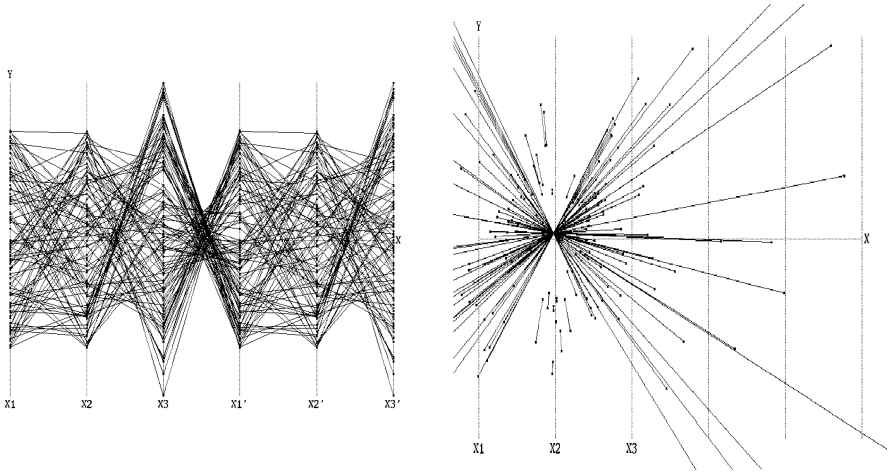


Fig. 11.28 Coplanarity>.
(*Left*)The polygonal lines on the first 3 axes represent a set of coplanar points in 3-D.(Right) Coplanarity! Forming lines on the plane, with the 3-point-collinearity property Fig. 11.27(left), the resulting lines intersect at poit

$$\bar{\pi}_{123} = \left(\frac{c_2 + 2c_3}{S} \ , \quad \frac{c_0}{S} \right) \ , \ \bar{\pi}_{1'23} = \left(\frac{3c_1 + c_2 + 2c_3}{S}, \frac{c_0}{S} \right) \ , \qquad (11.6)$$

for $S = c_1 + c_2 + c_3$. Three subscripts correspond to the 3 variables appearing in the plane's equation and the axes triple used for their construction, and distinguish them from the points with two subscripts representing lines. The 2nd and 3rd axes can also be consecutively translated, as indicated in Fig. 11.28(left), repeating the construction to generate two more points denoted by $\bar{\pi}_{1'2'3}, \bar{\pi}_{1'2'3'}$. These points can also be found otherwise in an easier way. The gist of all this is shown in Fig. 11.29(right). The distance between successive points is $3c_i$. The equation of the plane π can actually be read from the picture!

In general, a hyperlane in N-dimensions is represented uniquely by $(N - 1)$ points each with N indices. There is an algorithm which constructs these points *recursively*, raising the dimensionality by one at each step, as is done here starting from points (0-dimensional) and constructing lines (1-dimensional). By the way, all the nice higher dimensional projective dualities like *point* $\leftrightarrow$ *hyperplane*, *rotation* $\leftrightarrow$ *translation* etc hold. Further, a multidimensional object, represented in $\|$-cs, can still be recognized after it has been acted on by projective transformation (i.e. translation, rotation, scaling and perspective). The recursive construction and its properties are at the heart of the $\|$-cs visualization.

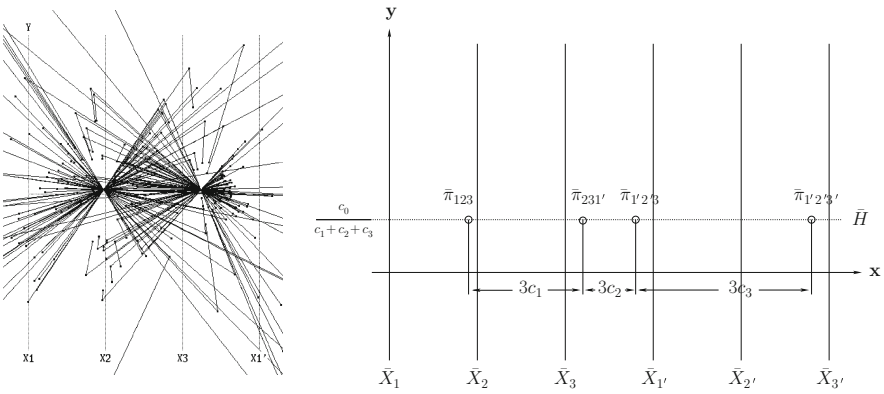


Fig. 11.29 Plane representation.
(*Left*)The two points where the lines intersect uniquely determine a plane π in 3-D. (*Right*) From four points, constructed similarty by consecutive axes translation, the coefficients of $\pi : c_1x_1 + c_2x_2 + c_3x_3 = c_0$ can be read from the picture!

Challenge: Visualizing Families of Proximate Planes

Returning to 3-D, it turns out that for points as in Fig. 11.28 which are "nearly" coplanar (i.e. have small errors) the construction produces a pattern very similar to that in Fig. 11.29(left). A little experiment is in order. Let us consider a family of *proximate* (i.e. close) planes generated by

$$\Pi = \{\pi : c_1x_1 + c_2x_2 + c_3x_3 = c_0, \quad c_i \in [c_i^-, c_i^+], \quad i = 0, 1, 2, 3\}\,, \qquad (11.7)$$

randomly chosing values of the c_i within the allowed intervals to determine several planes $\pi \in \Pi$, keeping at first $c_0 = 1$, and plotting the two points $\bar{\pi}_{123}, \bar{\pi}_{1'23}$ as shown in Fig. 11.30 (left). Not only is closeness apparent but more significantly the distribution of the points is not chaotic. The outline of two hexagonal patterns can be discerned. The family of "close" planes is visualizable but also the variations in several directions. These polygons can be easily constructed making it possible to see, estimate and compare errors or proximity *interactively*.

It can be proved that in 3-D the set of pairs of points representing the family of proximate planes form two convex hexagons when $c_0 = 1$ with an example is shown in Fig. 11.30 (right), and are contained in octagons each with two vertical edges for varying c_0. In general, a family of proximate hypeplanes in N-D is represented by $N - 1$ convex $2N$-agons when $c_0 = 1$ or $2(N + 1)$-agons for c_0 varying. These polygonal regions can be constructed with $O(N)$ computational complexity. Choosing a point in one of the polygonal regions, an algorithm matches the possible remaining $N - 2$ points, one each from the remaining convex polygons, which represent and identify hyperplanes in the family by $N - 1$ points.

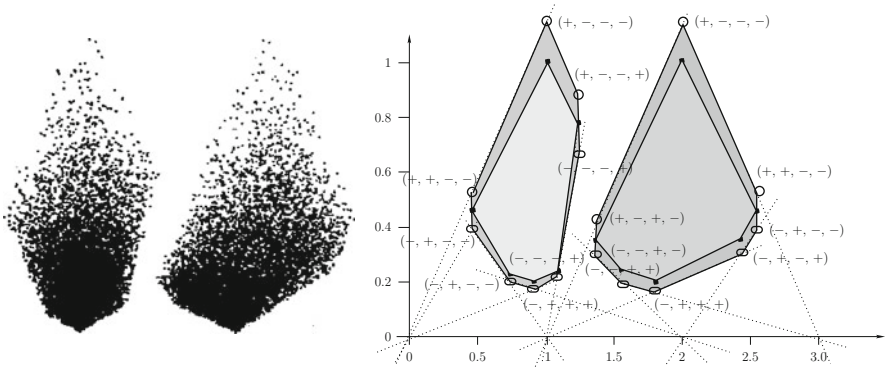


Fig. 11.30 A family of close planes.
(*Left*) Pair of point clusters representing close planes. (*Right*) The hexagonal regions (*interior*) are the regions containing the points $\bar{\pi}_{123}$ (*left*) and $\bar{\pi}_{1'23}$ for the family of planes with $c_0 = 1$ and $c_1 \in [1/3, 1.5], c_2 \in [1/3, 2.5], c_3 \in [1/3, 1]$. For c_0 varying, here $c_0 \in [0.85, 1.15]$, the regions (*exterior*) are octagonal with two vertical edges

We pose the thesis that visualization is not about seeing lots of things but rather discovering **relations** among them. While the display of randomly sampled *points* from a family of proximate hyperplanes is utterly chaotic (the mess in Fig. 11.28 (right) from points in just *one* plane), their **proximate coplanarity relation** corresponds to a clear and compact pattern. With ∥-cs we can focus and *concentrate* the relational information into patterns rather than wallow in the details, ergo the remark "without loss of information" when referring to ∥-cs. This is the methodology's real strength and where he future lies. Here then is a visualization challenge: how else can proximate coplanarity be detected and seen?

11.5.3 Nonlinear Multivariate Relations: Hypersurfaces

A relation among 2 real variables is represented geometrically by a unique region in 2-D. Analogously, a relation between N variables corresponds to a hypersurface in N-D, hence the need to say something about the representation of hypersurfaces in ∥-cs. A smooth surface in 3-D (and also N-D) can be described as the envelope of all its tangent planes. This is the basis for the representation shown in Fig. 11.31. Every point of the surface is mapped into the two points representing the *tangent plane at the point*. This generates 2 planar regions and for N-D there are $N-1$ such regions. These regions are *linked*, just as the polygons above, to provide the proper $N-1$ points representing each tangent hyperplane and from which the hypersurface can be reconstructed. Classes of surfaces can be immediately distinguished from their ∥-cs display. For developable surfaces the regions consists of boundary curves only with

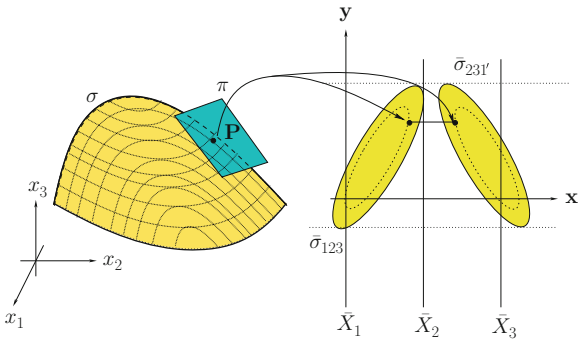


Fig. 11.31 Surface representation. A smooth surface σ is represented by two planar regions $\bar{\sigma}_{123}, \bar{\sigma}_{231'}$ consisting of pairs of points representing its tangent planes

no interior points, regions for ruled surfaces have grids consisting of straight lines, quadric surfaces have regions with conic boundaries; these are some examples.

There is a simpler but inexact surface representation which is useful when used judiciously as in the previous example Fig. 11.26. The polygonal lines representing points on the boundary are plotted and their envelope "represents" the surface; the " " are a reminder that this is not a *unique* representation. In Fig. 11.32 (left) are the upper and lower envelopes for a sphere in 5-D consisting of 4 overlapping hyperbolae which must be distinguished from those in Fig. 11.31 (right), which is exact and, interestingly enough are also hyperbolae, the curves determined by points representing the sphere's *tangent planes*. Retaining the exact surface description (i.e. its equation) internally, interior points can be constructed and displayed as shown for the 5-D sphere in Fig. 11.32 (left). On the right the same construction is shown but for a more complex 20-dimensional convex hypersurface ("model"). The intermediate curves (upper and lower) also provide valuable information and previews of coming attractions. They indicate a neighborhood of the point (represented by the polygonal line) and provide a feel for the local curvature. Note the narrow strips around $X13, X14, X15$ (as compared to the surrounding ones), indicating that at this state these are the critical variables where the point is bumping the boundary. A theorem guarantees that a polygonal line which is in-between all the intermediate curves/envelopes represents an interior point of the hypersurface and all interior points can be found in this way. If the polygonal line is tangent at anyone of the intermediate curves then it represents a boundary point, while if it crosses anyone of the intermediate curves it represents an exterior point. The later enables us to see, in an application, the first variable for which the construction failed and what is needed to make corrections. By varying the choice of value over the available range of the variable interactively, sensitive regions (where small changes produce large changes downstream) and other properties of the model can be easily discovered. Once the construction of a point is completed it is possible to vary the values of each variable and see how this effects the remaining variables. So one can do trade-off analysis in this way and provide a powerful tool for, Decision Support,

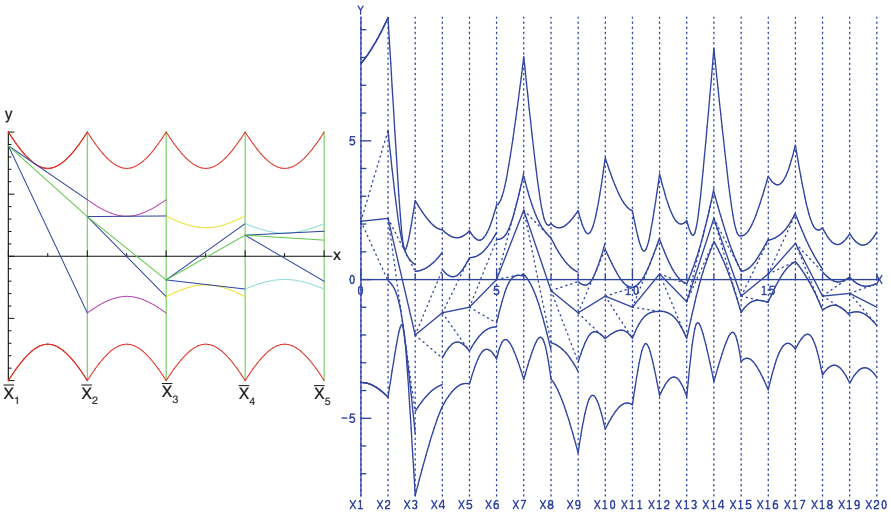


Fig. 11.32 Interior point construction.
The interior point (*polygonal line*) construction on a sphere in 5-D (*left*) and for a convex hypersurface in 20-D (*right*)

Process Control and other applications. As new data becomes available the model can be updated so that decisions are made based on the most recent information. This algorithm is used in the earlier example, shown in Fig. 11.26, to interact with a model of a country's economy.

11.6 Future

We are drowing in data and starving for knowledge

*S*earching for *patterns* in the ‖-cs data display is what skillful exploration is about. If there are multivariate relations in the display the patterns *are there* though they may be obscured by overlaps and that is not all. Our vision is not multidimensional. We do not perceive a room which is 3-dimensional from its points which are 0-dimensional, but from the 2-dimensional planes which enclose and define it. The recursive construction algorithm does exactly that for the visualization of p-dimensional objects from their $p-1$-dimensional subsets; one dimension less. We advocate including this algorithm within our armory of interactive analysis tools – **recursive interactivity**! Revisit Figs. 11.27 and 11.28 to note that relational information resides at the ***crossings***. Whatever p-dimensional relations exist are revealed by the pattern from the representation of the tangent hypeplanes of the corresponding hypersurface. The polygonal lines are completely discarded for the *relation is concentrated in the pattern*: linear relations into points, proximate coplanarity into convex polygons, quadrics into conics and more.

What can be achieved in the representation of complex relations by patterns is illustrated with some examples (a comprehensive coverage of surface representation is in Hung and Inselberg 2007) in Figs. 11.33–11.36, bumps, cusps, folds, non-orientability, hypersphere, convex and non-convex surfaces. Many of these results were *first discovered visually* and then lead to mathematical proofs; in the true spirit of Geometry. These are state of the art results showing what is achievable and how easily it generalizes to N-D. Can one imagine a higher dimensional non-orientable surface like the Möbius strip, non-convexities (bumps, crevices etc) which unlike projections *are not hidden* in the representation. New vistas for visualization emerge, transforming (rotating, translating) objects in N-space seeing the result in the representations, exploring for invariants which characterize surface properties (i.e. convexity, ruled etc), developing intelligent agents to aid the exploration, classifying objects according to application-specific taxonomies and more. The challenge is speeding up the algorithms to reach real-time performance breaching the gridlock of multidimensional visualization. The secrets of massive

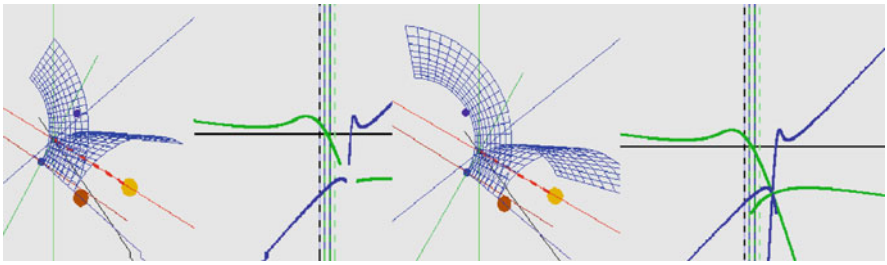


Fig. 11.33 A developable surface is represented by 2 curves. The line of cusps (left) is represented by the 2 inflection points (one on each curve) (right). The crossing curves represent the plane tangent to both leaves of the surface – a *bitangent*. The green and blue curves represent the $\bar{\pi}_{0'}$ and $\bar{\pi}_{1'}$ points respectively. The corresponding hypersurface in N-D is represented by $(N-1)$ such curves

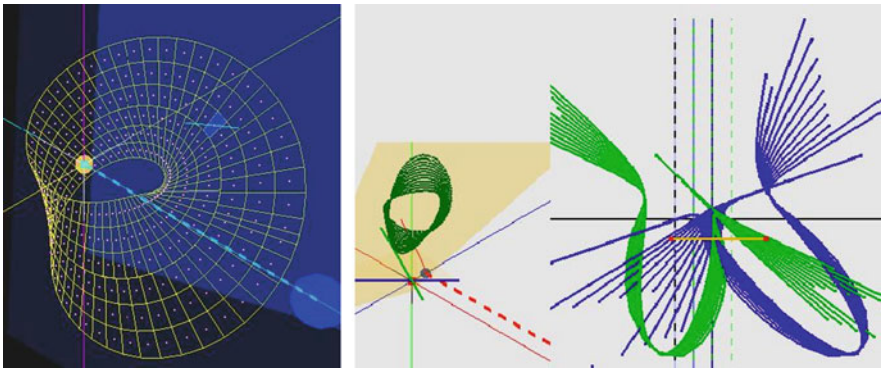


Fig. 11.34 Möbius strip and its representation. The two cusps on the left show that it corresponds to an "inflection-point in 3-D" the twist – see the duality in Fig. 11.33

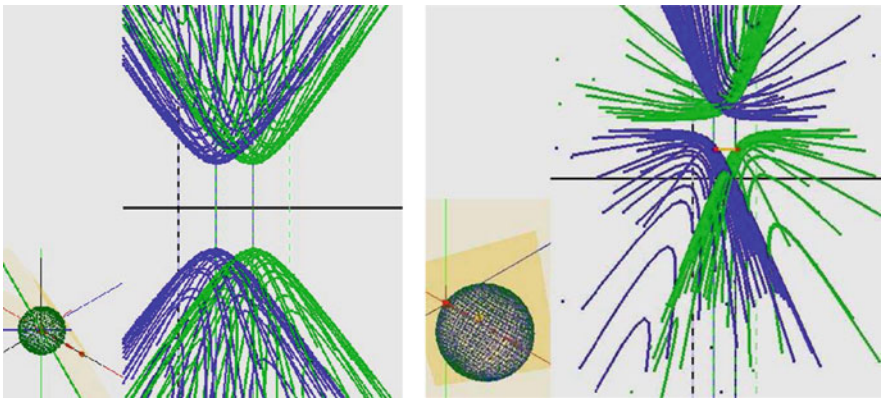


Fig. 11.35 Representation of a sphere centered at the origin (*left*) and after a translation along the x_1 axis (*right*) causing the two hyperbolas to rotate in opposite directions. This is an instance of the *translation* $\leftrightarrow$ *rotation*. In $N - D$ a sphere is represented by $N - 1$ hyperbolae – see Fig. 11.32

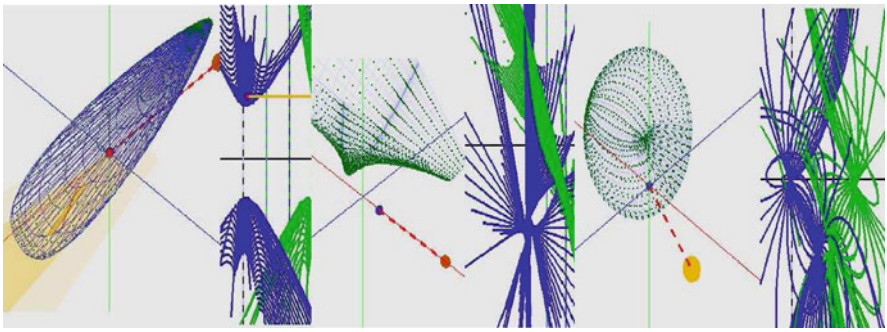


Fig. 11.36 Convex surfaces are represented by hyperbolic-like regions (*left*).
Non-convexities: "bump" (*center*), three "dimples" represented by swirls (*right*)

datasets can then be unveiled and the multidimensional relations *seen* as patterns – their *multidimensional graphs*.

Acknowledgements I am grateful to David Adjiashvili who wrote the magnificent interactive software diplaying the ‖-cs representation of surfaces seen in Fig. 11.33 through 11.36. Senior Fellow San Diego SuperComputing Center & Multidimensional Graphs Ltd, Raanana 43556, Israel

References

Agarwal, R., Gehrke, J.E., Gunopoulos, D., Raghavan, P.: Automatic Subspace Clustering of High Dimensional data for Data Mining. USA Patent 6003029 (1999)
Bollobas, B.: Graph Theory. Springer, New York (1979)

Chatterjee, A.: Visualizing Multidimensional Polytopes and Topologies for Tolerances. Ph.D. thesis, Department of Computer Science, University of Southern California (1995)
Chatterjee, A., Das, P.P., Bhattacharya, S.: Visualization in linear programming using parallel coordinates. Pattern Recogn. **26-11**, 1725–36 (1993)
Choi, H., Lee, H.: PCAV: Internet Attack Visualization in Parallel Coordinates, LNCS 3783, 454–466. Springer, New York (2005)
Chomut, T.: Exploratory Data Analysis in Parallel Coordinates. M.Sc. thesis, Department of Computer Science, UCLA (1987)
Cohan, S.M., Yang, D.C.H.: Mobility analysis in parallel coordinates. J. Mech. Mach. **21**, 63–71 (1986)
Conti, G.: Security Data Visualization. No Starch Press, San Francisco (2007)
Desai, A., Walters, L.C.: Graphical representation of data envelopment analyses:management implications from parallel axes representations. Dec. Scien. **22**(2), 335–353 (1991)
Eickemeyer, J.: Visualizing p-flats in N-space using Parallel Coordinates. Ph.D. thesis, Department of Computer Science, UCLA (1992)
Fiorini, P., Inselberg, A.: Configuration Space Representation in Parallel Coordinates. IEEE Conf. Rob. Aut. 1215–1220 (1989)
Friendly, M., al: Milestones in Thematic Cartography. www.math.yorku.ca/scs/SCS/Gallery/milestones/ (2005)
Gennings, C., Dawson, K.S., Carter, W.H., Myers, R.H.: Interpreting plots of a multidimensional dose-response surface in parallel coordinates. Biometrics **46**, 719–35 (1990)
Han, J., Kamber, M.: Data Mining Concepts and Technology. Morgan-Kaufman, San Francisco (2001)
Harary, F.: Graph Theory. Addison-Wesley, Reading, Mass (1969)
Hauser, H.: Parallel Sets: Visual Analysis of Categorical Data. Proceedings of IEEE Infovis (2005)
Hung, C.K., Inselberg, A.: Parallel Coordinate Representation of Smooth Hypersurfaces. USC Tech. Report # CS - 92 -531, Los Angeles (1992)
Hung, C.K., Inselberg, A.: Description of Surfaces in Parallel Coordinates by Linked Planar Regions, Mathematics of Surfaces XII, 177-208, LNCS 4647. Springer, New York (2007)
Hurley, C.B., Olford, R.W.: Pairwise Display of High-Dimensional Information via Eulerian Tours and Hamiltonian Decompositions, Journal of Computational and Graphical Statistics **19**(4), 861–886 (2010).
Inselberg, A.: The plane with parallel coordinates. Vis. Comput. **1**, 69–97 (1985)
Inselberg, A.: Multidimensional Detective, in Proceedings of IEEE Information Visualization '97, 100-107. IEEE Computer Society, Los Alamitos, CA (1997)
Inselberg, A.: Parallel Coordinates : **VISUAL** Multidimensional Geometry and its Applications. Springer, New York (2009)
Inselberg, A., Avidan, T.: The Automated Multidimensional Detective, In Proceedings of IEEE Information Visualization '99, 112-119. IEEE Computer Society, Los Alamitos, CA (1999)
Inselberg, A., Avidan, T.: Classification and Visualization for High-Dimensional Data, In Proceedings of KDD, 370-4. ACM, New York (2000)
Inselberg, A., Boz, M., Dimsdale, B.: Planar Conflict Resolution Algorithm for Air-Traffic Control and the One-Shot Problem, in IBM PASC Tech. Rep. G320-3559. IBM Palo Alto Scientific Center (1991)
Inselberg, A., Dimsdale, B.: Parallel Coordinates: A Tool For Visualizing Multi-Dimensional Geometry, Proceedings of IEEE Conference on Visualization, 361-378. IEEE Computer Society, Los Alamitos, CA (1990)
Inselberg, A., Reif, M., Chomut, T.: Convexity algorithms in parallel coordinates. J. ACM **34**, 765–801 (1987)
Jones, C.: Visualization and Optimization. Kluwer Academic Publishers, Boston (1996)
Matskewich, T., Inselberg, A., Bercovier, M.: Approximated Planes in Parallel Coordinates. In Proceedings of Geometry Modeling Conference, St. Malo, Vanderbilt University Press, 257–266 (2000)

Schmid, C., Hinterberger, H.: Comparative Multivariate Vis. Across Conceptually Different Graphic Displays, in Proceedings of 7th SSDBM. IEEE Computer Society, Los Alamitos, CA (1994)

Theus, M., Urbanek, S.: Interactive Graphics for Data Analysis. CRC Press, Boca Raton FL (2009)

Tufte, E.R.: Visual Explanation. Graphic Press, Connecticut (1996)

UCI. Machine Learning Database Repository at. www.ics.uci.edu/~mlearn/MLRepository.html.

Ward, M.O.: XmdvTool: integrating multiple methods for visualizing multivariate data, Proceedings IEEE Conference on Visualization, CA, 326-333. IEEE Computer Society, Los Alamitos, CA (1994)

Wegman, E.: Hyperdimensional data analysis using parallel coordinates. J. Am. Stat. Assoc. **85**, 664–675 (1990)

Chapter 12
Interactive and Dynamic Graphics

Jürgen Symanzik

12.1 Introduction

Interactive and dynamic statistical graphics enable data analysts in all fields to carry out visual investigations leading to insights into relationships in complex data. Interactive and dynamic statistical graphics involve methods for viewing data in the form of point clouds or modeled surfaces. Higher-dimensional data can be projected into one-, two- or three-dimensional planes in a set of multiple views or as a continuous sequence of views which constitutes motion through the higher-dimensional space containing the data.

Strictly, there is some difference between interactive graphics and dynamic graphics. When speaking of interactive graphics only, we usually mean that a user actively interacts with, i.e., manipulates, the visible graphics by input devices such as keyboard, mouse, or others and makes changes based on the visible result. When speaking of dynamic graphics only, we usually mean that the visible graphics change on the computer screen without further user interaction. An example for interactive graphics might be the selection of interval lengths and starting points when trying to construct an optimal histogram while looking at previous histograms. An example for dynamic graphics might be an indefinitely long grand tour with no user interaction. Typically, interactive graphics and dynamic graphics are closely related and we will not make any further distinction among the two here and just speak of interactive and dynamic statistical graphics.

Two terms closely related to interactive and dynamic statistical graphics are exploratory data analysis (EDA) and visual data mining (VDM).

EDA, as defined by Tukey (1977), "is detective work—numerical detective work—or counting detective work—or graphical detective work." Modern techniques and software in EDA, based on interactive and dynamic statistical

J. Symanzik (✉)
Department of Mathematics and Statistics, Utah State University, Logan, UT, USA
e-mail: juergen.symanzik@usu.edu

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_12,

graphics, are a continuation of Tukey's idea to use graphics to find structure, general concepts, unexpected behavior, etc. in data sets by looking at the data. To cite Tukey (1977) again, "today, exploratory and confirmatory can—and should—proceed side by side." Interactive and dynamic statistical graphics should not replace common analytic and inferential statistical methods—they should rather extend these classical methods of data analysis.

Data mining (DM) itself (Klösgen and Zytkow 2002; Witten and Frank 2000), see also Chap. III.13, is a field whose scientific basis has only began to be explored over the last few years. DM exists as a result of the convergence of several fields including data bases, statistics, and artificial intelligence. Friedman (1998) discusses the connection between DM and statistics in more details and Wegman (2000) provides a definition of DM that links it with EDA and graphics: "Data mining is exploratory data analysis with little or no human interaction using computationally feasible techniques, i.e., the attempt to find interesting structure unknown a priori." Simultaneously with an increasing interest in DM there has been the evolution of computer graphics, especially in the area of virtual reality (VR). Within the statistics framework, the area of EDA has evolved into a more sophisticated area of interactive and dynamic statistical graphics. Recently, DM has been combined with statistical graphics, resulting in VDM (Böhlen et al. 2003; Cox et al. 1997; Inselberg 1998; Macedo et al. 2000; Symanzik et al. 1999a). However, there exist several different definitions of the term VDM. Soukop and Davidson (2002) dedicate less than one page to "dynamic visualizations that allow user interaction" in their book on VDM.

In this chapter we will provide a general overview on existing methods and software for interactive and dynamic graphics. We will also provide a snapshot of current developments that may become a standard in the near future but may also be quickly forgotten again. All sections are supported by an extensive list of references that will allow every reader from novice to expert to become more familiar with a particular concept of interactive and dynamic graphics. More specifically, in Sect. 12.2, we will discuss early developments and software related to interactive and dynamic graphics. In Sect. 12.3, we will discuss the main concepts and in Sect. 12.4 some software products related to interactive and dynamic graphics. Interactive 3D graphics will be discussed in Sect. 12.5 and applications of interactive and dynamic graphics in geography, medicine, and environmental sciences will be discussed in Sect. 12.6. We conclude this chapter with an outlook on possible future developments in Sect. 12.7.

All graphical displays throughout this chapter are based on the "Places" data set that was distributed to interested members of the American Statistical Association (ASA) several years ago so that they could apply contemporary data analytic methods to describe these data and then present results in a poster session at the ASA annual conference. The data are taken from the Places Rated Almanac (Boyer and Savageau 1981). The data are reproduced on disk by kind permission of the publisher, and with the request that the copyright notice of Rand McNally, and the names of the authors appear in any paper or presentation using these data. The data consist of nine measures of livability for 329 cities in the U.S.: Climate and Terrain, HousingCost, Health Care and Environment, Crime, Transportation, Education,

The Arts, Recreation, and Economics. For all but two of the above criteria, the higher the score, the better. For HousingCost and Crime, the lower the score the better. The scores are computed using several statistics for each criterion (see the Places Rated Almanac for details). Latitude and longitude have been added by Paul Tukey. Population numbers have been added as well.

12.2 Early Developments and Software

There is a strong history of statistical graphics research on developing tools for visualizing relationships between many variables. Much of this work is documented in videos available from the ASA Statistical Graphics Section Video Library at http://stat-graphics.org/movies/

Additional material on statistical graphics can also be found in journals such as "Journal of Computational and Graphical Statistics", "Computational Statistics", and "Computational Statistics & Data Analysis" and in "Computing Science and Statistics", the proceedings of the Interface conferences. The following paragraphs only serve as a basic overview for readers unfamiliar with dynamic statistical graphics but they are not intended as a full introduction into this topic.

A video clip of the successive stages in a multidimensional scaling algorithm (Kruskal 1962) is one of the first examples how to apply dynamic statistical graphics. A second example by Chang (1970) shows an interactive search for a structured two-dimensional projection in five dimensions where three of the five dimensions are noise. PRIM-9 (Picturing, Rotation, Isolation and Masking in up to 9 dimensions), documented in Fisherkeller et al. (1974a) and Fisherkeller et al. (1974b), is the landmark example of early dynamic statistical graphics. Projections formed the fundamental part of the visualization system and were complemented with isolation and masking. A good explanation of the importance of projection as a tool for visualizing structure in high-dimensional data can be found in Furnas and Buja (1994).

One major breakthrough in using projections for visualizing higher dimensions was made by Asimov (1985) in his work on the grand tour. The grand tour, further exploited in Buja and Asimov (1986a), in an abstract sense shows a viewer all possible projections in a continuous stream (which could be considered to be moving planes through p-dimensional space). Several possibilities for "showing all possible projections" were explored in the original work, but the most successful method to arise from it is based on interpolating between random planes. Another common approach to displaying high-dimensional data can be found in Becker and Cleveland (1988) where data is plotted in a scatterplot matrix, i.e., a matrix of pairwise scatterplots. Users can do linked brushing among the plots, i.e., mark points with different symbols and colors, while this information is also immediately displayed in all related (linked) plots.

The historical development of interactive and dynamic statistical graphics is well documented in a series of books and articles. Chambers et al. (1983) and du Toit et al. (1986) can be placed somewhere inbetween Tukey's original idea of EDA and the beginning of modern dynamic and interactive statistical graphics. Wegman and DePriest (1986) is a collection of papers presented at a workshop sponsored by the Office of Naval Research (ONR), held in Luray, Virginia, from 24 through 27 May, 1983. About half of the papers are related to statistical image processing while the other half is related to (interactive) statistical graphics. Cleveland and McGill (1988) contains a collection of papers about dynamic graphics for statistics, originally published between 1969 through 1988. This book is a very good reference to see the progress in dynamic graphics concepts and software over two decades, starting from the very early stages through the late 1980s. Buja and Tukey (1991) is based on the proceedings of the Institute for Mathematics and its Applications (IMA) 1989 summer program on "Robustness, Diagnostics, Computing and Graphics in Statistics". An earlier "Handbook of Statistics, Volume 9, Computational Statistics", edited by Rao (1993), contains several then state-of-the-art overviews on interactive and dynamic statistical graphics, most notably the chapters by Wegman and Carr (1993) and Young et al. (1993). Nagel et al. (1996) dedicate two (out of six) chapters of their book to dynamic graphics—one being an overview and one discussing applications. Theus (1996) is fully dedicated to the theory and applications of interactive statistical graphics. Wilhelm et al. (1996) contains reviews of software for interactive statistical graphics.

Major statistical journals often dedicate special issues to interactive and dynamic graphics, e.g., "Computational Statistics" (Volume 14, Issue 1, 1999) on "Interactive Graphical Data Analysis" and "Computational Statistics & Data Analysis" (Volume 43, Number 4, 2003) on "Data Visualization". A strong case for the use of statistical graphics has been made by Andreas Buja (Symanzik 2008).

12.3 Concepts of Interactive and Dynamic Graphics

This section will provide some deeper insights into concepts of interactive and dynamic graphics mentioned in the previous sections. Buja et al. (1996) contains a taxonomy of interactive data visualization based on the notions of focusing, linking, and arranging views of data. Unwin (1999) discusses some of the main concepts in the context of interactive graphics software.

12.3.1 Scatterplots and Scatterplot Matrices

Perhaps the most basic concepts for statistical graphics are scatterplots (see Figs. 12.1–12.3 and 12.4). In a simple scatterplot, we place different symbols (sometimes also called glyphs) at x- and y-positions in a two-dimensional plot

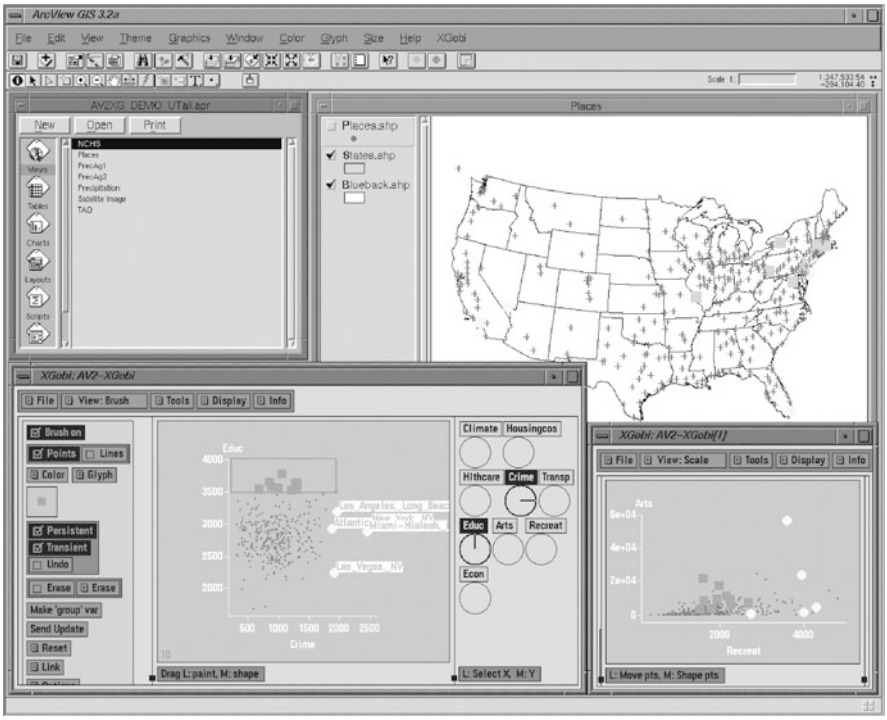


Fig. 12.1 Screenshot of the "Places" data in ArcView/XGobi. A map view of the 329 spatial locations is displayed in ArcView at the top. The two XGobi windows at the bottom are showing scatterplots of Crime (*horizontal*) vs. Education (*vertical*) [*left*] and Recreation (*horizontal*) vs. Arts (*vertical*) [*right*]. Locations of high Crime have been brushed and identified, representing some of the big cities in the U.S. Also, locations of high Education (above 3,500) have been brushed, mostly representing locations in the northeastern U.S. All displays have been linked

area. These positions are determined by two of the variables. The type, size, and color of the symbols may depend on additional variables. Usually, explanatory information such as axes, labels, legends, and titles are added to a scatterplot. Additional information such as a regression line or a smoothed curve can be added as well.

If the data consist of more than two variables (e.g., somewhere between three to ten), the data can be displayed by a scatterplot matrix (see Figs. 12.2 and 12.3) that shows all pairwise scatterplots of the variables. The essential property of a scatterplot matrix is that any adjacent pair of plots have one of their axes in common. When plotting the full array of all $n \times (n-1)$ pairwise scatterplots, each plot in the upper triangle of plots has a matching plot in the lower triangle of plots, with the exception that the axes in these pairs of plots have been flipped. Therefore, sometimes only the upper or lower triangle of scatterplots is displayed; thus gaining plotting speed and allowing each individual plot to be somewhat larger.

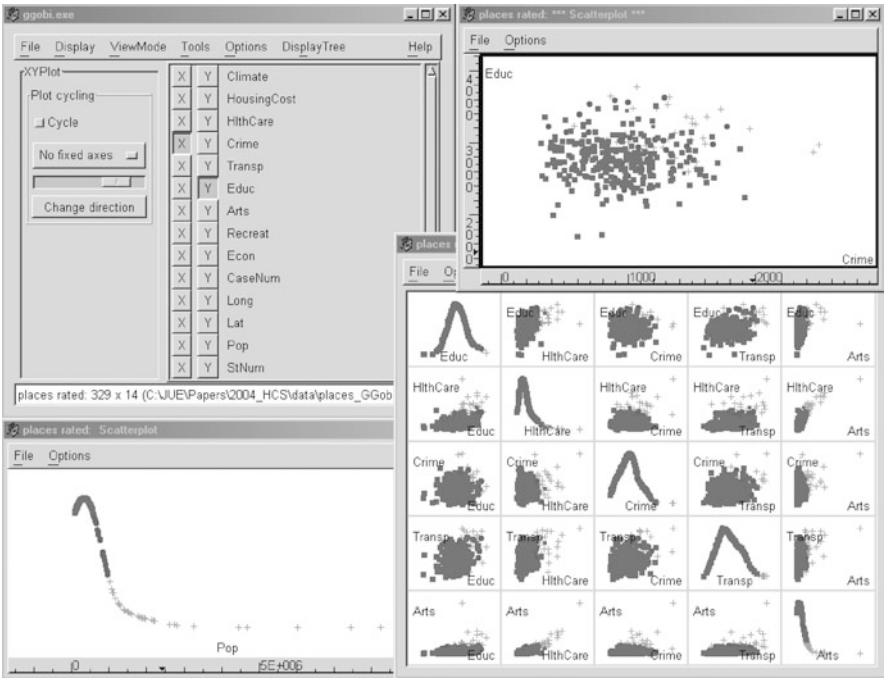


Fig. 12.2 Screenshot of the "Places" data in GGobi. A scatterplot of Crime (*horizontal*) vs. Education (*vertical*) is displayed at the top right, a scatterplot matrix of five of the variables is displayed at the bottom right, and a density (*1D*) plot of Population is displayed at the bottom left. The data has been brushed with respect to Population: one group for a Population less than 500,000, one group for a Population between 500,000 and 1,000,000, and one group for a Population above 1,000,000. The scatterplot of Crime and Education seems to reveal that higher Population is associated with higher Crime and higher Education. The scatterplot matrix seems to reveal that higher Population is also associated with higher Arts and higher HealthCare. All displays have been linked.

Early examples of scatterplot matrices can be found in Chambers et al. (1983) and Cleveland (1985) for example. In fact, Chambers et al. (1983) initially called an array of pairwise scatterplots for three variables a draftsman's display and for four (or more) variables a generalized draftsman's display. In their (generalized) draftsman's display, each point is plotted with the same symbol. When encoding additional information through the use of different plotting symbols, Chambers et al. (1983) speak of symbolic (generalized) draftsman's displays. Today, we hardly make any distinction of these different types of displays and just speak of scatterplot matrices.

Murdoch (2002) and Unwin (2002) discuss features good scatterplots and related interactive software should provide, e.g., meaningful axes and scales, features for rescaling and reformatting, good handling of overlapping points and missing data, panning and zooming, and querying of points. Carr et al. (1987) describes techniques for scatterplot matrices particularly useful for large numbers of observations.

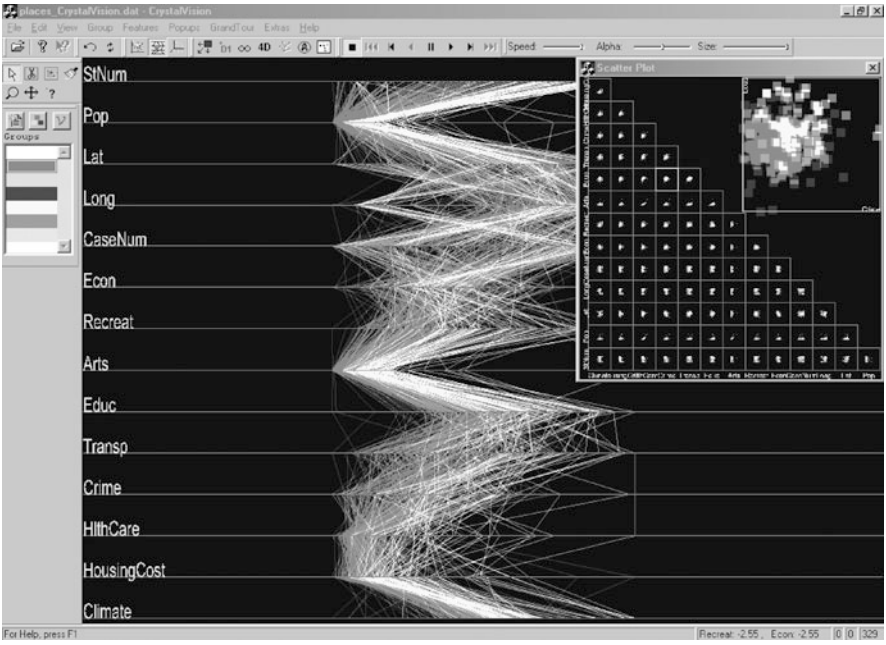


Fig. 12.3 Screenshot of the "Places" data in CrystalVision. A parallel coordinate plot of all variables is shown as the main plot. A scatterplot matrix of all variables with a scatterplot of Crime (*horizontal*) vs. Education (*vertical*) is shown as a popup in the top right. The data has been brushed according to high and low Population. According to the parallel coordinate plot, higher Population is associated with higher Arts and HousingCost. The scatterplot of Crime and Education seems to reveal that higher Population is also associated with higher Crime and higher Education.All displays have been linked

12.3.2 Brushing and Linked Brushing/Linked Views

Brushing, as introduced in Becker and Cleveland (1988) and Becker et al. (1988b), initially was considered as a collection of several dynamic graphical methods for analyzing data displayed in a scatterplot matrix. The central idea behind brushing is a brush, usually a rectangular area on the computer screen, that is moved by the data analyst to different positions on the scatterplot or any other graphical display. Four brushing operations were introduced in Becker and Cleveland (1988): highlight, shadow highlight, delete, and label. The most commonly used brushing technique is highlighting—often in the context of linked brushing, i.e., for linked views. All points that are inside the brush in the currently selected display are highlighted, i.e., marked with a different symbol or color. Simultaneously, points that correspond to those points are automatically highlighted with the same symbol/color in all linked views.

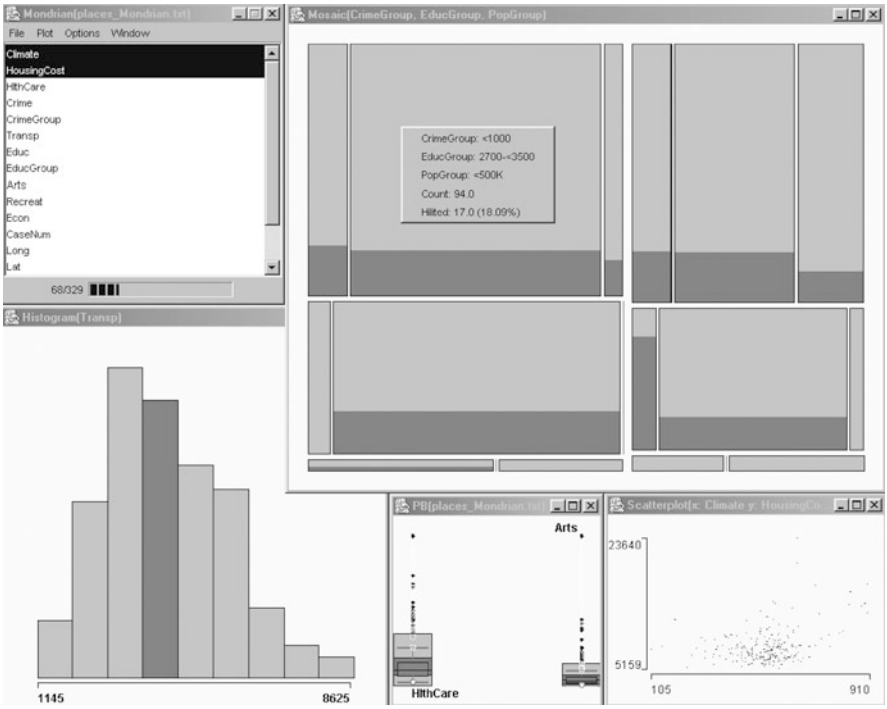


Fig. 12.4 Screenshot of the "Places" data in Mondrian. The variables Crime, Education, and Population have been discretized for this figure. A mosaic plot of Crime (first vertical division, grouped as below 1,000 [*left*] and above 1,000 [*right*]), Education (first horizontal division, grouped as 2,700 to 3,500 [*top*], below 2,700 [*middle*], and above 3,500 [*bottom*]), and Population (second vertical division, grouped as 500,000 to 1,000,000 [*left*], below 500,000 [*middle*], and above 1,000,000 [*right*]) is displayed at the top right. A histogram of Transportation is shown at the bottom left, boxplots of HealthCare and Arts are shown at the bottom middle, and a scatterplot of Climate (*horizontal*) vs. HousingCost (*vertical*) is shown at the bottom right. The mosaic plot shows that Crime, Education, and Population are not independent. The different displays show how average Transportation (that has been brushed in the histogram) is related to the other variables. All displays have been linked

A very useful brushing technique is the transient paint mode. As the brush is moved, the new points that come inside the brush are highlighted while points that move outside the brush are no longer highlighted.

While brushing initially was only developed for scatterplot matrices, it quickly has been adapted to other types of linked graphical displays. Linked brushing among different displays is one of the most useful techniques used within dynamic and statistical graphics. Linked brushing can be applied to graphical representation of continuous data, summary data such as histograms (Stuetzle 1988), or even displays of categorical data such as mosaic plots (Hofmann 2000, 2003). All dynamic statistical graphics software packages support linked brushing among different types of graphical displays these days.

When dealing with massive data sets, it is often beneficial to focus on particular subgroups of the data and also be able to quickly return to a previous stage of the analysis. Selection sequences (Hofmann and Theus 1998; Theus et al. 1998) are an extension of the conventional linked-highlighting paradigm as they store the whole hierarchical path of a selection and allow an easy editing, redefinition, and interrogation of each selection in the path of the analysis. In a selection sequence, we can easily jump from one branch of the hierarchic selection tree to another.

12.3.3 Focusing, Zooming, Panning, Slicing, Rescaling, and Reformatting

Focusing techniques, as introduced in Buja et al. (1991), are based on the idea that it often might be easier for a human analyst to understand several individual displays, each focused on a particular aspect of the underlying data, rather than looking at the full data set. Focusing techniques include subset selection techniques, e.g., panning and zooming or slicing, and dimensionality reduction techniques, e.g., projection. Methods for focusing can be automatic, interactive, or a combination of both. While focusing shows only part of the data at a time, it is important to display multiple linked views of the data, perhaps each focusing on a different aspect of the data, to maintain the full picture of the data.

Zooming is a technique that can be used for inspecting details of the data when overplotting arises. Zooming can be done via some kind of a magnifying glass or by manually selecting subsections of the visible axes, e.g., via sliders. The main idea behind zooming is that when several points overplot in the full display, it may indeed turn out that these points are exactly the same when zooming into the neighborhood of these points—or, what most frequently happens, that these points have a particular structure and are not exactly the same.

Panning is closely related to zooming. An analyst should know which subset of the data is currently visible. Therefore, an information plot should reveal the current location on which subregion we have zoomed.

Slicing, as described in Furnas (1988) and Furnas and Buja (1994), is a technique that takes sections (or slices) of a high-dimensional data set. While slicing (and projections) are useful means for an exploratory data analysis, these techniques also have their limitations. However, these limitations may be overcome by combining slicing and projections in so-called prosections (Furnas and Buja 1994). An extension of individual prosection views is the prosection matrix (Tweedie and Spence 1998), some kind of a density plot summarizing multi-dimensional volumetric information. The prosection matrix is a useful representation for engineering design, allowing an analyst to interactively find a design that leads to a maximal manufacturing yield.

Rescaling is a technique to quickly change the scale of the displayed variables, e.g., by taking the log, square root, standardize, or by mapping to a 0–1 scale. When

looking at multiple variables, it might also be beneficial to have a common scale (from the minimum across all variables to the maximum across all variables). By interactively rescaling variables, an analyst may identify useful transformations for a follow-up modeling step of the data.

Reformatting includes features as simple as swapping x and y axes in a scatterplot or changing the order of coordinate axes in a parallel coordinate plot.

Unwin (2002) provides more details on several of the techniques described above.

12.3.4 Rotations and Projections

Rotation, as introduced in Fisherkeller et al. (1974b) and later refined in Becker et al. (1988b), is a very powerful tool for understanding relationships among three or more variables. The familiar planar scatterplot is enhanced by rotation to give the illusion of a third dimension. We typically rotate plots in search of some interesting views that do not align with the plot axes and therefore cannot be seen in a scatterplot matrix. Usually, a three-dimensional point cloud representing three of the variables is shown rotating on a computer screen. The rotation shows us different views of the points and it produces a 3D effect while moving, allowing us to see depth. Basic rotation controls with a mouse have been introduced in Becker et al. (1988b).

Mathematically speaking, each rotation within a 3D space onto a 2D computer screen is based on a projection. Obviously, it is mathematically possible to project high-dimensional data onto low-dimensional subspaces and gain insights into the underlying data through dynamic visualizations of such projections. One particular example of a continuous sequence of projections, the grand tour, will be discussed in the next section. Cook and Buja (1997) discuss methods how to manually control high-dimensional data projections. Cook (1997) provides a variety of training data sets that help new users get a visual feeling of the underlying high-dimensional data set when seen as a projection into low-dimensional space.

12.3.5 Grand Tour

Often, simple plot rotation, as discussed in the previous section, does not suffice to see all interesting views of the data. To produce a plethora of possible interesting views, the grand tour has been introduced in Asimov (1985) and Buja and Asimov (1986b). In Asimov (1985), the grand tour has been described as "a method for viewing multivariate statistical data via orthogonal projections onto a sequence of two-dimensional subspaces. The sequence of subspaces is chosen so that it is dense in the set of all two-dimensional subspaces." Some of the features the grand tour can be used for are examining the overall structure and finding clusters or outliers in high-dimensional data sets.

In the context of the grand tour, an alternating sequence of brushing, looking at additional projections from the grand tour, brushing, and so on, is referred to as the brush-tour strategy in the remainder of this chapter. We can only be sure that a cluster visible in one projection of the grand tour really is a cluster if its points remain close to each other in a series of projections and these points move similarly when the grand tour is activated. If points move apart, we probably found several subclusters instead of one larger cluster.

Wegman (1992) discusses a form of the grand tour for general d-dimensional space. The algorithms for computing a grand tour are relatively computationally intensive. Wegman and Shen (1993) discuss an approximate one- and two-dimensional grand tour algorithm that was much more computationally efficient than the Asimov winding algorithm. That algorithm was motivated in part by a discussion of the Andrews (multidimensional data) plot, discussed in Sect. 12.3.9, which can also be regarded as a highly restricted pseudo tour.

12.3.6 Parallel Coordinate Plots

Parallel coordinate plots (Inselberg 1985, 2009; Wegman 1990) (see Fig. 12.3) are a geometric device for displaying points in high-dimensional spaces, in particular, for dimensions greater than three. The idea is to sacrifice orthogonal axes by drawing the axes parallel to each other resulting in a planar diagram where each d-dimensional point $(x_1, \ldots, x_d)$ is uniquely represented by a continuous line. The parallel coordinate representation enjoys some elegant duality properties with the usual Cartesian coordinates and allows interpretations of statistical data in a manner quite analogous to two-dimensional Cartesian scatterplots. This duality of points in Cartesian plots and lines in parallel coordinates extends to conic sections. This means that an ellipse in Cartesian coordinates maps into a hyperbola in parallel coordinates. Similarly, rotations in Cartesian coordinates become translations in parallel coordinates.

The individual parallel coordinate axes represent one-dimensional projections of the data. We can isolate clusters by looking for separation between data points on any axis or between any pair of axes. Because of the connectedness of the multidimensional parallel coordinate diagram, it is usually easy to see whether or not this clustering propagates through other dimensions.

The use of parallel coordinate plots for a d-dimensional grand tour sequence, sometimes called a parallel coordinate grand tour, has been described in Wegman (1992) and Wegman and Carr (1993). By using such a parallel coordinate grand tour, an analyst can find orientations where one or more clusters are evident. The general strategy for detecting clusters is the following: We begin with a static plot of the data in parallel coordinates. If there are any gaps along a horizontal axis (which incidentally does not need to coincide with the coordinate axes), then we color the individual clusters with distinct colors. Once all clusters are identified in the original coordinate system, we run the grand tour until an orientation of

the axes is found in which another gap in one of the horizontal axes is found. Again we color the individual subclusters with distinct colors. This procedure is repeated until no further subclusters can be identified. This is another example of the brush–tour strategy referred to in Sect. 12.3.5. Indeed, when to stop is a matter of judgement, since the procedure can be repeated until practically every data point can be individually colored. The crucial issue, which really depends on the dynamic graphics, is to see that clusters identified in this manner track coherently with the grand tour animation. That is, data points of the same color stay together as the grand tour rotation proceeds. If they do not, then there are likely to be substructures that can be identified through further grand tour exploration.

Slopes of parallel coordinate line segments can also be used to distinguish clusters. That is, if a group of line segments slopes, say, at 45° to the horizontal and another group slopes at, say, at 135° to the horizontal, then even though the lines fully overlap in both adjacent parallel coordinate axes and there is no horizontal gap, these sets of lines represent two distinct clusters of points. Fortunately, when such indication of clustering exists, the grand tour will also find an orientation of axes in which there is a horizontal gap. Thus the general strategy is to alternate color brushing of newly discovered clusters with grand tour rotations until no further clusters can be easily identified.

In some software packages, the parallel axes in a parallel coordinate plot are drawn as horizontal lines (e.g., in ExplorN) while in other software packages they are drawn as vertical lines (e.g., in XGobi). While it may be argued that this makes no difference from a mathematical point of view, the wider aspect ratio in the horizontal mode coupled with a more usual sense of plotting data along an abscissa rather than along the ordinate tends to allow for an easier human interpretation. Detailed interpretations are given in Wegman (1990).

12.3.7 Projection Pursuit and Projection Pursuit Guided Tours

While the grand tour, as discussed in Sect. 12.3.5, is a dynamic tool, projection pursuit (Friedman and Tukey 1974; Huber 1985; Kruskal 1969), see also Chap. III.6, is a static tool. Projection pursuit results in a series of static plots of projections that are classified as "interesting" with respect to a particular projection pursuit index. Many projection pursuit indexes, e.g., the ones discussed in Jones and Sibson (1987), Friedman (1987), Hall (1989), Morton (1989), Morton (1992), Cook et al. (1993), and Posse (1995), are based on the idea to search for the most non-normal projections. Usually, each projection pursuit index, a function of all possible projections of the data, results in many hills and valleys. Friedman (1987) suggests a projection pursuit algorithm that initially searches for relatively high values of the function and then starts derivative-based searches to find the global maximum.

The combination of grand tour and projection pursuit, called projection pursuit guided tour (Cook et al. 1995), helps to direct the grand tour towards "interesting" projections. This combination of the two methods into an interactive and dynamic

framework not only shows the "interesting" projections but it maintains the motion so the user has a feeling how successive "interesting" projections have been obtained.

12.3.8 Pixel or Image Grand Tours

The idea of the pixel or image grand tour (IGT) evolved from an initial application of one-dimensional tours to image data. Multiple registered images can be regarded as a multidimensional image in which each pixel location has a vector attached to it. For example, ordinary red, green, and blue (RGB) color images are vector-valued images. The basic idea of the image tour is to apply the same one-dimensional grand tour to each vector for all pixel locations in an image. This combines the vectors into a scalar function of time which can be rendered as a time-varying gray-scale image. The Wegman and Shen (1993) algorithm generalizes easily to two dimensions, so that an alternate approach to the IGT is to project the multidimensional vector into two dimensions and render the image as a false color image with two complementary colors such as red and cyan. It should be noted that red and cyan are complementary colors in the RGB color model used for most computer monitors whereas red and green are complementary colors in the conventional color model, introduced by the Commission Internationale de l' Éclairage (CIE) in 1931. A detailed comparison of these two and other color models can be found in Foley et al. (1990), Chap. 13. The initial discussion of the IGT was given by Wegman et al. (1998). Additional examples of the IGT can be found in Symanzik et al. (2002b).

Currently, the IGT software, written in C++ by Qiang Luo, is available for Silicon Graphics, Inc., (SGI) workstations. To obtain a fast rendering rate of large images, the software intensively uses SGI hardware features such as the α-channel hardware. There exists also a MATLAB version of the IGT written by Wendy Martinez. Both versions of the IGT software are not accessible through a Web site but can be obtained from the corresponding software developers.

12.3.9 Andrews Plots

The Andrews (multidimensional data) plot, as introduced in Andrews (1972) is based on a series of Fourier interpolations of the coordinates of multi-dimensional data points. Points that are close in some metric will tend to have similar Fourier interpolations and therefore will tend to cluster in the Andrews plot. Thus, the Andrews plot is an informative graphical tool most useful to detect clustering.

Ideas underlying the Andrews plot and the grand tour are quite similar. However, in contrast to the grand tour, the Andrews plot is a static plot while the grand tour is dynamic. Although dynamic renditions of the Andrews plot exist, and

these sometimes also are (incorrectly) referred to as one-dimensional grand tour (Crawford and Fall 1990), the Andrews plot is not a grand tour since it cannot sweep out all possible directions as pointed out in Wegman and Shen (1993). Three-dimensional generalizations of the Andrews plot and other pseudo grand tours have been introduced in Wegman and Shen (1993) as well.

12.3.10 Density Plots, Binning, and Brushing with Hue and Saturation

Carr et al. (1987) present techniques for visualizing data in scatterplots and scatterplot matrices when the data consists of a large number of observations, i.e., when overplotting of points frequently occurs using standard techniques. A key idea to address in the visualization of a large number of observations is based on the estimation and representation of densities. For this purpose, the data is often binned into an $n \times n$ matrix for two-dimensional representation (or an $n \times n \times n$ matrix for three-dimensional representation). Possibilities to visualize the number of data points in each bin can be based on gray-scale (or color) density representations or by symbol area such as using differently sized hexagon symbols, where the area of the plot symbol is proportional to the count in each bin. Carr (1991) further extends these ideas and presents additional low-dimensional displays for data that consists of a large number of observations. Scott (1992) provides a general overview on techniques for density estimation, including averaged shifted histograms (ASH) and kernel density estimators, including possible visualization techniques via contour surfaces, (transparent) α-level contours, and contour shells. Further details on multivariate density estimation and visualization can be found in Chap. III.4.

Wegman and Luo (1997a) use hue and saturation for plotting and brushing. For each individual point, the hue is almost fully desaturated with black. When points are overplotted, the hue components are added. The saturation level should be interactively adjustable by the analyst. If many points overplot, the pixel will be fully saturated. If fewer points overplot, the pixel will be shown in a less saturated color. Often, computer hardware devices such as the α-channel allow the blending of pixel intensities with no speed penalties. When using saturation for parallel coordinate plots and the level of saturation corresponds with the degree of overplotting, this creates a kind of parallel coordinate density plot (Wegman and Luo 1997a,b).

12.3.11 Interactive and Dynamic Graphics for Categorical Data

Although categorical data are quite common in the real world, little research has been done for their analysis and visualization when compared to quantitative data. However, there exist useful interactive and dynamic graphics for categorical data (Ostermann and Nagel 1993; Theus and Wilhelm 1998). For example, brushing and

linking of categorical data represented via bar charts and pie charts can be as useful as for quantitative data (Hummel 1996). Modified bar charts where the same height is used for each category and the width is varied according to the number of counts are called spine plots (Hummel 1996). When interactively highlighting a category of interest, spine plots allow the analyst to visually compare the proportions in the different subcategories by looking at the heights of the highlighted areas. Examples of interactive graphics for categorical data such as spine plots and interactive mosaic plots (see Fig. 12.4) can be found in Hofmann (2000, 2003). Valero-Mora et al. (2003) discuss spreadplots (and their implementation in ViSta), a method for laying out and simultaneously controlling graphics for categorical data.

Blasius and Greenacre (1998) present a collection of papers dealing with the visualization of categorical data. Main topics include graphics for visualization, correspondence analysis, multidimensional scaling and biplots, and visualization and modeling. Several of these approaches benefit from interactive and dynamic graphics.

12.4 Graphical Software

In this section, we concentrate on three main streams of software for interactive and dynamic statistical graphics: Software developed by researchers affiliated with the University of Augsburg, in particular REGARD, MANET, and Mondrian; software developed by researchers affiliated with George Mason University (GMU), in particular ExplorN and CrystalVision; and software developed by researchers affiliated with Bell Labs, AT&T, and Iowa State University (ISU), in particular XGobi and GGobi. Wilhelm et al. (1996) contains an in depth review of software for interactive statistical graphics. Wilhelm et al. (1999) is one of the few publications where the different interactive graphical concepts provided by these three main streams (represented by MANET, ExplorN, and XGobi, respectively) are applied to the same data set and thus allow a direct comparison of their features and capabilities in visual clustering and classification.

12.4.1 REGARD, MANET, and Mondrian

In this section we present a series of software developments that was initiated in the late 1980's by John Haslett and Antony Unwin at Trinity College, Dublin, and later was continued by Antony Unwin and his collaborators at the Institut für Mathematik, University of Augsburg. Other main collaborators that contributed to the development of these software tools that should be mentioned here are Heike Hofmann, Martin Theus, Adalbert Wilhelm, and Graham Wills.

Some of the early developments are Diamond Fast (Unwin and Wills 1988) and Spider (Craig et al. 1989). Diamond Fast is a software package for the exploration

of multiple time series with interactive graphics. Spider is a software package for the exploration of spatially referenced data. One of its main features are moving statistics, an extension of brushing for spatial data (Craig et al. 1989). Spider also supports histograms, density estimates, scatterplot matrices, and linked brushing. It runs on Macintosh computers only.

REGARD (Unwin 1994; Unwin et al. 1990) is a software package that also provides high interaction graphics tools for spatial data. REGARD stands for "Radical Effective Graphical Analysis of Regional Data" and runs on Macintosh computers only. REGARD supports four types of layers of spatial data, i.e., points, regions, lines, and pictures. The central display in REGARD is the map window that is linked to statistical displays such as boxplots, scatterplots, and rotating plots. A map may be loaded as one picture in a picture layer or as several pictures in several layers, thus allowing to turn on or off different aspects of a map (such as state boundaries or a road network). Additional interactive features are interrogation, highlighting, resizing, and rescaling. Advanced features include zooming into submaps, animation across ordered variables, cross-layer linking, network analysis tools, and interactive query tools across all graphical displays.

MANET (Unwin et al. 1996) is a statistical graphics research program for EDA and written in C++. MANET stands for "Missings Are Now Equally Treated" and runs on Macintosh computers only. It is freely available from the following Web site: http://stats.math.uni-augsburg.de/Manet/.

MANET offers all standard one- and two-dimensional graphics for continuous data as well as for discrete data: dotplots, scatterplots, histograms, boxplots, bar charts. Some special graphics for discrete and spatial data are integrated: spine plots, mosaic plots and polygon plots. MANET grew out of a project to keep track of missing values in statistical graphics. In MANET all displays are fully linked and instantaneously updated. Displays are kept as simple as possible to not distract the user.

The standard use of linked views in MANET is to highlight clusters that are apparent in one dimension and to see these one-dimensional clusters in the light of other variables. By systematically subsetting the sample points, we can also detect two- and higher-dimensional clusters. Once a cluster has been detected, a classification rule can be set up by taking the boundary values of the cluster. In MANET those values can easily be obtained by interrogating the plot symbols.

One-dimensional views show the one-dimensional clusters directly. Two-dimensional clusters become visible by highlighting a subset in one variable and conditioning another plot on this subset. For three- and higher-dimensional clusters, we have to combine various subsets in different plots into one conditioning set and then we have to look at the remaining plots to check for clusters. The generation of such combined selections is not only possible in MANET but it is also very efficiently implemented through selection sequences.

In MANET, both dotplots and boxplots are drawn in a non-standard way. In dotplots the brightness of a point shows the frequency of its occurrence. This method, called tonal highlighting, is used to visualize overplotting of points. A bright color represents many points while a dark color represents just a few points.

There is no tonal highlighting for selected points in MANET. The layout of boxplots is changed so that a standard boxplot can be superimposed for selected points. The inner fifty percent box is drawn as a dark grey box. The outer regions, usually represented as whiskers, are drawn as light grey boxes.

A recent new development, Mondrian (Theus 2002, 2003; Theus and Urbanek 2009), is a data visualization system written in JAVA and therefore runs on any hardware platform. Mondrian is freely available from the following Web site: http://www.rosuda.org/Mondrian/.

The main emphasis of Mondrian is on visualization techniques for categorical and geographical data. All plots in Mondrian (see Fig. 12.4) are fully linked and offer various interrogations. Any case selected in one plot in Mondrian is highlighted in all other linked plots. Currently, implemented plots comprise mosaic plots, scatterplots, maps, bar charts, boxplots, histograms, and parallel coordinate plots. Mosaic plots in Mondrian are fully interactive. This includes not only linking, highlighting and interrogations, but also an interactive graphical modeling technique for loglinear models.

12.4.2 HyperVision, ExplorN, and CrystalVision

In this section we present a series of software developments that was initiated in the late 1980s by Daniel B. Carr (initially while at Battelle Pacific Northwest Laboratories) and Edward J. Wegman at GMU. Other main collaborators that contributed to the development of these software tools that should be mentioned here are Qiang Luo and Wesley L. Nicholson.

EXPLOR4 (Carr and Nicholson 1988) is a research tool, originally implemented on a VAX 11/780 and written in FORTRAN. Its main features are rotation, masking, scatterplots and scatterplot matrix, ray glyph plots, and stereo views.

HyperVision, presented in Bolorforoush and Wegman (1988), is a software product that has been implemented in PASCAL on an IBM RT under the AIX operating system as well as for MS-DOS machines. The latter implementation has a mouse-driven painting capability and can do real-time rotations of 3D scatterplots. Other displays are parallel coordinate plots, parallel coordinate density plots, relative slope plots, and color histograms. The main interactive features in HyperVision in addition to linked brushing are highlighting, zooming, and nonlinear rescaling of each axis.

ExplorN (Carr et al. 1997) is a more advanced software package than HyperVision and EXPLOR4, but with similar basic features. It runs on SGI workstations only, using either the GL or the OpenGL tools.

ExplorN supports scatterplot matrices, parallel coordinate plots, icon-enhanced three-dimensional stereoscopic plots, d-dimensional grand tours and partial grand tours (i.e., tours based on a subset of the variables with the remaining variables being held fixed), and saturation brushing all in a high interaction graphics package.

The ExplorN software is intended to demonstrate principles rather than to be an operational tool so that some refinements normally found in operational software are not there. These include history tracking, easy point identification, identification of mixture weights in the grand tour, relabeling of axes during and after a grand tour as well as simultaneous multiple window views.

Although ExplorN also supports conventional scatterplots and scatterplot matrices, one of its outstanding features are parallel coordinate displays and partial grand tours. Since it is easy to see pairwise relationships for adjacent variables in parallel coordinate plots, but less easy for nonadjacent variables, a complete parallel coordinate investigation would require running through all possible permutations. Instead of this, we recommend using the d-dimensional parallel coordinate grand tour that is implemented in ExplorN. An important interactive procedure for finding clusters using parallel coordinate plots is via the brush–tour.

CrystalVision is a recently developed successor of ExplorN, freely accessible at http://crystalvision.galaxy.gmu.edu/. Its main advantage over the older package is that it is available for PCs. Similar to ExplorN, CrystalVision's (see Fig. 12.3) main focus is on parallel coordinate plots, scatterplots, and grand tour animations. Examples of its use, e.g., its EDA techniques applied to scanner data provided by the U.S. Bureau of Labor Statistics (BLS), can be found in Wegman and Dorfman (2003).

12.4.3 Data Viewer, XGobi, and GGobi

In this section we present a series of software developments that was initiated in the mid 1980s by Andreas Buja, Deborah F. Swayne, and Dianne Cook at the University of Washington, Bellcore, AT&T Bell Labs, and ISU. Other main collaborators that contributed to the development of these software tools that should be mentioned here are Catherine Hurley, John A. McDonald, and Duncan Temple Lang.

The Data Viewer (Buja et al. 1988, 1986; Hurley 1988, 1989; Hurley and Buja 1990) is a software package originally developed on a Symbolics Lisp Machine that supports object-oriented programming. The Data Viewer is a system for the exploratory analysis of high-dimensional data sets that allows interactive labeling, identification, brushing, and linked windows. Additional features are viewport transformations such as expanding or shrinking of the data and shifting of the data. The Data Viewer supports several types of projections, including simple 3D rotations, correlation tour (Buja et al. 1988), and grand tour.

Many of the design and layout concepts of the Data Viewer as well as parts of its functionality provided the basic ideas for the follow-up XGobi (see Fig. 12.1), first described in Swayne et al. (1991) and Swayne and Cook (1992). Development on XGobi took place for about a decade; its almost final version is documented in Swayne et al. (1998). XGobi is implemented in the X Windows System, so it runs on any UNIX system, and it runs under Microsoft Windows or the Macintosh operating system if an X emulator is used. XGobi can be freely downloaded from http://www.research.att.com/areas/stat/xgobi/.

XGobi is a data visualization system with interactive and dynamic methods for the manipulation of views of data. It offers 2D displays of projections of points and lines in high-dimensional spaces, as well as parallel coordinate plots. Projection tools include dotplots and ASH of single variables, scatterplots of pairs of variables, 3D data rotations, and grand tours. Views of the data can be panned and zoomed. Points can be labeled and brushed with glyphs and colors. Lines can be edited and colored. Several XGobi processes can be run simultaneously and linked for labeling, brushing, and sharing of projections. Missing data are accommodated and their patterns can be examined; multiple imputations can be given to XGobi for rapid visual diagnostics (Swayne and Buja 1998). XGobi can be cloned, i.e., an identical new XGobi process with exactly the same data and all brushing information can be invoked.

Rotating plots are nowadays implemented in most statistical packages, but the implementation in XGobi goes beyond most of the others. In addition to the standard grand tour, XGobi supports the projection pursuit guided tour. More details on projection pursuit indices available in XGobi can be found in Cook et al. (1993) and Cook et al. (1995). Additional index functions that result in speed improvements of the calculations have been presented in Klinke and Cook (1997).

GGobi (Swayne et al. 2003; Cook and Swayne 2007) is a direct descendant of XGobi, but it has been thoroughly redesigned. GGobi (see Fig. 12.2) can be freely downloaded from http://www.ggobi.org/.

At first glance, GGobi looks quite unlike XGobi because GGobi uses a newer graphical toolkit called GTK+ (http://www.gtk.org), with a more contemporary look and feel and a larger set of user interface components. Through the use of GTK+, GGobi can be used directly on Microsoft Windows, without any emulator. In addition, GGobi can be used on any UNIX and Linux system.

In contrast to XGobi, the plot window in GGobi has been separated from the control panel. In XGobi, there is in general a single plot per process; to look at multiple views of the same data, we have to launch multiple XGobi processes. In contrast, a single GGobi session can support multiple plots of various types: scatterplots, parallel coordinate plots, scatterplot matrices, and time series plots have been implemented thus far. Other changes in GGobi's appearance and repertoire of tools (when compared to XGobi) include an interactive color lookup table manager, the ability to add variables "on the fly", and a new interface for view scaling (panning and zooming). At this point, some of the advanced grand tour and projection pursuit guided tour features from XGobi have not been fully reimplemented in GGobi (but hopefully will be available in the near future).

12.4.4 Other Graphical Software

While the previous sections summarize software that focuses on interactive and dynamic graphics, there exist several statistical languages that provide a tight integration of interactive graphics and numerical computations. Examples for such

languages are S/S-PLUS (Becker 1994; Becker et al. 1988a; Chambers 1997), R (Ihaka and Gentleman 1996), and XploRe (Härdle et al. 1995). Other examples of software that link interactive graphics, computation, and spread sheets, often through the Web, are the Data Representation System (DRS) by Inoue et al. (2002), DAVIS by Huh and Song (2002), KyPlot by Yoshioka (2002), and the XploRe Quantlet Client/Server (XQC/XQS) architecture (Kleinow and Lehmann 2002).

12.5 Interactive 3D Graphics

A natural extension of 2D interactive and dynamic graphics is the use of anaglyphs and stereoscopic displays on a computer screen and eventually the use of VR environments to obtain a 3D representation of statistical data and linked objects from geography or medicine.

12.5.1 Anaglyphs

A German teacher, Wilhelm Rollmann, initially described the effect of stereoscopic graphics drawn in red and green colors that are looked at with the naked eye (Rollmann 1853a), i.e., what is now called free-viewing stereoscopic images. Later the same year, Rollmann (1853b) describes the effect of looking at such colored pictures using filter glasses of corresponding complementary colors. As a reminder, red and green are complementary colors in the conventional color model whereas red and cyan are complementary colors in the RGB color model used for most computer monitors. Eventually, the work by Wilhelm Rollmann has been judged by Vuibert (1912) and Rösch (1954) as the birth of anaglyphs. The mathematics underlying anaglyphs and stereoscopic displays can be found in Hodges (1992) and Wegman and Carr (1993) for example.

Stereoscopic displays and anaglyphs have been used within statistics by Daniel B. Carr, Richard J. Littlefield, and Wesley L. Nicholson (Carr and Littlefield 1983; Carr et al. 1983; Carr and Nicholson 1985; Carr et al. 1986). In particular anaglyphs can be considered as an important means to represent three-dimensional pictures on flat surfaces. They have been used in a variety of sciences but they found only little use in statistics. One of the first implementations of red–green anaglyphs was the "real-time rotation of three-dimensional scatterplots" in the Mason Hypergraphics software package, described in Bolorforoush and Wegman (1988), page 125. Independently from the work on anaglyphs conducted in the U.S., interactive statistical anaglyph programs also were developed by Franz Hering, Jürgen Symanzik, and Stephan von der Weydt at the Fachbereich Statistik, University of Dortmund (Hering 1994; Hering and Symanzik 1992; Hering and von der Weydt 1989; Symanzik 1992, 1993a,b).

Wegman and DePriest (1986) is one of the rare sources in statistics where anaglyphs are used in the papers of Banchoff (1986), Carr et al. (1986), and Gabriel and Odoroff (1986). Moreover, Wegman and DePriest (1986) seems to be the first *statistical* reference where colored (red–green) anaglyphs have been published in print.

12.5.2 Virtual Reality

Many different definitions of the term VR can be found throughout the literature. Cruz-Neira (1993) summarizes several possible definitions of VR, including the following working definition for this chapter: "Virtual reality refers to immersive, interactive, multi-sensory, viewer-centered, three-dimensional computer generated environments and the combination of technologies required to build these environments." A brief chronology of events that influenced the development of VR can be found in Cruz-Neira (1993). A more detailed overview on VR can be found in Pimentel and Teixeira (1995) or Vince (1995) for example.

Carolina Cruz-Neira and her colleagues developed an ambitious visualization environment at the Electronic Visualization Lab (EVL) of the University of Illinois in Chicago, known simply as the CAVE (Cruz-Neira 1995; Cruz-Neira et al. 1993a,b, 1992; Roy et al. 1995). The abbreviation CAVE stands for *CAVE Audio Visual Experience Automatic Virtual Environment*. Carolina Cruz-Neira moved to ISU in 1995 where she was involved in the development of a second, larger CAVE-like environment known as the C2. The CAVE, C2, and several other of its successors belong to immersive projection technology (IPT) systems where the user is visually immersed within the virtual environment.

The use of ISU's C2 for statistical visualization is based on the framework of three-dimensional projections of p-dimensional data, using as a basis the methods developed and available in XGobi. The implementation of some of the basic XGobi features in the C2 resulted in VRGobi (see Fig. 12.5). The main difference between XGobi and VRGobi is that the XGobi user interface is rather like a desktop with pages of paper whereas VRGobi is more like having the whole room at the user's disposal for the data analysis.

VRGobi and the statistical visualization in the C2 have been extensively explored and documented in the literature (Cook 2001; Cook et al. 1998, 1997a; Nelson et al. 1998, 1999; Symanzik et al. 1996a, 1997). Main developers of VRGobi, over time, were Dianne Cook and Carolina Cruz-Neira, with major contributions by Brad Kohlmeyer, Uli Lechner, Nicholas Lewin, Laura Nelson, and Jürgen Symanzik. Additional information on VRGobi can be found at http://www.las.iastate.edu/newnews/Cook0219.html.

The initial implementation of VRGobi contains a three-dimensional grand tour. Taking arbitrary three-dimensional projections can expose features of the data not visible in one-dimensional or two-dimensional marginal plots.

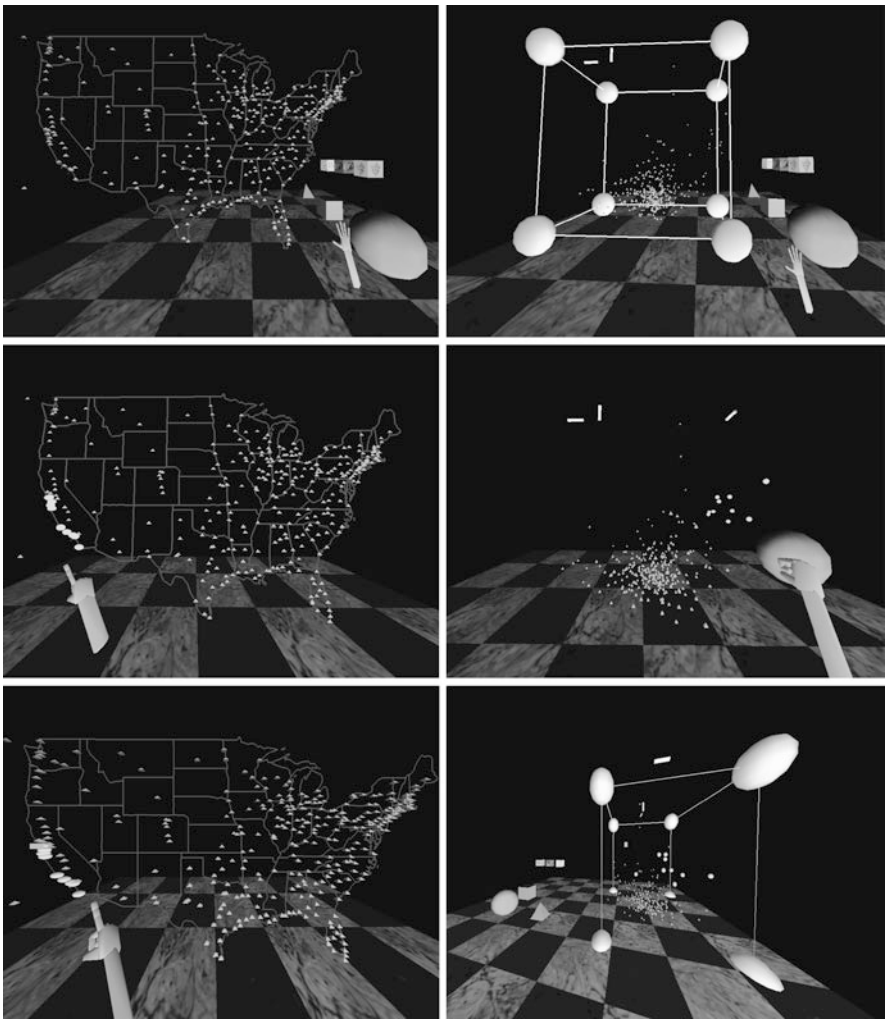


Fig. 12.5 Screenshots of the "Places" data in VRGobi, previously published in Symanzik et al. (1996a). A map view [*left*] and a three-dimensional point cloud displaying HousingCost, Climate, and Education are shown [*right*]. The control panel, glyph types, and the boundary box that delimits the plot area are visible [*top row*]. Cities with nice Climate and high HousingCost have been brushed and happen to fall into California [*middle row*]. Among the brushed points is one city (*San Francisco*) with an outstanding value for Education [*bottom row*]. When running VRGobi in the C2 (instead of producing screenshots from one of the control monitors), the rendered arm may be replaced by a human user who is possibly wearing a data glove

One of the most difficult developments for VRGobi was the user interface (and not the statistical display components). While it is relatively simple to create popup menus that allow to select colors and symbols for brushing in a desktop environment, designing an appealing and operational three-dimensional interface

for the C2 was a real challenge. Eventually, four main components make up VRGobi: the viewing box, the three-dimensional control panel, the variable spheres (similar to the variable circles used in XGobi), and possibly a map view.

A three-dimensional map view, if used, allows the user to explore data in its spatial context within VRGobi, similar to the ArcView/XGobi link (Cook et al. 1996, 1997b) for the desktop.

IPT environments are remarkably different from display devices that are commonly available for data analysis. They extend beyond the small gain of one more dimension of viewing space, to being a completely defined "real" world space. In VRGobi, the temptation is to grab the objects or climb a mountain in the map view and to step aside when a point approaches our face during the grand tour. The objects surround the viewer and it is possible to walk through the data.

In Nelson et al. (1998, 1999), experiments have been conducted on structure detection, visualization, and ease of interaction. Because only 15 human subjects participated in these experiments, it could not be expected that statistically significant results were obtained. However, at least these experiments showed that there was a clear trend that the test subjects performed considerably better on visualization tasks in the C2 than with XGobi on the workstation display. In contrast, interaction tasks such as brushing provided better results for the workstation. However, subjects with some limited VR experiences already performed considerably better on the interaction tasks in the C2 than subjects with no prior VR experience, suggesting that there is some learning needed to effectively use the VR hardware.

The high cost factor of the CAVE, C2, and similar IPT environments motivated the development of the PC-based MiniCAVE environment. The MiniCAVE is an immersive stereoscopic projection-based VR environment developed at GMU. It is oriented toward group interactions. As such, it is particularly suited to collaborative efforts in scientific visualization, data analysis, and VDM.

Initially researchers began with a 333 megahertz Pentium II machine running Windows NT. The SGI-based VR applications that make use of the OpenGL standard could be ported relatively easily to a PC environment. Using the Windows NT drivers, it was also possible to integrate the Crystal Eyes shutter glasses into the PC environment. The development of the MiniCAVE, now patented (Patent No. 6,448,965 "Voice-Controlled Immersive Virtual Reality System") to GMU, has been documented in Wegman et al. (1999) and Wegman and Symanzik (2002).

The one-wall MiniCAVE with speech recognition has been implemented on a dual 450 megahertz Pentium III machine at GMU. In addition, a polarized light LCD projector with both front and rear projection is used. Versions of ExplorN and CrystalVision have been ported to the MiniCAVE environment.

In addition to the work on VR-based data visualization conducted at ISU and GMU, independent work also has been conducted elsewhere, e.g., at Georgia Tech and the Delft Technical University, The Netherlands, resulting in the Virtual Data Visualizer (van Teylingen et al. 1997), and at the University of South Carolina, using the Virtual Reality Modeling Language (VRML) for VR applications on the World Wide Web (Rossini and West 1998). Böhlen et al. (2003) describe 3DVDM, a 3D

VDM system, that is aimed at the visual exploration of large data bases. More details are available at http://www.inf.unibz.it/dis/projects/3dvdm/.

Cook (2001) lists three fields, "environmental studies, especially data having a spatial component; shape statistics; and manufacturing quality control", that would benefit most from VR and other IPT environments. Certainly, recent experimental desktop links of VR and visualization software with spatial statistical applications such as the links between ViRGIS and RA_3DIO with XGobi (Schneider et al. 2000; Symanzik et al. 1998b) would benefit considerably when being conducted in an IPT environment. In addition to the fields in Cook (2001), we think that medical, genetic, and biological statistical data would also considerably benefit when being explored in an IPT environment.

12.6 Applications in Geography, Medicine, and Environmental Sciences

12.6.1 Geographic Brushing and Exploratory Spatial Data Analysis

Linking statistical plots with geography for analyzing spatially referenced data has been discussed widely in recent years. Monmonier (1988, 1989) describe a conceptual framework for geographical representations in statistical graphics and introduce the term geographic brushing in reference to interacting with the map view of geographically referenced data. But geographic brushing does not only mean pure interaction with the map. In addition, this term has a much broader meaning, e.g., finding neighboring points and spatial structure in a geographic setting.

In fact, the idea to apply interactive and dynamic graphics for EDA in a spatial (geographic) context resulted in the term exploratory spatial data analysis (ESDA). However, ESDA is more than just EDA applied to spatial data. In fact, specialized ESDA methods have been developed that take the special nature of spatial data explicitly into account. ESDA is discussed in more details in Anselin (1998), Anselin (1999), and Fotheringham et al. (2000), Chap. 4. Edsall (2003) provides examples for the use of dynamic and interactive parallel coordinate plots for the exploration of large spatial and spatio-temporal data bases.

Many software solutions have been developed that link geographic displays with interactive statistical software packages. In McDonald and Willis (1987), a grand tour is linked to an image to assess the clustering of landscape types in the band space of a LandSat image taken over Manaus, Brazil. In Carr et al. (1987) and Monmonier (1989), a scatterplot matrix is linked to a map view. In REGARD, map views are linked with histograms and scatterplots and, moreover, diagnostic plots for assessing spatial dependence are also available. Another exploratory system that links histograms and scatterplots with latitude and longitude (and depth) coordinates is discussed in MacDougall (1992). In Carr et al. (1992), (bivariate) ray-glyph maps

have been linked with scatterplots. Nagel (1994) discusses the interactive analysis of spatial data, mostly environmental and disease data, under ISP. Klein and Moreira (1994) report on an interface between the image program MTID and XGobi, used for the exploratory analysis of agricultural images. DiBiase et al. (1994) provide an overview on existing multivariate (statistical) displays for geographic data. Other developments are the cartographic data visualizer, `cdv` (Dykes 1996), where a variety of plots are linked with geography, the Space–Time–Attribute Creature/Movie, STAC/M (Openshaw and Perrée 1996), that searches for patterns in Geographic Information System (GIS) data bases under the control of a Genetic Algorithm, and an exploratory spatial analysis system in XLisp-Stat (Brunsdon and Charlton 1996).

In combination with the GIS ArcView, XGobi and XploRe also have been used to detect structure and abnormalities in geographically referenced data sets such as satellite imagery, forest health monitoring, and precipitation data (Cook et al. 1996, 1997b; Symanzik et al. 1998a, 2000a, 1996b) (see Fig. 12.1). In addition to the ArcView/XGobi/XploRe environment, there are several other examples where GIS's and (graphical) statistical packages have been linked. Williams et al. (1990) demonstrate how S and the GRASS GIS can be jointly used for archaeological site classification and analysis. Scott (1994) links STATA with ArcView. The spatial data analysis software SpaceStat has been linked with ARC/INFO (Anselin et al. 1993) and with ArcView (Anselin and Bao 1996, 1997). In Haining et al. (1996), the designing of a software system for interactive exploration of spatial data by linking to ARC/INFO has been discussed, and in Zhang and Griffith (1997), a spatial statistical analysis module implemented in ArcView using Avenue has been discussed. MathSoft (1996) describes the S+GISLink, a bidirectional link between ARC/INFO and S-PLUS, and Bao (1997) describes the S+Grassland link between S-PLUS and the Grassland GIS. Finally, a comparison of the operational issues of the SpaceStat/ArcView link and the S+Grassland link has been given in Bao and Anselin (1997).

12.6.2 Interactive Micromaps

Over the last decade, researchers have developed many improvements to make statistical graphics more accessible to the general public. These improvements include making statistical summaries more visual and providing more information at a time. Research in this area involved converting statistical tables into plots (Carr 1994; Carr and Nusser 1995), new ways of displaying geographically referenced data (Carr et al. 1992), and, in particular, the development of linked micromap (LM) plots (see Fig. 12.6), often simply called micromaps (Carr and Pierson 1996; Carr et al. 1998, 2000a). LM plots were first presented in a poster session sponsored by the ASA Section on Statistical Graphics at the 1996 Joint Statistical Meetings in Chicago ("Presentation of Data in Linked Attribute and Geographic Space" by Anthony R. Olsen, Daniel B. Carr, Jean-Yves P. Courbois, and Suzanne Pierson).

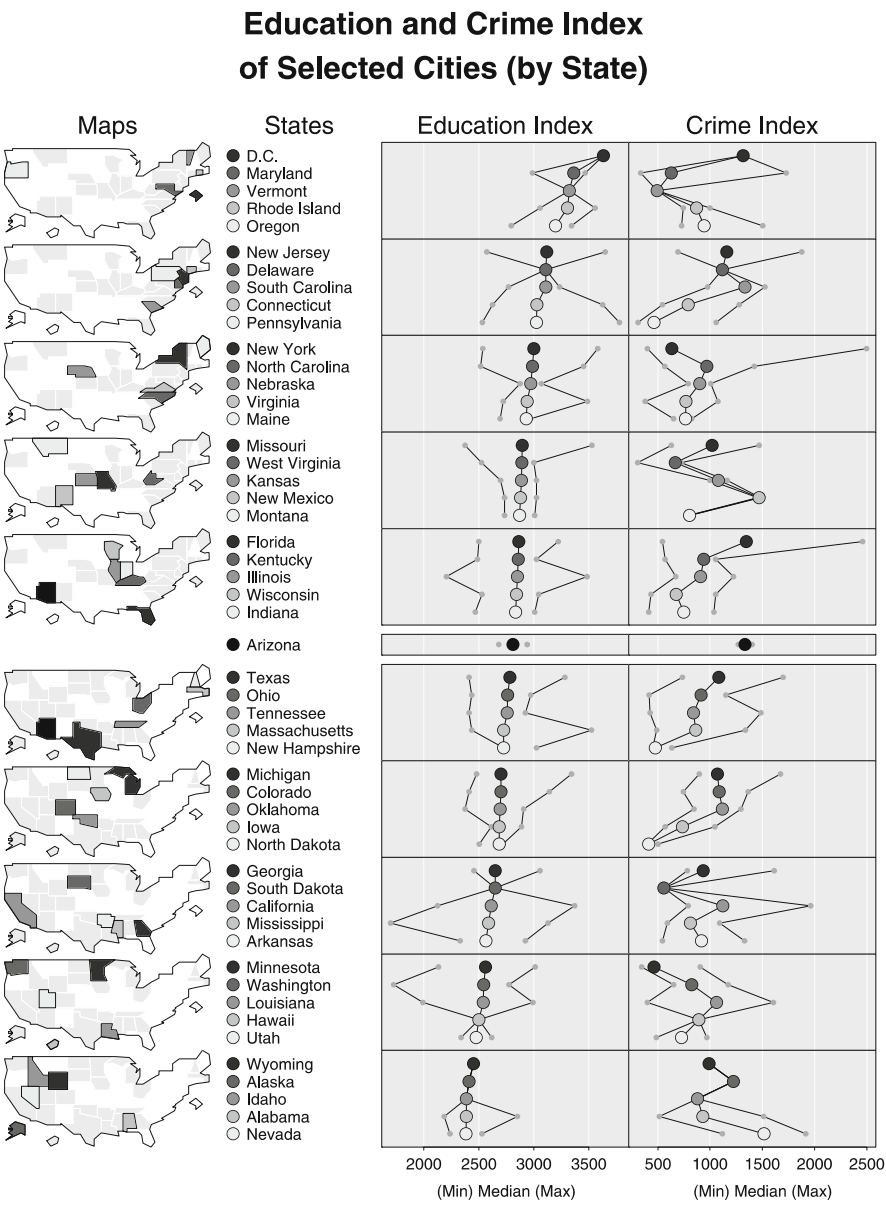


Fig. 12.6 Linked micromap plot of the "Places" data, adapted from Daniel B. Carr's sample S-PLUS code. The variables Education and Crime have been summarized at the state level for this figure. For each of the 50 states (plus Washington, D.C.), the minimum, median, and maximum of the Education and Crime indexes have been obtained for the cities that geographically belong to a state. It should be noted that for several of the states data exist for only one city. The map and statistical displays have been sorted with respect to decreasing median Education Index. The zig–zag curve of the related median Crime Index is an indicator of little correlation between these two variables Numerically, the ecological correlation between median Education Index and median Crime Index is almost equal to zero.

More details on the history of LM plots and their connection to other research can be found in these early references on micromaps. Recent references on LM plots (Carr 2001; Carr and Pickle 2010; Carr et al. 2000b) focus on their use for communicating summary data from health and environmental studies. Sample **R code**, data files, and resulting plots from Daniel B. Carr's micromap work can be accessed at http://mason.gmu.edu/~dcarr/Micromaps/.

Linked micromap plots provide a new statistical paradigm for the viewing of geographically referenced statistical summaries in the corresponding spatial context. The main idea behind LM plots is to focus the viewer's attention on the statistical information presented in a graphical display. Multiple small maps are used to provide the appropriate geographic reference for the statistical data.

Initially, LM plots were only static representations on paper. The next stage of LM plots was aimed at interactive displays on the Web. Eventually, generalized maps for all states in the U.S. and several counties were automatically created for use on the U.S. Environmental Protection Agency (EPA) Cumulative Exposure Project (CEP) Web site (Symanzik et al. 2000b). Most current applications of interactive LM plots on the Web make use of these generalized maps.

The idea of using micromaps on the Web was first considered for the EPA CEP Web site (formerly accessible at http://www.epa.gov/CumulativeExposure/). Initially, the EPA wanted to provide fast and convenient Web-based access to its hazardous air pollutant (HAP) data for 1990. Unfortunately, no part of the interactive CEP Web site was ever published due to concerns that the 1990 data was outdated at the intended release date in 1998. Only a static version of the CEP Web site without tables and micromaps was accessible. More details on the work related to the planned interactive CEP Web site can be found in Symanzik et al. (1999b,c, 2000b).

The U.S. Department of Agriculture (USDA)—National Agricultural Statistics Service (NASS) Research and Development Division released a Web site (http://www.nass.usda.gov/research/sumpant.htm) in September 1999 that uses interactive micromaps to display data from the 1997 Census of Agriculture. While the end user who accesses this Web site gets the impression of fully interactive graphics, this is not the case. The 10 micromaps (5 crops $\times$ 2 arrangements) plus one overview micromap were precalculated in S-PLUS and were stored as jpg images. It is not possible to create any new micromap display "on the fly" on this Web site.

The National Cancer Institute (NCI) released a Web site in April 2003 that provides interactive access to its cancer data via micromaps. This Web site is Java-based and creates micromaps "on the fly". Wang et al. (2002) and Carr et al. (2002) provide more details on the design of the NCI Web site that is accessible at http://www.statecancerprofiles.cancer.gov/micromaps.

nViZn (read envision) is a JAVA-based software development kit (SDK), formerly developed and distributed by SPSS. It is the follow-up to the Graphics Production Library (GPL), described in Carr et al. (1996), developed within the BLS. nViZn (Wilkinson et al. 2000) is based on a formal grammar for the specification of statistical graphics (Wilkinson 1999), see also Chap. II.13. In addition to capabilities already present in the original GPL, nViZn has many additional features. Most useful for the display of data in a geographic context are the capabilities that

enable a programmer to create interactive tables and linked micromaps in nViZn. Experiences with nViZn, its advantages and current problems, and its capabilities for the display of Federal data via LM plots are described in more detail in Jones and Symanzik (2001), Symanzik and Jones (2001), and Symanzik et al. (2002a).

Micromap implementations that allow the user to create new LM plots "on the fly" often provide features to switch from one geographic region or subregion to another, choose among several variables, resort the data increasingly or decreasingly according to different statistics (such as mean, median, minimum, or maximum of the data values in the underlying geographic region), and display different graphical summaries of the data (e.g., dotplots, boxplots, confidence intervals, or even time series). So, in an interactive environment, a user might want to create a LM plot of Education and Arts (sorted by increasing maximum Education) after having studied the LM plot in Fig. 12.6—and then immediately resort the display by decreasing maximum Arts.

12.6.3 Conditioned Choropleth Maps

Conditioned choropleth maps (CCmaps), described in Carr et al. (2002); Carr and Pickle (2010); Carr et al. (2000b), focus on spatial displays that involve one dependent variable and two independent variables. CCmaps promote interactive hypothesis generation, common for epidemiological and environmental applications. In fact, applications from the National Center for Health Statistics (NCHS) and the EPA motivated the development of CCmaps. CCmaps are written in Java and can be freely obtained from http://mason.gmu.edu/~dcarr/CCmaps.

The main interactive component of CCmaps are partitioning sliders that allow to dynamically partition the study units into a 3×3 layout of maps. The sliders allow a user to create, examine, and contrast subsets for the purpose of generating hypotheses about patterns in spatially referenced data. For example, in a medical application one of the sliders might control the age intervals and the second slider might control the years of active smoking in a study on cancer mortality rates across the U.S. The resulting 9 maps will allow an analyst to develop hypotheses on spatial patterns within panels or among panels. Additional features of this CCmaps implementation are dynamic quantile–quantile (QQ) plots and pan and zoom widgets to allow closer inspection of data at the U.S. county level.

12.7 Outlook

12.7.1 Limitations of Graphics

Wegman (1995) discusses aspects of data set size, computational feasibility, and in particular limits of visualization for "large" (about 10^8 bytes) and "huge" (about 10^{10} bytes) data sets, where "large" and "huge" are terms introduced in Huber's

taxonomy of large data sets (Huber 1992, 1994). As pointed out in Wegman (1995), even in the most wildly optimistic scenario, i.e., an angular resolution of 4.38 minutes of arc as suggested in Maar (1982), immersion, and a 4:5 aspect-ratio, the human eye would only be able to distinguish $17,284 \times 13,828 = 2.39 \times 10^8$ pixels. Using single-pixel coding, it seems to be impossible to visualize "large" to "huge" data sets.

Huber (1994) initially suggests to prepare "medium" (about 10^6 bytes) derived data sets that are easier to visualize and grasp as a whole and can still be worked with established techniques of high interaction graphics. Unfortunately, as Wegman (1995) further describes, common ways of parsing data sets down, e.g., clustering, discriminant analysis, and principal components, are computationally complex (often of a magnitude of $O(n^{3/2})$ or even $O(n^2)$) and therefore are not valid alternatives. It seems that simple random thinning is the only methodology of choice, but this may have the side effect of missing some of the tail structure an analyst may actually be looking for.

Possible solutions are the use of 3D VR techniques that may display up to 10^{10} voxels (Wegman 1995), further advances in selection sequences, or new strategies to increase visual scalability i.e., the capability of visualization tools to effectively display large data sets (Eick and Karr 2002). However, the conclusion in Wegman (1995) that "visualization of data sets say of size 10^6 or more is clearly a wide open field" is still valid today.

12.7.2 Future Developments

Historically, one of the problems with interactive and dynamic statistical graphics was to publish visible results. The ASA Statistical Graphics Section Video Lending Library was one attempt to capture at least some snapshots of software and applications of interactive and dynamic statistical graphics and preserve them for the future. Publishing a sequence of screenshots in a written paper clearly has not the same effect as watching the full interaction and being able to manipulate the graphics.

However, due to the recent move of books being published with accompanying CDs or DVDs and many conference proceedings being published on CD, it is now possible to immediately publish a movie accompanying a written paper or integrate interactive graphics within a paper. Examples are Wojciechowski and Scott (2000) and Symanzik et al. (2002b) where the papers are accompanied by several movie segments. It is to be expected that more and more future publications on interactive and dynamic graphics will be accompanied by an interactive application or by movies.

To be useful for the future, interactive and dynamic graphics have to adapt to challenges posed by "large" and "huge" (in terms of Huber's taxonomy) data sets as outlined in Sect. 12.7.1. Examples for such data sets are data from earth or planetary observation systems, real-time S-PLUS data (such as from surveillance systems),

or any other type of massive data streams. Wegman (2000) provides a futuristic vision, indicating that major advancements can be expected in DM, visualization, and quantization methods.

For smaller data sets (from "tiny" to "medium" in terms of Huber's taxonomy) we can expect to see progress in new user paradigms that will allow to interact with the data through voice or gestures as well as multiple users to manipulate the visible view simultaneously. It can also be expected that more graphical software will make use of the Web and mobile data transmission and reception techniques. The same software may therefore be available for a variety of hardware platforms with screens as small as a clock or cell phone or as big as a 3D IMAX theatre.

As pointed out in Carr et al. (2002), "there are many barriers to acceptance of new methodology by federal agencies." This can be easily extended towards other users of newly developed interactive and dynamic graphical software. Clearly, just promoting a new idea or graphical software product is not enough in many cases. It is likely that more usability tests of new graphical software products as well as comparative reviews of old and new tools will be conducted in the future.

References

Andrews, D.F.: Plots of high-dimensional Data. Biometrics **28**, 125–136 (1972)

Anselin, L.: Exploratory spatial data analysis in a geocomputational environment. In: Longley, P.A., Brooks, S.M., McDonnell, R., Macmillan, B. (eds.) Geocomputation – A Primer, pp. 77–94. Wiley, Chichester (1998)

Anselin, L.: Interactive techniques and exploratory spatial data analysis. In: Longley, P.A., Goodchild, M.F., Maguire, D.J., Rhind, D.W. (eds.) Geographical Information Systems, Volume 1: Priniciples and Technical Issues (Second Edition), pp. 253–266. Wiley, New York, NY (1999)

Anselin, L., Bao, S.: Exploratory Spatial Data Analysis Linking SpaceStat and ArcView. Technical Report 9618, West Virginia University, Morgantown, WV (1996)

Anselin, L., Bao, S.: Exploratory spatial data analysis linking spacestat and arcview. In: Fischer, M.M., Getis, A. (eds.) Recent Developments in Spatial Analysis, pp. 35–59. Springer, Berlin (1997)

Anselin, L., Dodson, R.F., Hudak, S.: Linking GIS and spatial data analysis in practice. Geogr. Syst. **1**(1), 3–23 (1993)

Asimov, D.: The grand tour: A tool for viewing multidimensional data. SIAM J. Sci. Stat. Comput. **6**(1), 128–143 (1985)

Banchoff, T.F.: Visualizing two-dimensional phenomena in four-dimensional space: A computer graphics approach. In Wegman, E.J., DePriest, D.J. (eds.) Statistical Image Processing and Graphics, pp. 187–202. Marcel Dekker, New York, NY (1986)

Bao, S.: User's Reference for the S+Grassland Link. Mathsoft, Incorporation, Seattle, WA (1997)

Bao, S., Anselin, L.: Linking Spatial Statistics with GIS: Operational Issues in the SpaceStat-ArcView Link and the S+Grassland Link. In: 1997 Proceedings of the Section on Statistical Graphics, pp. 61–66, American Statistical Association, Alexandria, VA (1997)

Becker, R.A.: A brief history of S. In: Dirschedl, P., Ostermann, R. (eds.) Computational Statistics, pp. 81–110. Physica, Heidelberg (1994)

Becker, R.A., Chambers, J.M., Wilks, A.R.: The New S Language – A Programming Environment for Data Analysis and Graphics. Wadsworth and Brooks/Cole, Pacific Grove, CA (1988a)

Becker, R.A., Cleveland, W.S.: Brushing scatterplots. In: Cleveland, W.S., McGill, M.E. (eds.) Dynamic Graphics for Statistics, pp. 201–224. Wadsworth & Brooks/Cole, Belmont, CA (1988)
Becker, R.A., Cleveland, W.S., Weil, G.: The Use of Brushing and Rotation for Data Analysis. In: Cleveland, W.S., McGill, M.E. (eds.) Dynamic Graphics for Statistics, pp. 247–275. Wadsworth & Brooks/Cole, Belmont, CA (1988b)
Blasius, J., Greenacre, M. (ed.): Visualization of Categorical Data. Academic Press, San Diego, CA (1998)
Böhlen, M., Bukauskas, L., Eriksen, P.S., Lauritzen, S.L., Mazeika, A., Musaeus, P., Mylov, P.: 3D Visual Data Mining – Goals and Experiences. Comput. Stat. Data Anal: Special Issue on Data Visualization **43**(4), 445–469 (2003)
Bolorforoush, M., Wegman, E.J.: On some graphical representations of multivariate data. In: Wegman, E.J., Gantz, D.T., Miller, J.J. (eds.) Proceedings of the 20th Symposium on the Interface between Computing Science and Statistics, pp. 121–126. American Statistical Association, Alexandria, VA (1988)
Boyer, R., Savageau, D.: Places Rated Almanac. Rand McNally, Chicago, IL (1981)
Brunsdon, C., Charlton, M.: Developing an exploratory spatial analysis system in XLisp-stat. In: Parker, D. (eds.) Innovations in GIS 3, pp. 135–145. Taylor & Francis, London, UK (1996)
Buja, A., Asimov, D.: Grand tour methods: An outline. Comput. Sci. Stat. **17**, 63–67 (1986a)
Buja, A., Asimov, D.: Grand tour methods: An outline. In: Allen, D.M. (eds.) Proceedings of the 17th Symposium on the Interface between Computer Science and Statistics, Lexington, KY, pp. 63–67. Elsevier, Amsterdam (1986b)
Buja, A., Asimov, D., Hurley, C., McDonald, J.A.: Elements of a viewing pipeline for data analysis. In: Cleveland, W.S., McGill, M.E. (eds.) Dynamic Graphics for Statistics, pp. 277–308. Wadsworth & Brooks/Cole, Belmont, CA (1988)
Buja, A., Cook, D., Swayne, D.F.: Interactive high-dimensional data visualization. J. Comput. Graph. Stat. **5**(1), 78–99 (1996)
Buja, A., Hurley, C., McDonald, J.A.: A data viewer for multivariate data. In: Boardman, T.J., Stefanski, I.M. (eds.) Proceedings of the 18th Symposium on the Interface between Computer Science and Statistics, Fort Collins, CO, pp. 171–174. American Statistical Association, Washington, DC (1986)
Buja, A., McDonald, J.A., Michalak, J., Stuetzle, W.: Interactive data visualization using focusing and linking. In: Nielson, G.M., Rosenblum, L.J. (eds.) Proceedings of Visualization '91, Los Alamitos, CA, pp. 156–163. IEEE Computer Society Press (1991)
Buja, A., Tukey, P.A. (ed.): Computing and Graphics in Statistics. Springer, New York, NY (1991)
Carr, D.B.: Looking at large data sets using binned data plots. In: Buja, A., Tukey, P.A. (eds.) Computing and Graphics in Statistics, pp. 7–39. Springer, New York, NY (1991)
Carr, D.B.: Converting Tables to Plots. Technical Report 101, Center for Computational Statistics, George Mason University, Fairfax, VA (1994)
Carr, D.B.: Designing Linked Micromap Plots for States with Many Counties. Stat. Med. **20**(9–10), 1331–1339 (2001)
Carr, D.B., Chen, J., Bell, B.S., Pickle, L., Zhang, Y.: Interactive Linked Micromap Plots and Dynamically Conditioned Choropleth Maps. In dg.o2002 Proceedings. Digital Government Research Center (DGRC). http://www.dgrc.org/conferences/2002_proceedings.jsp. (2002)
Carr, D.B., Littlefield, R.J.: Color anaglyph stereo scatterplots – construction details. In: Gentle, J.E. (eds.) Proceedings of the 15th Symposium on the Interface between Computer Science and Statistics, pp. 295–299, North-Holland Publishing Company, New York, NY (1983)
Carr, D.B., Littlefield, R.J., Nicholson, W.L.: Color anaglyph stereo scatterplots – construction and application. In: 1983 Proceedings of the Section on Statistical Computing, pp. 255–257, American Statistical Association, Alexandria, VA (1983)
Carr, D.B., Littlefield, R.J., Nicholson, W.L., Littlefield, J.S.: Scatterplot matrix techniques for large N. J. Am. Stat. Assoc. **82**(398), 424–436 (1987)
Carr, D.B., Nicholson, W.L.: Evaluation of Graphical Techniques for Data in Dimensions 3 to 5: Scatterplot Matrix, Glyph, and Stereo Examples. In: 1985 Proceedings of the Section on Statistical Computing, pp. 229–235, American Statistical Association, Alexandria, VA (1985)

Carr, D.B., Nicholson, W.L.: EXPLOR4: A program for exploring four-dimensional data using stereo-ray glyphs, dimensional constraints, rotation, and masking. In: Cleveland, W.S., McGill, M.E. (eds.) Dynamic Graphics for Statistics, pp. 309–329. Wadsworth & Brooks/Cole, Belmont, CA (1988)

Carr, D.B., Nicholson, W.L., Littlefield, R.J., Hall, D.L.: Interactive color display methods for multivariate data. In: Wegman, E.J., DePriest, D.J. (eds.) Statistical Image Processing and Graphics, pp. 215–250. Marcel Dekker, New York, NY (1986)

Carr, D.B., Nusser, S.M.: Converting tables to plots: A challenge from iowa state. Stat. Comput. Stat. Graph. Newsletter **6**(3), 11–18 (1995)

Carr, D.B., Olsen, A.R., Courbois, J.P., Pierson, S.M., Carr, D.A.: Linked micromap plots: Named and described. Stat. Comput. Stat. Graph. Newsletter **9**(1), 24–32 (1998)

Carr, D.B., Olsen, A.R., Pierson, S.M., Courbois, J.P.: Using linked micromap plots to characterize omernik ecoregions. Data Min. Knowl. Discov. **4**(1), 43–67 (2000a)

Carr, D.B., Olsen, A.R., White, D.: Hexagon Mosaic Maps for Displays of Univariate and Bivariate Geographical Data. Cartography Geogr. Inform. Syst. **19**(4), 228–236, 271 (1992)

Carr, D.B., Pickle, L.W.: Visualizing Data Patterns with Micromaps. Chapman & Hall/CRC, Boca Raton (2010)

Carr, D.B., Pierson, S.M.: Emphasizing statistical summaries and showing spatial context with micromaps. Stat. Comput. Stat. Graph. Newsletter **7**(3), 16–23 (1996)

Carr, D.B., Valliant, R., Rope, D.J.: Plot interpretation and information webs: A time-series example from the bureau of labor statistics. Stat. Comput. Stat. Graph. Newsletter **7**(2), 19–26 (1996)

Carr, D.B., Wallin, J.F., Carr, D.A.: Two new templates for epidemiology applications: Linked micromap plots and conditioned choropleth maps. Stat. Med. **19**(17–18), 2521–2538 (2000b)

Carr, D.B., Wegman, E.J., Luo, Q.: ExplorN: Design Considerations Past and Present. Technical Report 137, Center for Computational Statistics, George Mason University, Fairfax, VA (1997)

Chambers, J.M.: Evolution of the S language. Comput. Sci. Stat. **28**, 331–337 (1997)

Chambers, J.M., Cleveland, W.S., Kleiner, B., Tukey, P.A.: Graphical Methods for Data Analysis. Wadsworth & Brooks/Cole, Pacific Grove, CA (1983)

Chang, J.: Real-Time Rotation. ASA Statistical Graphics Video Library http://stat-graphics.org/movies (1970)

Cleveland, W.S.: The Elements of Graphing Data. Wadsworth, Monterey, CA (1985)

Cleveland, W.S., McGill, M.E. (ed.): Dynamic Graphics for Statistics. Wadsworth & Brooks/Cole, Belmont, CA (1988)

Cook, D.: Calibrate your eyes to recognize high-dimensional shapes from their low-dimensional projections. J. Stat. Software **2**(6) (1997); http://www.jstatsoft.org/v02/i06/.

Cook, D.: Virtual reality: Real ponderings. Chance **14**(1), 47–51 (2001)

Cook, D., Buja, A.: Manual controls for high-dimensional data projections. J. Comput. Graph. Stat. **6**(4), 464–480 (1997)

Cook, D., Buja, A., Cabrera, J.: Projection pursuit indexes based on orthonormal function expansions. J. Comput. Graph. Stat. **2**(3), 225–250 (1993)

Cook, D., Buja, A., Cabrera, J., Hurley, C.: Grand tour and projection pursuit. J. Comput. Graph. Stat. **4**(3), 155–172 (1995)

Cook, D., Cruz-Neira, C., Kohlmeyer, B.D., Lechner, U., Lewin, N., Nelson, L., Olsen, A.R., Pierson, S.M., Symanzik, J.: Exploring environmental data in a highly immersive virtual reality environment. Environ. Monit. Assess. **51**(1/2), 441–450 (1998)

Cook, D., Cruz-Neira, C., Lechner, U., Nelson, L., Olsen, A.R., Pierson, S.M., Symanzik, J.: Using Dynamic Statistical Graphics in a Highly Immersive Virtual Reality Environment to Understand Multivariate (Spatial) Data. In: Bulletin of the International Statistical Institute, 51st Session Istanbul 1997, Proceedings Book 2, pp. 31–34 (1997a)

Cook, D., Majure, J.J., Symanzik, J., Cressie, N.: Dynamic graphics in a GIS: Exploring and analyzing multivariate spatial data using linked software. Comput. Stat.: Special Issue on Computeraided Analysis of Spatial Data **11**(4), 467–480 (1996)

Cook, D., Swayne, D.F.: Interactive and Dynamic graphics for Data Analysis – With R and GGobi. Springer, New York (2007)
Cook, D., Symanzik, J., Majure, J.J., Cressie, N.: Dynamic graphics in a GIS: More examples using linked software. Comput. Geosci.: Special Issue on Exploratory Cartographic Visualization **23**(4), 371–385 (1997b).
Cox, K.C., Eick, S.G., Wills, G.J., Brachman, R.J.: Visual data mining: Recognizing telephone calling fraud. Data Min. Knowl. Discov. **1**, 225–231 (1997)
Craig, P., Haslett, J., Unwin, A., Wills, G.: Moving Statistics – An Extension of "Brushing" for Spatial Data. In: Berk, K., Malone, L. (eds.) Proceedings of the 21st Symposium on the Interface between Computing Science and Statistics, pp. 170–174. American Statistical Association, Alexandria, VA (1989)
Crawford, S.L., Fall, T.C.: Projection Pursuit Techniques for Visualizing High-Dimensional Data Sets. In: Nielson, G.M., Shrivers, B., Rosenblum, L.J. (eds.) Proceedings of Visualization in Scientific Computing, pp. 94–108. IEEE Computer Society Press, Los Alamitos, CA (1990)
Cruz-Neira, C.: Virtual Reality Overview. SIGGRAPH '93 Course Notes #23. pp. 1–18 (1993)
Cruz-Neira, C.: Projection-based Virtual Reality: The CAVE and its Applications to Computational Science. PhD thesis, University of Illinois at Chicago (1995)
Cruz-Neira, C., Leigh, J., Papka, M., Barnes, C., Cohen, S.M., Das, S., Engelmann, R., Hudson, R., Roy, T., Siegel, L., Vasilakis, C., DeFanti, T., and Sandin, D.J.: Scientists in Wonderland: A Report on Visualization Applications in the CAVE Virtual Reality Environment. In: IEEE 1993 Symposium on Research Frontiers in Virtual Reality, pp. 59–66 (1993a)
Cruz-Neira, C., Sandin, D.J., DeFanti, T.A.: Surround-Screen Projection-Based Virtual Reality: The Design and Implementation of the CAVE. In: ACM SIGGRAPH '93 Proceedings, pp. 135–142, Anaheim, CA (1993b)
Cruz-Neira, C., Sandin, D.J., DeFanti, T.A., Kenyon, R.V., Hart, J.C.: The CAVE: Audiovisual experience automatic virtual environment. Comm. ACM **35**(6), 64–72 (1992)
DiBiase, D., Reeves, C., MacEachren, A.M., von Wyss, M., Krygier, J.B., Sloan, J.L., Detweiler, M.C.: Multivariate Display of Geographic Data: Applications in Earth System Science. In: MacEachren, A.M., Taylor, D. R.F. (eds.) Visualization in Modern Cartography, pp. 287–312. Pergamon (Elsevier), Oxford, UK (1994)
du Toit, S. H.C., Steyn, A. G.W., Stumpf, R.H.: Graphical Exploratory Data Analysis. Springer, New York, NY (1986)
Dykes, J.A.: Dynamic Maps for Spatial Science: A Unified Approach to Cartographic Visualization. In: Parker, D. (eds.) Innovations in GIS 3, pp. 177–187. Taylor & Francis, London, UK (1996)
Edsall, R.M.: The Parallel Coordinate Plot in Action: Design and Use for Geographic Visualization. Comput. Stat. Data Anal.: Special Issue on Data Visualization **43**(4), 605–619 (2003)
Eick, S.G., Karr, A.F.: Visual scalability. J. Comput. Graph. Stat. **11**(1), 22–43 (2002)
Fisherkeller, M.A., Friedman, J.H., Tukey, J.W.: PRIM-9: An Interactive Multidimensional Data Display and Analysis System. ASA Statistical Graphics Video Library http://stat-graphics.org/movies (1974a)
Fisherkeller, M.A., Friedman, J.H., Tukey, J.W.: PRIM-9: An Interactive Multidimensional Data Display and Analysis System. Technical Report SLAC-PUB-1408, Stanford Linear Accelerator Center, Stanford, CA (1974b)
Foley, J.D., van Dam, A., Feiner, S.K., Hughes, J.F.: Computer Graphics – Principles and Practice (2nd Edn.), Addison-Wesley, Reading, MA (1990)
Fotheringham, A.S., Brunsdon, C., Charlton, M. (ed.): Quantitative Geography: Perspectives on Spatial Data Analysis. Sage, London (2000)
Friedman, J.H.: Exploratory projection pursuit. J. Am. Stat. Assoc. **82**, 249–266 (1987)
Friedman, J.H.: Data mining and statistics: What's the connection? Comput. Sci. Stat. **29**(1), 3–9 (1998)
Friedman, J.H., Tukey, J.W.: A projection pursuit algorithm for exploratory data analysis. IEEE Trans. Comput. C **23**, 881–889 (1974)

Furnas, G.W.: Dimensionality Constraints on Projection and Section Views of High Dimensional Loci. In: Wegman, E.J., Gantz, D.T., Miller, J.J. (eds.) Proceedings of the 20th Symposium on the Interface between Computing Science and Statistics, pp. 99–107. American Statistical Association, Alexandria, VA (1988)

Furnas, G.W., Buja, A.: Prosection Views: Dimensional Inference Through Sections and Projections (with Discussion). J. Comput. Graph. Stat. **3**(4), 323–385 (1994)

Gabriel, K.R., Odoroff, C.L.: Illustrations of Model Diagnosis by Means of Three-Dimensional Biplots. In: Wegman, E.J., DePriest, D.J. (eds.) Statistical Image Processing and Graphics, pp. 257–274. Marcel Dekker, New York, NY (1986)

Haining, R., Ma, J., Wise, S.: Design of a Software System for Interactive Spatial Statistical Analysis Linked to a GIS. Comput. Stat.: Special Issue on Computeraided Analysis of Spatial Data **11**(4), 449–466 (1996)

Hall, P.: Polynomial projection pursuit. Ann. Stat. **17**, 589–605 (1989)

Härdle, W., Klinke, S., Turlach, B.A.: XploRe: An Interactive Statistical Computing Environment. Springer, New York, NY (1995)

Hering, F.: Anaglyphs in Statistics – A Tool for Interactive Multivariate Data Analysis. In: Faulbaum, F. (eds.) SoftStat '93 – Advances in Statistical Software 4, pp. 277–283, Stuttgart, Jena, Gustav Fischer, New York (1994)

Hering, F., Symanzik, J.: Anaglyphen 3D – Ein Programm zur interaktiven Anaglyphendarstellung. Forschungsbericht 92/1, Fachbereich Statistik, Universität Dortmund, German (1992)

Hering, F., von der Weydt, S.: Interaktive Anaglyphendarstellungen als Hilfsmittel zur Analyse mehrdimensionaler Daten. Forschungsbericht 89/7, Fachbereich Statistik, Universität Dortmund, German (1989)

Hodges, L.F.: Tutorial: Time-multiplexed stereoscopic computer graphics. IEEE Comput. Graph. Appl. **12**(2), 20–30 (1992)

Hofmann, H.: Exploring categorical data: Interactive mosaic plots. Metrika **51**(1), 11–26 (2000)

Hofmann, H.: Constructing and reading mosaicplots. Comput. Stat. Data Anal.: Special Issue on Data Visualization **43**(4), 565–580 (2003)

Hofmann, H., Theus, M.: Selection Sequences in MANET. Comput. Stat.: Special Issue on Strategies for Data Analysis **13**(1), 77–87 (1998)

Huber, P.J.: Projection pursuit (with discussion). Ann. Stat. **13**, 435–525 (1985)

Huber, P.J.: Issues in Computational Data Analysis. In: Dodge, Y., Whittaker, J. (eds.) COMPSTAT 1992: Proceedings in Computational Statistics, vol. 2, pp. 3–13. Physica, Heidelberg (1992)

Huber, P.J.: Huge Data Sets. In: Dutter, R., Grossmann, W. (eds.) COMPSTAT 1994: Proceedings in Computational Statistics, pp. 3–13, Physica, Heidelberg (1994)

Huh, M.Y., Song, K.: DAVIS: A java-based data visualization system. Comput. Stat. **17**(3), 411–423 (2002)

Hummel, J.: Linked bar charts: Analysing categorical data graphically. Comput. Stat. **11**, 23–33 (1996)

Hurley, C.: A Demonstration of the Data Viewer. In Wegman, E.J., Gantz, D.T., Miller, J.J., (eds.) Proceedings of the 20th Symposium on the Interface between Computing Science and Statistics, pp. 108–114. American Statistical Association, Alexandria, VA (1988)

Hurley, C.: The Data Viewer: A Program for Graphical Data Analysis. PhD thesis, Statistics Department, University of Washington, Seattle (1989)

Hurley, C., Buja, A.: Analyzing high-dimensional data with motion graphics. SIAM J. Sci. Stat. Comput. **11**(6), 1193–1211 (1990)

Ihaka, R., Gentleman, R.: R: A language for data analysis and graphics. J. Comput. Graph. Stat. **5**(3), 299–314 (1996)

Inoue, T., Asahi, Y., Yadohisa, H., Yamamoto, Y.: A statistical data representation system on the web. Comput. Stat. **17**(3), 367–378 (2002)

Inselberg, A.: The plane with parallel coordinates. Vis. Comput. **1**, 69–91 (1985)

Inselberg, A.: Visual data mining with parallel coordinates. Comput. Stat.: Special Issue on Strategies for Data Analysis **13**(1), 47–63 (1998)

Inselberg, A.: Parallel Coordinates: Visual Multidimensional Geometry and its Applications. Springer, Dordrecht, Heidelberg, London, New York (2009)
Jones, L., Symanzik, J.: Statistical visualization of environmental data on the web using nViZn. Comput. Sci. Stat. **33**, (CD) (2001)
Jones, M.C., Sibson, R.: What is projection pursuit? (with discussion). J. Roy. Stat. Soc. A **150**, 1–36 (1987)
Klein, R., Moreira, R.I.: Exploratory Analysis of Agricultural Images via Dynamic Graphics. Technical Report 9/94, Laboratório Nacional de Computação Científica, Rio de Janeiro, Brazil (1994)
Kleinow, T., Lehmann, H.: Client/Server based statistical computing. Comput. Stat. 17(3): 315–328 (2002)
Klinke, S., Cook, D.: Binning of Kernel-based Projection Pursuit Indices in XGobi. Comput. Stat. Data Anal. **27**(3), 363–369 (1997)
Klösgen, W., Zytkow, J.M. (ed.): Handbook of Data Mining and Knowledge Discovery. Oxford University Press, New York, NY (2002)
Kruskal, J.B.: Multidimensional Scaling. ASA Statistical Graphics Video Library. http://stat-graphics.org/movies (1962)
Kruskal, J.B.: Toward a practical method which helps uncover the structure of a set of observations by finding the line transformation which optimizes a new "Index of Condensation". In: Milton, R.C., Nelder, J.A. (eds.) Statistical Computation, pp. 427–440. Academic Press, New York (1969)
Maar, D.: Vision. Freeman, New York, NY (1982)
MacDougall, E.B.: Exploratory analysis, dynamic statistical visualization, and geographic information systems. Cartography Geogr. Inform. Syst. **19**(4), 237–246 (1992)
Macedo, M., Cook, D., Brown, T.J.: Visual data mining in atmospheric science data. Data Min. Knowl. Discov. **4**(1), 69–80 (2000)
MathSoft: S+GISLink. MathSoft, Incorporation, Seattle, WA (1996)
McDonald, J.A., Willis, S.: Use of the Grand Tour in Remote Sensing. ASA Statistical Graphics Video Library http://stat-graphics.org/movies (1987).
Monmonier, M.: Geographical Representations in Statistical Graphics: A Conceptual Framework. In: 1988 Proceedings of the Section on Statistical Graphics, pp. 1–10, American Statistical Association, Alexandria, VA (1988)
Monmonier, M.: Geographic brushing: Enhancing exploratory analysis of the scatterplot matrix. Geogr. Anal. **21**(1), 81–84 (1989)
Morton, S.C.: Interpretable Projection Pursuit. Technical Report 106, Laboratory for Computational Statistics, Stanford University (1989)
Morton, S.C.: Interpretable Exploratory Projection Pursuit. In: Page, C., LePage, R. (eds.) Proceedings of the 22nd Symposium on the Interface between Computing Science and Statistics, pp. 470–474. Springer, New York, NY (1992)
Murdoch, D.J.: Drawing a scatterplot. Chance **13**(3), 53–55 (2002)
Nagel, M.: Interactive Analysis of Spatial Data. In: Dirschedl, P., Ostermann, R. (eds.) Computational Statistics, pp. 295–314. Physica, Heidelberg (1994)
Nagel, M., Benner, A., Ostermann, R., Henschke, K.: Grafische Da- tenanalyse. Gustav Fischer, Stuttgart, German (1996)
Nelson, L., Cook, D., Cruz-Neira, C.: XGobi vs the C2: An Experiment Comparing Data Visualization in an Immersive Virtual Environment with a Workstation Display. In: Cruz-Neira, C., Riedel, O. (eds.) 2nd International Immersive Projection Technology Workshop, May 11–12, 1998, Iowa State University, Ames, IA, (CD) (1998)
Nelson, L., Cook, D., Cruz-Neira, C.: XGobi vs the C2: Results of an Experiment Comparing Data Visualization in a 3-D Immersive Virtual Reality Environment with a 2-D Workstation Display. Comput. Stat.: Special Issue on Interactive Graphical Data Analysis **14**(1), 39–51 (1999)
Openshaw, S., Perrée, T.: User-Centred Intelligent Spatial Analysis of Point Data. In: Parker, D. (eds.) Innovations in GIS 3, pp. 119–134. Taylor & Francis, London, UK (1996)
Ostermann, R., Nagel, M.: Dynamic graphics for discrete data. Comput. Stat. **8**, 197–205 (1993)

Pimentel, K., Teixeira, K.: Virtual Reality through the New Looking Glass, (2nd edn,). McGraw-Hill, New York, NY (1995)

Posse, C.: Tools for two-dimensional exploratory projection pursuit. J. Comput. Graph. Stat. **4**(2), 83–100 (1995)

Rao, C.R. (ed.): Handbook of Statistics, Vol. 9: Computational Statistics. North Holland/Elsevier Science Publishers, Amsterdam (1993)

Rollmann, W.: Notiz zur Stereoskopie. Annalen der Physik und Chemie **89**, 350–351 (1853a); in German

Rollmann, W.: Zwei neue stereoskopische Methoden. Annalen der Physik und Chemie **90**, 186–187 (1853b); in German.

Rösch, S.: 100 Jahre Anaglyphen. Die Farbe **3**(1/2), 1–6 (1954); in German.

Rossini, A.J., West, R.W.: Virtual reality and statistical research. Comput. Sci. Stat. **29**(1), 215–219 (1998)

Roy, T., Cruz-Neira, C., DeFanti, T.A.: Cosmic Worm in the CAVE: Steering a High Performance Computing Application from a Virtual Environment. Presence Teleoperators Virt. Environ. **4**(2), 121–129 (1995)

Schneider, M., Stamm, C., Symanzik, J., Widmayer, P.: Virtual Reality and Dynamic Statistical Graphics: A Bidirectional Link in a Heterogeneous, Distributed Computing Environment. In: Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA'2000), Las Vegas, Nevada, June 26–29, 2000, vol. IV, pp. 2345–2351. CSREA Press (2000)

Scott, D.W.: Multivariate Density Estimation: Theory, Practice, and Visualization. Wiley, New York, NY (1992)

Scott, L.M.: Identification of a GIS Attribute Error Using Exploratory Data Analysis. Prof. Geogr. **46**(3), 378–386 (1994)

Soukop, T., Davidson, I.: Visual Data Mining. Wiley, New York, NY (2002)

Stuetzle, W.: Plot Windows. In: Cleveland, W.S., McGill, M.E. (eds.) Dynamic Graphics for Statistics, pp. 225–245. Wadsworth & Brooks/Cole, Belmont, CA (1988)

Swayne, D.F., Buja, A.: Missing Data in Interactive High-Dimensional Data Visualization. Comput. Stat.: Special Issue on Strategies for Data Analysis **13**(1), 15–26 (1998)

Swayne, D.F., Cook, D.: Xgobi: A Dynamic Graphics Program Implemented in X With a Link to S. Comput. Sci. Stat. **22**, 544–547 (1992)

Swayne, D.F., Cook, D., Buja, A.: XGobi: Interactive Dynamic Graphics in the X Window System with a Link to S. In: 1991 Proceedings of the Section on Statistical Graphics, pp. 1–8, American Statistical Association, Alexandria, VA (1991)

Swayne, D.F., Cook, D., Buja, A.: XGobi: Interactive dynamic graphics in the X window system. J. Comput. Graph. Stat. **7**(1), 113–130 (1998)

Swayne, D.F., Temple Lang, D., Buja, A., Cook, D.: GGobi: Evolving from XGobi into an Extensible Framework for Interactive Data Visualization. Comput. Stat. Data Anal.: Special Issue on Data Visualization **43**(4), 423–444 (2003)

Symanzik, J.: Computerdarstellungen von Anaglyphen – Ein Hilfsmittel der multivariaten Statistik. Diplomarbeit, Fachbereich Informatik, Universität Dortmund, German (1992)

Symanzik, J.: Anaglyphen 3D – A Program for the Interactive Representation of Three-Dimensional Perspective Plots of Statistical Data. In: Opitz, O., Lausen, B., Klar, R. (eds.) Information and Classification. Concepts, Methods and Applications. Proceedings of the 16th Annual Conference of the "Gesellschaft für Klassifikation e. V.", pp. 384–389, Springer, Berlin, Heidelberg (1993a)

Symanzik, J.: Three-Dimensional Statistical Graphics based on Interactively Animated Anaglyphs. In: 1993 Proceedings of the Section on Statistical Graphics, pp. 71–76. American Statistical Association, Alexandria, VA (1993b)

Symanzik, J.: Interview with Andreas Buja. Comput. Stat. **23**(2):177–184 (2008)

Symanzik, J., Ascoli, G.A., Washington, S.S., Krichmar, J.L.: Visual data mining of brain cells. Comput. Sci. Stat. **31**, 445–449 (1999a)

Symanzik, J., Axelrad, D.A., Carr, D.B., Wang, J., Wong, D., Woodruff, T.J.: HAPs, Micromaps and GPL – Visualization of Geographically Referenced Statistical Summaries on the World Wide Web. In: Annual Proceedings (ACSM-WFPS-PLSO-LSAW 1999 Conference CD). American Congress on Surveying and Mapping (1999b)

Symanzik, J., Carr, D.B., Axelrad, D.A., Wang, J., Wong, D., Woodruff, T.J.: Interactive Tables and Maps – A Glance at EPA's Cumulative Exposure Project Web Page. In: 1999 Proceedings of the Section on Statistical Graphics, pp. 94–99. American Statistical Association, Alexandria, VA (1999c)

Symanzik, J., Cook, D., Klinke, S., Lewin, N.: Exploration of Satellite Images in the Dynamically Linked ArcView/XGobi/XploRe Environment. In: Bodt, B.A. (eds.) Proceedings of the Third Annual U.S. Army Conference on Applied Statistics, 22–24 October 1997, pp. 23–33. Aberdeen Proving Ground, MD. Army Research Laboratory ARL-SR-74 (1998a)

Symanzik, J., Cook, D., Kohlmeyer, B.D., Cruz-Neira, C.: Dynamic Statistical Graphics in the CAVE Virtual Reality Environment. Working Paper for the Dynamic Statistical Graphics Workshop, Sydney, July 7, 1996. Department of Statistics, Iowa State University, Ames, IA (1996a); Preprint 96–10, available at http://www.math.usu.edu/~symanzik/papers/1996_dsg.pdf.

Symanzik, J., Cook, D., Kohlmeyer, B.D., Lechner, U., Cruz-Neira, C.: Dynamic statistical graphics in the C2 virtual reality environment. Comput. Sci. Stat. **29**(2), 41–47 (1997)

Symanzik, J., Cook, D., Lewin-Koh, N., Majure, J.J., Megretskaia, I.: Linking arcview and XGobi: Insight behind the front end. J. Comput. Graph. Stat. **9**(3), 470–490 (2000a)

Symanzik, J., Hurst, J., Gunter, L.: Recent Developments for Interactive Statistical Graphics on the Web Using "nViZn". In: 2002 Proceedings, American Statistical Association, Alexandria, VA (CD) (2002a)

Symanzik, J., Jones, L.: "nViZn" Federal Statistical Data on the Web. In 2001 Proceedings, American Statistical Association, Alexandria, VA (CD) (2001)

Symanzik, J., Majure, J.J., Cook, D.: Dynamic Graphics in a GIS: A Bidirectional Link between ArcView 2.0 and XGobi. Comput. Sci. Stat. **27**, 299–303 (1996b)

Symanzik, J., Pajarola, R., Widmayer, P.: XGobi and XploRe Meet ViRGIS. In: 1998 Proceedings of the Section on Statistical Graphics, pp. 50–55. American Statistical Association, Alexandria, VA (1998b)

Symanzik, J., Wegman, E.J., Braverman, A.J., Luo, Q.: New applications of the image grand tour. Comput. Sci. Stat. **34**, (CD) (2002b)

Symanzik, J., Wong, D., Wang, J., Carr, D.B., Woodruff, T.J., Axelrad, D.A.: Web-based Access and Visualization of Hazardous Air Pollutants. In Geographic Information Systems in Public Health: Proceedings of the Third National Conference August pp. 18–20, 1998, San Diego, California (2000b); Agency for Toxic Substances and Disease Registry. http://www.atsdr.cdc.gov/GIS/conference98/.

Theus, M.: Theorie und Anwendung Interaktiver Statistischer Graphik. Dr. Bernd Wißner, Augsburg; German (1996)

Theus, M.: Interactive data visualization using Mondrian. J. Stat. Software **7**(11), (2002); http://www.jstatsoft.org/v07/i11/.

Theus, M.: Abstract: Interactive data visualization using Mondrian. J. Comput. Graph. Stat. **12**(1), 243–244 (2003)

Theus, M., Hofmann, H., Wilhelm, A.F.X.: Selection sequences – interactive analysis of massive data sets. Comput. Sci. Stat. **29**(1), 439–444 (1998)

Theus, M., Urbanek, S.: Interactive Graphics for Data Analysis, Principles and Examples. Chapman & Hall/CRC, Boca Raton (2009)

Theus, M., Wilhelm, A.F.X.: Counts, Proportions, Interactions – A View on Categorical Data. In: 1998 Proceedings of the Section on Statistical Graphics, pp. 6–15. American Statistical Association, Alexandria, VA (1998)

Tukey, J.W.: Exploratory Data Analysis. Addison Wesley, Reading, MA (1977)

Tweedie, L., Spence, R.: The prosection matrix: A tool to support the interactive exploration of statistical models and data. Comput. Stat.: Special Issue on Strategies for Data Analysis **13**(1), 65–76 (1998)

Unwin, A.: REGARDing Geographic Data. In: Dirschedl, P., Ostermann, R. (eds.) Computational Statistics, pp. 315–326. Physica, Heidelberg (1994)

Unwin, A.: Requirements for Interactive Graphics Software for Exploratory Data Analysis. Comput. Stat.: Special Issue on Interactive Graphical Data Analysis **14**(1), 7–22 (1999)

Unwin, A.: Scatterplotting. Chance **15**(2), 39–42 (2002)

Unwin, A., Hawkins, G., Hofmann, H., Siegl, B.: Interactive graphics for data sets with missing values – MANET. J. Comput. Graph. Stat. **5**(2), 113–122 (1996)

Unwin, A., Wills, G.: Eyeballing Time Series. In: 1988 Proceedings of the Section on Statistical Computing, pp. 263–268. American Statistical Association, Alexandria, VA (1988)

Unwin, A., Wills, G., Haslett, J.: REGARD – Graphical Analysis of Regional Data. In: 1990 Proceedings of the Section on Statistical Graphics, pp. 36–41. American Statistical Association, Alexandria, VA (1990)

Valero-Mora, P.M., Young, F.W., Friendly, M.: Visualizing categorical data in ViSta. Comput. Stat. Data Anal.: Special Issue on Data Visualization, **43**(4), 495–508 (2003)

van Teylingen, R., Ribarsky, W., van der Mast, C.: Virtual data visualizer. IEEE Transactions on Visualization and Computer Graphics **3**(1), 65–74 (1997)

Vince, J.: Virtual Reality Systems. ACM Press/Addison-Wesley, Wokingham, UK (1995)

Vuibert, H.: Les Anaglyphes Géométriques. Librairie Vuibert, Paris; French (1912)

Wang, X., Chen, J.X., Carr, D.B., Bell, B.S., Pickle, L.W.: Geographic Statistics Visualization: Web-based Linked Micromap Plots. Comput. Sci. Eng. **4**(3), 90–94 (2002)

Wegman, E.J.: Hyperdimensional data analysis using parallel coordinates. J. Am. Stat. Assoc. **85**, 664–675 (1990)

Wegman, E.J.: The grand tour in k-dimensions. Comput. Sci. Stat. **22**, 127–136 (1992)

Wegman, E.J.: Huge data sets and the frontiers of computational feasibility. J. Comput. Graph. Stat. **4**(4), 281–295 (1995)

Wegman, E.J.: Visions: New techniques and technologies in statistics. Comput. Stat.: Special Issue on New Techniques and Technologies for Statistics **15**(1), 133–144 (2000)

Wegman, E.J., Carr, D.B.: Statistical Graphics and Visualization. In: Rao, C.R. (eds.) Handbook of Statistics, Vol. 9: Computational Statistics, pp. 857–958. North Holland/Elsevier Science Publishers, Amsterdam (1993)

Wegman, E.J., DePriest, D.J. (ed.): Statistical Image Processing and Graphics. Marcel Dekker, New York, NY (1986)

Wegman, E.J., Dorfman, A.: Visualizing cereal world. Comput. Stat. Data Anal.: Special Issue on Data Visualization **43**(4), 633–649 (2003)

Wegman, E.J., Luo, Q.: High dimensional clustering using parallel coordinates and the grand tour. Comput. Sci. Stat. **28**, 361–368 (1997a)

Wegman, E.J., Luo, Q.: High Dimensional Clustering Using Parallel Coordinates and the Grand Tour. In: Klar, R., Opitz, O. (eds.) Classification and Knowledge Organization, pp. 93–101. Springer, New York (1997b)

Wegman, E.J., Poston, W.L., Solka, J.L.: Image Grand Tour. In Automatic Target Recognition VIII – Proceedings of SPIE, 3371, 286–294; republished in: Sadjadi, F. (Eds.) vol. 6, Automatic Target Recognition. The CD-ROM, 1999. SPIE, Bellingham, WA (1998)

Wegman, E.J., Shen, J.: Three-dimensional andrews plots and the grand tour. Comput. Sci. Stat. **25**, 284–288 (1993)

Wegman, E.J., Symanzik, J.: Immersive projection technology for visual data mining. J. Comput. Graph. Stat. **11**(1), 163–188 (2002)

Wegman, E.J., Symanzik, J., Vandersluis, J.P., Luo, Q., Camelli, F., Dzubay, A., Fu, X., Khumbah, N.-A., Moustafa, R.E.A., Wall, R.L., Zhu, Y.: The MiniCAVE – A Voice-Controlled IPT Environment. In: Bullinger, H.-J., Riedel, O. (eds.) 3. International Immersive Projection Technology Workshop, 10./11. May 1999, Center of the Fraunhofer Society Stuttgart IZS, pp. 179–190. Springer, Berlin, Heidelberg (1999)

Wilhelm, A.F.X., Unwin, A., Theus, M.: Software for Interactive Statistical Graphics – A Review. In: Faulbaum, F., Bandilla, W., (eds.) SoftStat '95 – Advances in Statistical Software 5, pp. 3–12. Stuttgart. Lucius & Lucius (1996)

Wilhelm, A.F.X., Wegman, E.J., Symanzik, J.: Visual Clustering and Classification: The Oronsay Particle Size Data Set Revisited. Comput. Stat.: Special Issue on Interactive Graphical Data Analysis, **14**(1), 109–146 (1999)

Wilkinson, L.: The Grammar of Graphics. Springer, New York, NY (1999)

Wilkinson, L., Rope, D.J., Carr, D.B., Rubin, M.A.: The language of graphics. J. Comput. Graph. Stat. **9**(3), 530–543 (2000)

Williams, I., Limp, W.F., Briuer, F.L.: Using Geographic Information Systems and Exploratory Data Analysis for Archaeological Site Classification and Analysis. In: Allen, K.M.S., Green, S.W., Zubrow, E.B.W. (eds.) Interpreting Space: GIS and Archaeology, pp. 239–273. Taylor & Francis, London, UK (1990)

Witten, I.H., Frank, E.: Data Mining – Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann/Academic Press, San Francisco, CA/San Diego, CA (2000)

Wojciechowski, W.C., Scott, D.W.: High-dimensional visualization using continuous conditioning. Comput. Sci. Stat. **32**, (CD) (2000)

Yoshioka, K.: KyPlot – A user-Oriented tool for statistical data analysis and visualization. Comput. Stat. **17**(3), 425–437 (2002)

Young, F.W., Faldowski, R.A., McFarlane, M.M.: Multivariate Statistical Visualization. In: Rao, C.R. (eds.) Handbook of Statistics, vol. 9: Computational Statistics, pp. 959–998. North Holland/Elsevier Science Publishers, Amsterdam (1993)

Zhang, Z., Griffith, D.A.: Developing user-friendly spatial statistical analysis modules for GIS: An example using ArcView. Comput. Environ. Urban Systems **21**(1), 5–29 (1997)

Chapter 13
The Grammar of Graphics

Leland Wilkinson

13.1 Introduction

The Grammar of Graphics, or GOG, denotes a system with seven orthogonal components (Wilkinson 1999). By *orthogonal*, we mean there are seven graphical component sets whose elements are aspects of the general system and that every combination of aspects in the product of all these sets is meaningful. This sense of the word orthogonality, a term used by computer designers to describe a combinatoric system of components or building blocks, is in some sense similar to the orthogonal factorial analysis of variance (ANOVA), where factors have levels and all possible combinations of levels exist in the ANOVA design. If we interpret each combination of features in a GOG system as a point in a network, then the world described by GOG is represented in a seven-dimensional rectangular lattice.

A consequence of the orthogonality of such a graphic system is a high degree of *expressiveness*. That is, it comprises a system that can produce a huge variety of graphical forms (chart types). In fact, it is claimed that virtually the entire corpus of known charts can be generated by this relatively parsimonious system, and perhaps a great number of meaningful but undiscovered chart types as well.

The second principal claim of GOG is that this system describes the meaning of what we do when we construct statistical graphs or charts. It is more than a taxonomy. It is a computational system based on the classical mathematics of representing functions and relations in Cartesian and other spaces. Because of this mathematical foundation, GOG specifications can serve as parsimonious and natural descriptions of famous statistical charts devised by Playfair, Minard, Jevons, Pearson, Bertin, Tukey, and other significant figures in the history of statistical graphics.

L. Wilkinson (✉)
SYSTAT Software Inc. Chicago, IL, USA
e-mail: Leland.Wilkinson@systat.com

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_13,

13.1.1 Architecture

Figure 13.1 shows a *dataflow* diagram that contains the seven GOG components. This dataflow is a chain that describes the sequence of mappings needed to produce a statistical graphic from a set of data. The first component (Variables) maps data to an object called a *varset* (a set of variables). The next three components (Algebra, Scales, Statistics) are transformations on varsets. The next component (Geometry) maps a varset to a graph and the next (Coordinates) embeds a graph in a coordinate space. The last component (Aesthetics) maps a graph to a visible or perceivable display called a graphic.

The dataflow architecture implies that the subtasks needed to produce a graphic from data must be done in this specified order. Imposing an order would appear to be unnecessarily restrictive, but changes of this ordering can produce meaningless graphics. For example, if we compute certain statistics on variables (e.g., sums) before scaling them (e.g., log scales), we can produce statistically questionable results because the log of a sum is not the sum of the logs.

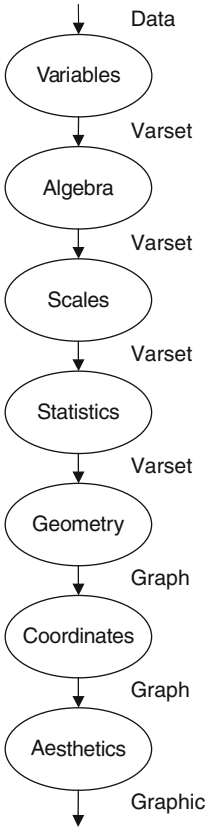


Fig. 13.1 Dataflow

The dataflow in Fig. 13.1 has many paths through it. We can choose different designs (factorial, nested, ...), scales (log, probability, ...), statistical methods (means, medians, modes, smoothers, ...), geometric objects (points, lines, bars, ...), coordinate systems (rectangular, polar, ...), and aesthetics (size, shape, color, ...). These paths reveal the richness of the system. The remainder of this article will summarize the seven GOG components, delineate these paths, and then briefly introduce sample applications.

13.2 Variables

We begin with data. We assume the data that we wish to graph are organized in one or more tables. The column(s) of each table represent a set of fields, each field containing a set of measurements or attributes. The row(s) of this table represent a set of logical records, each record containing the measurements of an object on each field. Usually, a relational database management system (RDBMS) produces such a table from organized queries specified in Structured Query Language (SQL) or another relational language. When we do not have data stored in a relational database (e.g., live data feeds), we need custom software to provide such a table using Extensible Markup Language (XML), Perl scripts, or other languages.

The first thing we have to do is convert such tables of data to something called a *varset*. A varset is a set of one or more variables. While a column of a table of data might superficially be considered to be a variable, there are differences. A variable is both more general (in regard to generalizability across samples) and more specific (in regard to data typing and other constraints) than a column of data. First, we define a variable, then a varset.

13.2.1 Variable

A statistical *variable* X is a mapping $f : O \to V$, which we consider as a triple:

$$X = [O, V, f]$$

The domain O is a set of objects.
The codomain V is a set of values.
The function f assigns to each element of O an element in V.

The image of O under f contains the *values* of X. We denote a possible value as x, where $x \in V$. We denote a value of an object as $X(o)$, where $o \in O$. A variable is *continuous* if V is an interval. A variable is *categorical* if V is a finite subset of the integers (or there exists an injective map from V to a finite subset of the integers).

Variables may be multidimensional. $\boldsymbol{X}$ is a p-dimensional variable made up of p one-dimensional variables:

$$\begin{aligned}\mathbf{X} &= (X_1, \ldots, X_p) \\ &= [O, V_i, f\,], \quad i = 1, \ldots, p \\ &= [O, \mathbf{V}, f\,]\end{aligned}$$

The element $\mathbf{x} = (x_1, \ldots, x_p)$, $\mathbf{x} \in \mathbf{V}$, is a p-dimensional value of $\mathbf{X}$. We use multidimensional variables in *multivariate analysis*.

A *random variable* X is a real–valued function defined on a sample space Ω:

$$X = [\Omega, \mathbb{R}, f]$$

Random variables may be multidimensional. In the elementary probability model, each element $\omega \in \Omega$ is associated with a probability function P. The range of P is the interval $[0, 1]$, and $P(\Omega) = 1$. Because of the associated probability model, we can make probability statements about outcomes in the range of the random variable, such as:

$$P(X = 2) = P(\omega : \omega \in \Omega, X(\omega) = 2)$$

13.2.2 Varset

We call the triple

$$\mathrm{X} = [V, \widetilde{O}, f]$$

a varset. The word *varset* stands for *variable set*. If X is multidimensional, we use boldface $\mathbf{X}$. A varset inverts the mapping used for variables. That is,
The domain V is a set of values.
The codomain $\widetilde{O}$ is a set of all possible ordered lists of objects.
The function f assigns to each element of V an element in $\widetilde{O}$.

We invert the mapping customarily used for variables in order to simplify the definitions of graphics algebra operations on varsets. In doing so, we also replace the variable's set of objects with the varset's set of ordered lists. We use lists in the codomain because it is possible for a value to be mapped to an object more than once (as in repeated measurements).

13.2.3 Converting a Table of Data to a Varset

To convert a table to a varset, we must define the varset's domain of values and range of objects and specify a reference function that maps each row in the table to an element in the varset.

We define the domain of values in each varset by identifying the measurements its values represent. If our measurements include weight, for example, we need to record the measurement units and the interval covered by the range of weights. If the domain is categorical, we need to decide whether there are categories not found in the data that should be included in the domain (refused to reply, don't know, missing, ...). And we may need to identify categories found in the data that are not defined in the domain (mistakes, intransitivities, ...). The varset's domain is a key component in the GOG system. It is used for axes, legends, and other components of a graphic. Because the actual data for a chart are only an instance of what we expect to see in a varset's domain, we let the domain control the structure of the chart.

We define the range of potential objects in each varset by identifying the class they represent. If the rows of our table represent measurements of a group of school children, for example, we may define the range to be school children, children in general, people, or (at the most abstract level) objects. Our decision about the level of generality may affect how a graphic will be titled, how legends are designed, and so on.

Finally, we devise a reference system by indexing objects in the domain. This is usually as simple as deriving a *caseID* from a table row index. Or, as is frequently done, we may choose the value of a key variable (e.g., Social Security Number) to create a unique index.

13.3 Algebra

Given one or more varsets, we now need to operate on them to produce combinations of variables. A typical scatterplot of a variable X against a variable Y, for example, is built from tuples (x_i, y_i) that are elements in a set product. We use graphics algebra on values stored in varsets to make these tuples. There are three binary operators in this algebra: *cross*, *nest*, and *blend*.

13.3.1 Operators

We will define these operators in set notation and illustrate them by using a table of real data. Table 13.1 shows 1980 and 2000 populations for selected world cities. During various periods in US history, it was fashionable to name towns and cities after their European and Asian counterparts. Sometimes this naming was driven by immigration, particularly in the colonial era (New Amsterdam, New York, New London). At other times, exotic names reflected a fascination with foreign travel and culture, particularly in the Midwest (Paris, Madrid). Using a dataset containing namesakes will help reveal some of the subtleties of graphics algebra.

Table 13.1 Cities and their Populations

Country	City	1980 Population	2000 Population
Japan	Tokyo	21900000	26400000
India	Mumbai	8067000	18100000
USA	New York	15600000	16600000
Nigeria	Lagos	4385000	13400000
USA	Los Angeles	9523000	13100000
Japan	Osaka	9990000	11000000
Philippines	Manila	5955000	10900000
France	Paris	8938000	9624000
Russia	Moscow	8136000	9321000
UK	London	7741000	7640000
Peru	Lima	4401000	7443000
USA	Chicago	6780000	6951000
Iraq	Bagdad	3354000	4797000
Canada	Toronto	3008000	4651000
Spain	Madrid	4296000	4072000
Germany	Berlin	3247000	3324000
Australia	Melbourne	2765000	3187000
USA	Melbourne	46536	71382
USA	Moscow	16513	21291
USA	Berlin	13084	10331
USA	Paris	9885	9077
USA	London	4002	5692
USA	Toronto	6934	5676
USA	Manila	2553	3055
USA	Lima	2025	2459
USA	Madrid	2281	2264
USA	Bagdad	2331	1578

We begin by assuming there are four varsets derived from this table: `country`, `city`, `pop1980`, and `pop2000` (we use lower case for varsets when they are denoted by names instead of single letters). Each varset has one column. The varsets resulting from algebraic operations will have one or more columns.

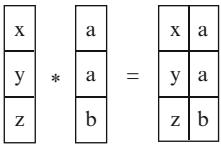
There is one set of objects for all four varsets: 27 cities. This may or may not be a subset of the domain for the four associated variables. If we wish to generalize analyses of this varset to other cities, then the set of possible objects in these varsets might be a subdomain of the set of all cities existing in 1980 and 2000. We might even consider this set of objects to be a subset of all possible cities in all of recorded history. While these issues might seem more the province of sampling and generalizability theory, they affect the design of a graphics system. Databases, for example, include facilities for *semantic integrity constraints* that ensure domain integrity in data tables. Data-based graphics systems share similar requirements.

There are sets of values for these varsets. The `country` varset has country names in the set of values comprising its domain. The definition of the domain

of the varset depends on how we wish to use it. For example, we might include spellings of city names in languages other than English. We might also include country names not contained in this particular varset. Such definitions would affect whether we could add new cities to a database containing these data. For `pop1980` and `pop2000`, we would probably make the domain be the set of positive integers.

Cross (∗)

Cross joins the left argument with the right to produce a set of tuples stored in the multiple columns of the new varset:



The resulting set of tuples is a subset of the product of the domains of the two varsets. The domain of a varset produced by a cross is the product of the separate domains.

One may think of a cross as a horizontal concatenation of the table representation of two varsets, assuming the rows of each varset are equivalent and in the same order. The following example shows a crossing of two varsets using set notation with simple integer keys for the objects:

$$\begin{aligned} \mathrm{A} &= [\{red, blue\}, \{\langle\cdot\rangle, \langle\cdot,\cdot\rangle, \ldots\}, \{red \to \langle 1,4\rangle, blue \to \langle 2,3\rangle\}] \\ \mathrm{B} &= [[-10,10], \{\langle\cdot\rangle, \langle\cdot,\cdot\rangle, \ldots\}, \{-10 \to \langle 1\rangle, 5 \to \langle 2,3\rangle, 10 \to \langle 4\rangle\}] \\ \mathrm{A} * \mathrm{B} &= [\{red, blue\} \times [-10,10], \{\langle\cdot\rangle, \langle\cdot,\cdot\rangle, \ldots\}, \\ &\qquad \{(red,-10) \to \langle 1\rangle, (blue,5) \to \langle 2,3\rangle, (red,10) \to \langle 4\rangle\}] \end{aligned}$$

If we plotted A ∗ B in two dimensions with a point graph, we would see n points between −10 and 10 stacked vertically above one or both of the two color names.

Figure 13.2 shows a graphic based on the algebraic expression `city*pop2000`. We choose the convention of representing the first variable in an expression on the horizontal axis and the second on the vertical. We also restrict the domain of `pop2000` to be [0, 32000000].

Although most of the US namesake cities have smaller populations, it is not easy to discern them in the graphic. We can separate the US from the other cities by using a variable called `group` that we derive from the country names. Such a new variable is created easily in a database or statistical transformation language with an expression like

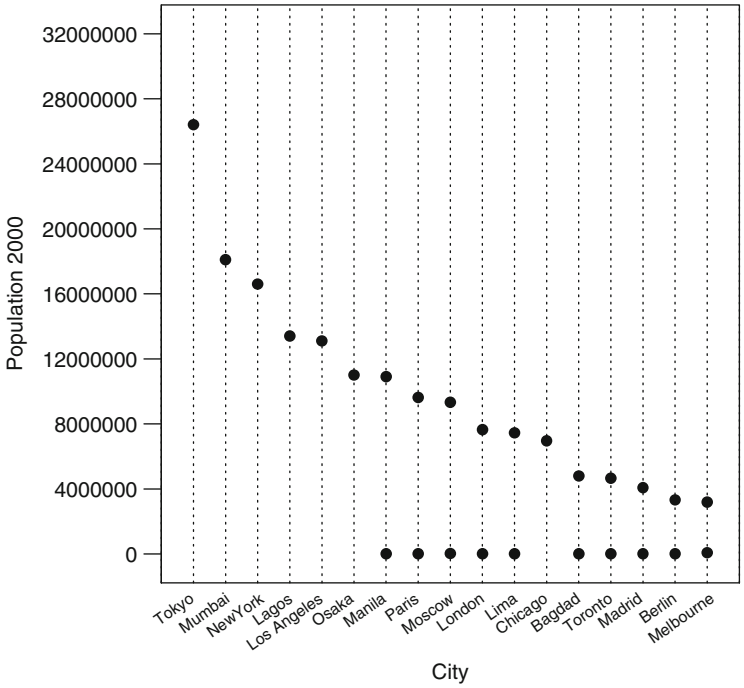


Fig. 13.2 `city*pop2000`

```
if (country == "USA") group = "USA";
else group = "World";
```

Figure 13.3 shows a graphic based on the three-dimensional algebraic expression `city*pop2000*group`. This expression produces a varset with three columns. The first column is assigned to the horizontal axis, the second to the vertical, and the third to the horizontal axis again, which has the effect of splitting the frame into two frames. This general pattern of alternating horizontal and vertical roles for the columns of a varset provides a simple layout scheme for complex algebraic expressions. We may think of this as a generalization of the Trellis layout scheme (Becker et al. 1996). We could, of course, represent this same varset in a 3D plot projected into 2D, but the default system behavior is to prefer 2D with recursive partitioning. We will describe this in more detail in Sect. 13.9.

Chicago stands out as an anomaly in Fig. 13.3 because of its relatively large population. We might want to sort the cities in a different order for the left panel or eliminate cities not found in the US, but the algebraic expression won't let us do that. Because `group` is crossed with the other variables, there is only one domain of cities shared by both country groups. If we want to have different domains for the two panels, we need our next operator, *nest*.

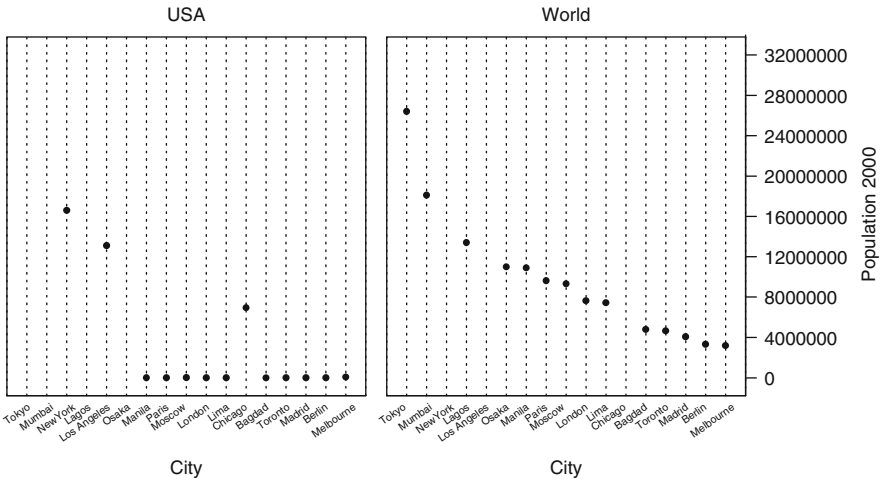


Fig. 13.3 `city * pop2000 * group`

Nest (/)

Nest partitions the left argument using the values in the right:

x
y
z

/

a
a
b

=

x	a
y	a
z	b

Although it is not required in the definition, we assume the nesting varset on the right is categorical. If it were continuous (having interval domain) there would be an infinite number of partitions. We do require predefined nested domains. To construct a nested domain, three options are possible:

1. Data values – identify the minimal domain containing the data by enumerating unique data tuples.
2. Metadata – define the domain using external rules contained in a metadata resource or from known principles.
3. Data organization – identify nested domains using the predefined structure of a hierarchical database or OLAP cube.

The following example shows a nesting of two categorical variables:

$$\mathrm{A} = [\{ant, fly, bee\}, \{\langle\cdot\rangle, \langle\cdot,\cdot\rangle, \ldots\}, \{ant \to \langle 1\rangle, fly \to \langle 2,3\rangle, bee \to \langle 4\rangle\}]$$

$$\mathrm{B} = [\{noun, verb\}, \{\langle\cdot\rangle, \langle\cdot,\cdot\rangle, \ldots\}, \{noun \to \langle 1,2,4\rangle, verb \to \langle 3\rangle\}]$$

$$\mathrm{A/B} = [\{(ant, noun), (fly, noun), (fly, verb), (bee, noun)\}, \{\langle\cdot\rangle, \langle\cdot,\cdot\rangle, \ldots\},$$

$$\{(ant, noun) \rightarrow \langle 1\rangle, (fly, noun) \rightarrow \langle 2\rangle, (fly, verb) \rightarrow \langle 3\rangle (bee, noun) \rightarrow \langle 4\rangle\}]$$

Nesting defines meaning conditionally. In this example, the meaning of *fly* is ambiguous unless we know whether it is a noun or a verb. Furthermore, there is no verb for *ant* or *bee* in the English language, so the domain of A/B does not include this combination.

If A is a continuous variable, then we have something like the following:

$$\begin{aligned}
\mathrm{A} &= [[0, 10], \{\langle\cdot\rangle, \langle\cdot, \cdot\rangle, \ldots\}, \{0 \rightarrow \langle 1\rangle, 8 \rightarrow \langle 2\rangle, 1.4 \rightarrow \langle 3\rangle, 3 \rightarrow \langle 4\rangle, 10 \rightarrow \langle 5, 6\rangle\}] \\
\mathrm{B} &= [\{1, 2\}, \{\langle\cdot\rangle, \langle\cdot, \cdot\rangle, \ldots\}, \{1 \rightarrow \langle 1, 2, 3\rangle, 2 \rightarrow \langle 4, 5, 6\rangle\}] \\
\mathrm{A/B} &= [\{[0, 8] \times \{1\}, [3, 10] \times \{2\}\}, \{\langle\cdot\rangle, \langle\cdot, \cdot\rangle, \ldots\}, \\
&\quad \{(0, 1) \rightarrow \langle 1\rangle, (8, 1) \rightarrow \langle 2\rangle, (1.4, 1) \rightarrow \langle 3\rangle, (3, 2) \rightarrow \langle 4\rangle, (10, 2) \rightarrow \langle 5, 6\rangle\}]
\end{aligned}$$

In this example, the elements of the nesting A/B result in intervals conditioned on the values of B. A represents 6 ratings (ranging from 0 to 10) of the behavior of patients by two psychiatrists. B represents the identity of the psychiatrist making each rating. The intervals $[0, 8]$ and $[3, 10]$ imply that psychiatrist 1 will not use a rating greater than 8 and psychiatrist 2 will not use a rating less than 3. Nesting in this case is based on the (realistic) assumption that the two psychiatrists assign numbers to their perceptions in a different manner. A rating of 2 by one psychiatrist cannot be compared to the same rating by the other, because of possible differences in location, scale, and even local nonlinearities. Much of psychometrics is concerned with the problem of equating ratings in this type of example so that nesting would not be needed, although it is not always possible to do so plausibly.

The name *nest* comes from design-of-experiments terminology. We often use the word *within* to describe its effect. For example, if we assess schools and teachers in a district, then *teachers within schools* specifies that teachers are nested within schools. Assuming each teacher in the district teaches at only one school, we would conclude that if our data contain two teachers with the same name at different schools, they are different people. Those familiar with experimental design may recognize that the expression A/B is equivalent to the notation A(B) in a design specification. Both expressions mean *A is nested within B*. Statisticians' customary use of parentheses to denote nesting conceals the fact that nesting involves an operator, however. Because nesting is distributive over blending, we have made this operator explicit and retained the conventional mathematical use of parentheses in an algebra.

Figure 13.4 shows a graphic based on the algebraic expression `city/group * pop2000`. The horizontal axis in each panel now shows a different set of cities: one for the USA and one for the rest of the world. This graphic differs from the one in Fig. 13.3 not only because the axes *look* different, but also because the meanings

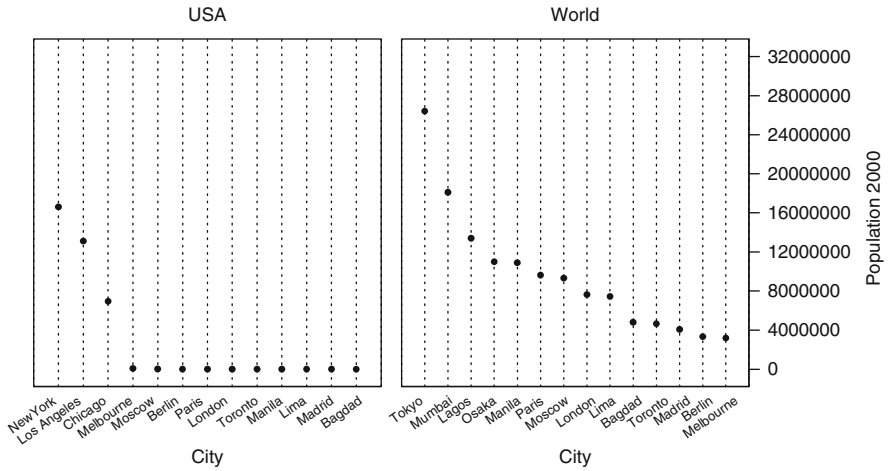
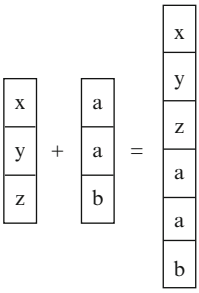


Fig. 13.4 `city/group * pop2000`

of the cities in each panels *are* different. For example, the city named Paris appears twice in both figures. In Fig. 13.3, on the one hand, we assume the name Paris in the left panel is comparable to the name Paris in the right. That is, it refers to a common name (Paris) occurring in two different contexts. In Fig. 13.4, on the other hand, we assume the name Paris references two different cities. They happen to have the same name, but are not equivalent. Such distinctions are critical, but often subtle.

Blend (+)

Blend produces a union of varsets:



Blend is defined only if the order of the tuples (number of columns) in the left and right varsets is the same. Furthermore, we should restrict blend to varsets with composable domains, even though we do not need this restriction for the operation to be defined. It would make little sense to blend Age and Weight, much less Name and Height.

In vernacular, we often use the conjunction *and* to signify that two sets are blended into one (although the word *or* would be more appropriate technically). For example, if we measure diastolic and systolic blood pressure among patients in various treatment conditions and we want to see blood pressure plotted on a common axis, we can plot diastolic *and* systolic against treatment. The following example shows a blending of two varsets, using integers for keys:

$$\begin{aligned} \mathrm{A} &= [[0, 120], \{\langle\cdot\rangle, \langle\cdot,\cdot\rangle, \ldots\}, \{0 \to \langle 1\rangle, 120 \to \langle 2\rangle, 90 \to \langle 3, 4\rangle\}] \\ \mathrm{B} &= [[10, 200], \{\langle\cdot\rangle, \langle\cdot,\cdot\rangle, \ldots\}, \{10 \to \langle 1\rangle, 200 \to \langle 2, 3\rangle, 90 \to \langle 4\rangle\}] \\ \mathrm{A} + \mathrm{B} &= [[0, 200], \{\langle\cdot\rangle, \langle\cdot,\cdot\rangle, \ldots\}, \\ &\quad \{0 \to \langle 1\rangle, 10 \to \langle 1\rangle, 120 \to \langle 2\rangle, 90 \to \langle 3, 4, 4\rangle, 200 \to \langle 2, 3\rangle\}] \end{aligned}$$

Figure 13.5 shows an example of a blend using our cities data. The graphic is based on the algebraic expression `city*(pop1980+pop2000)`. The horizontal axis represents the cities and the vertical axis represents the two repeated population measures. We have included different symbol types and a legend to distinguish the measures. We will see later how shape aesthetics are used to create this distinction.

As with the earlier graphics, we see that it is difficult to distinguish US and world cities. Figure 13.6 makes the distinction clear by splitting the horizontal axis into two nested subgroups. The graphic is based on the algebraic expression `(city/group)*(pop1980+pop2000)`. Once again, the vertical axis represents

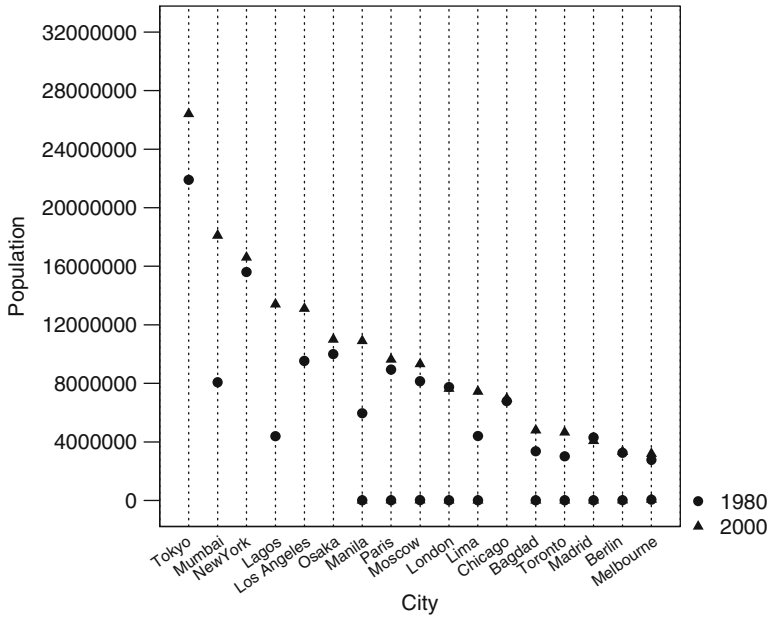


Fig. 13.5 `city*(pop1980+pop2000)`

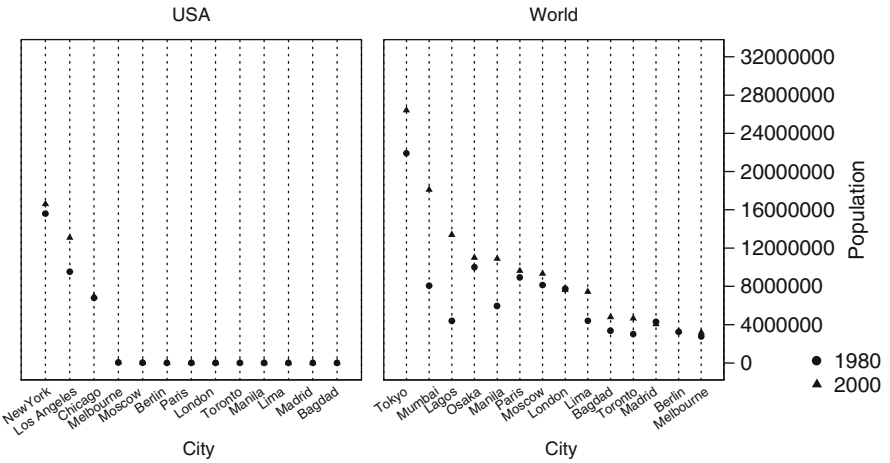


Fig. 13.6 `(city/group) * (pop1980 + pop2000)`

the two repeated population measures blended on a single dimension. We see most of the cities gaining population between 1980 and 2000.

13.3.2 Rules

The following rules are derivable from the definitions of the graphics operators:

Associativity

$$(X * Y) * Z = X * (Y * Z)$$
$$(X/Y)/Z = X/(Y/Z)$$
$$(X + Y) + Z = X + (Y + Z)$$

Distributivity

$$X * (Y + Z) = X * Y + X * Z$$
$$X/(Y + Z) = X/Y + X/Z$$
$$(X + Y) * Z = X * Z + Y * Z$$
$$(X + Y)/Z = X/Z + Y/Z$$

Commutativity

$$X + Y = Y + X$$

Identity

The identity element for blend is an empty list. Cross and nest have no identity.

Precedence

Nest takes precedence over cross and blend. Cross takes precedence over blend. This hierarchical order may be altered through the use of parentheses.

13.3.3 SQL Equivalences

Given a table X and a table Y in a database, we can use SQL to perform the operations in chart algebra. This section outlines how to do this.

Cross

Cross can be accomplished by a *cross join*:

```
SELECT a.*, b.*
FROM X a, Y b;
```

Of course, this operation is inefficient and requires optimization. Alternatively, one can do a simple *join* and generate the missing tuples with an iterator when needed.

Nest

Nest can be accomplished through a *nest* operation. The nest operator requires that the database allow tables as primitives, either as relation-valued attributes (Date and Darwen 1992) or as nested tables (Makinouchi 1977), (Abiteboul et al. 1989).

Alternatively, we can accumulate the subset of tuples in a nest operation with a simple *join*:

```
SELECT a.*, b.*
FROM X a, Y b
WHERE a.rowid = b.rowid;
```

If we use this latter method, we must distinguish the entries used for tags and those used for values.

Blend

Blend is performed through `UNION`. If `UNION all` is not available, we can concatenate key columns to be sure that all rows appear in the result set.

```
SELECT * from X
UNION all
SELECT * from Y;
```

Composition and Optimization

SQL statements can be composed by using the grammar for chart algebra. Compound statements can then be submitted for optimization and execution by a database compiler. Alternatively, pre-optimization can be performed on the chart algebra parse tree object and the optimized parse tree used to generate SQL. Secondary optimization can then be performed by the database compiler.

13.3.4 Related Algebras

Research on algebras that could be used for displaying data has occurred in many fields. We will summarize these approaches in separate sections.

Table Algebras

The US Bureau of Labor Statistics pioneered a language for laying out tables (Mendelssohn 1974). While not a formal algebra, this Table Production Language (TPL) contained many of the elements needed to assemble complex tables. Gyssens et al. (1996) outlined an algebra for displaying relational data; this algebra closely followed TPL, although the latter is not referenced. Wilkinson (1996) presented an algebra for structuring tables and graphics.

Design Algebras

Nelder (1965) and Wilkinson and Rogers (1973) developed a language for implementing factorial and nested experimental designs, following Fisher (1935). The operators in this language are similar to the cross and nest operators in the present paper. The algebraic design language was implemented in the GENSTAT statistical computer program for generating and analyzing general linear statistical models.

Query Algebras

Pedersen et al. (2002) described an algebra for querying OLAP cubes. The result sets from their algebraic expressions could be used for graphic displays. Agrawal et al. (1997) used a similar algebra for statistical modeling of data contained in a cube.

Display Algebras

Mackinlay (1986) developed an algebra for querying relational databases and generating charts. His general goal was to develop an intelligent system that could offer graphical responses to verbal or structural queries. Roth et al. (1994) followed a similar strategy in developing graphical representations of relational data. They extended Mackinlay's and others' ideas by using concepts from computational geometry.

13.3.5 Algebra XML

A parse tree for a given algebraic expression maps nicely to XML in a manner similar to the way MathML (http://www.w3.org/TR/MathML2/) is defined. We have developed an implementation, called VizML (http://xml.spss.com/ visualization), that includes not only the algebraic components of the specification, but also the aesthetic and geometric aspects. Ultimately, VizML makes it possible to embed chart algebraic operations in a database.

13.4 Scales

Before we compute summaries (totals, means, smoothers, . . .) and represent these summaries using geometric objects (points, lines, . . .), we must scale our varsets. In producing most common charts, we do not notice this step. When we implement log scales, however, we notice it immediately. We must log our data before averaging logs. Even if we do not compute nonlinear transformations, however, we need to specify a measurement model.

The measurement model determines how distance in a frame region relates to the ranges of the variables defining that region. Measurement models are reflected in the axes, scales, legends, and other annotations that demarcate a chart's frame. Measurement models determine how values are represented (e.g., as categories or magnitudes) and what the units of measurement are.

13.4.1 Axiomatic Measurement

In constructing scales for statistical charts, it helps to know something about the function used to assign values to objects. Stevens (1946) developed a taxonomy of such functions based on axioms of measurement. Stevens identified four basic scale types: *nominal*, *ordinal*, *interval*, and *ratio*.

To define a nominal scale, we assume there exists at least one equivalence class together with a binary equivalence relation ($\sim$) that can be applied to objects in the domain (e.g., the class of this object is the *same* as the class of that object). For a domain of objects D and a set of values $X(d), d \in D$, we say that a scale is nominal if

$$d_i \sim d_j \iff X(d_i) = X(d_j),\ \forall\ d_i, d_j \in D\,.$$

To define an ordinal scale, we assume there exists a binary total order relation ($\succ$) that can be applied to objects in the domain (e.g., this stone is *heavier than* that stone). We then say that a scale is ordinal if

$$d_i \succ d_j \iff X(d_i) > X(d_j),\ \forall\ d_i, d_j \in D\,.$$

To define an interval scale, we assume there exists a symmetric concatenation operation ($\oplus$) that can be applied to objects in the domain (e.g., the length of this stick *appended to* the length of that stick). We then say that a scale is interval if

$$d_i \oplus d_j \sim d_k \iff X(d_i) + X(d_j) = X(d_k),\ \forall\ d_i, d_j, d_k \in D\,.$$

To define a ratio scale, we assume there exists a magnitude comparison operation ($\oslash$) that can be applied to objects in the domain (e.g., the *ratio* of the brightness of this patch to the the brightness of that patch). We then say that a scale is ratio if

$$d_i \oslash d_j \sim d_k \iff X(d_i)/X(d_j) = X(d_k),\ \forall\ d_i, d_j, d_k \in D\,.$$

Axiomatic scale theory is often invoked by practitioners of data mining and graphics, but it is not sufficient for determining scales on statistical graphics produced by chart algebra. The blend operation, for example, allows us to union values on different variables. We can require that blended variables share the same measurement level (e.g., diastolic and systolic blood pressure), but this will not always produce a meaningful scale. For example, we will have a meaningless composite scale if we attempt to blend height and weight, both presumably ratio variables. We need a different level of detail so that we can restrict the blend operation more appropriately.

13.4.2 Unit Measurement

An alternative scale classification is based on units of measurement. Unit scales permit standardization and conversion of metrics. In particular, the International System of Units (SI) (Taylor 1997) unifies measurement under transformation rules encapsulated in a set of base classes. These classes are *length*, *mass*, *time*, *electric current*, *temperature*, *amount of substance*, and *luminous intensity*. Within the base classes, there are default metrics (meter, kilogram, second, etc.) and methods for

Table 13.2 Typical unit measurements

Length	Mass	Temperature	Time	Volume	Currency
meter	kilogram	kelvin	second	liter	dollar
point	gram	rankine	minute	teaspoon	euro
pica	grain	celsius	hour	tablespoon	pound
inch	slug	fahrenheit	day	cup	yen
foot	carat		week	pint	rupee
yard			month	quart	dinar
mile			quarter	gallon	
furlong			year	bushel	
fathom			century	barrel	

converting from one metric to another. From these base classes, a set of derived classes yields measurements such as *area*, *volume*, *pressure*, *energy*, *capacitance*, *density*, *power*, and *force*. Table 13.2 shows some examples of several SI base classes, derived classes, and an example of an economic base class that is not in SI. The *currency* class is time dependent, since daily exchange rates determine conversion rules and an inflation adjustment method varies with time.

Most of the measurements in the SI system fit within the interval and ratio levels of Stevens' system. There are other scales fitting Stevens' system that are not classified within the SI system. These involve units such as *category* (state, province, country, color, species), *order* (rank, index), and *measure* (probability, proportion, percent). And there are additional scales that are in neither the Stevens nor the SI system, such as *partial order*.

For our purposes, unit measurement gives us the level of detail needed to construct a numerical or categorical scale. We consider unit measurement a form of strong typing that enables reasonable default behavior. Because of the class structure and conversion methods, we can handle labels and relations for derived quantities such as miles-per-gallon, gallons-per-mile, and liters-per-kilometer. Furthermore, automatic unit conversion within base and derived classes allows meaningful blends. As with domain check overrides in a database (Date 1990), we allow explicit type overrides for the blend operation.

13.4.3 Transformations

We frequently compute transformations of variables in constructing graphics. Sometimes, we employ statistical transformations to achieve normality so that we can apply classical statistical methods such as linear regression. Other times, we transform to reveal local detail in a graphic. It helps to apply a log transform, for example, to stretch out small data values in a display. We might do this even when not applying statistical models.

These types of transformations fall within the scale stage of the grammar of graphics system. Because GOG encapsulates variable transformations within this

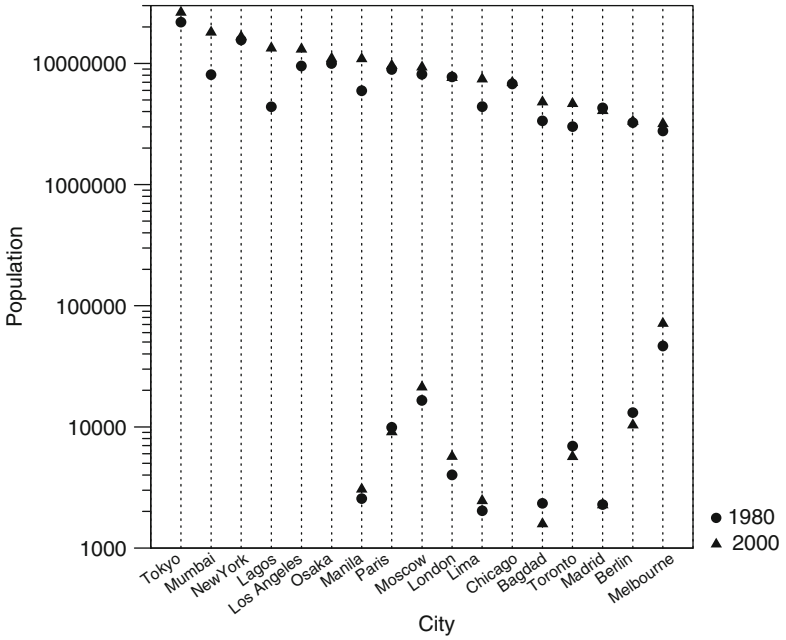


Fig. 13.7 `city` $*$ (`pop1980` + `pop2000`), *ylog*

stage, it accomplishes two tasks at the same time: 1) the values of the variables are transformed prior to analysis and display, and 2) nice scale values for axes and legends are computed based on the transformation. Figure 13.7 shows an example of this process for the city data. In order to highlight population changes in small cities, we represent the populations on a log scale. The algebraic expression is the same as in Fig. 13.5: `city` $*$ (`pop1980` + `pop2000`). Now we see that most of the cities gained population between 1980 and 2000 but half the US namesakes lost population.

13.5 Statistics

Visualization and statistics are inseparable. Statisticians have known this for a long time, but non-statisticians in the visualization field have largely ignored the role of statistics in charts, maps, and graphics. Non-statisticians often believe that visualization follows data analysis. We aggregate, summarize, model, and *then* display the results. In this view, visualization is the last step in the chain and statistics is the first.

In GOG, statistics falls in the middle of the chain. The consequence of this architecture is that statistical methods are an integral part of the system. We can

construct dynamic graphics, in which statistical methods can be changed (for exploratory purposes) without altering any other part of the specification and without restructuring the data. By including statistical methods in its architecture, GOG also makes plain the independence of statistical methods and geometric displays. There is no necessary connection between regression methods and curves or between confidence intervals and error bars or between histogram binning and histograms.

In GOG, the statistics component receives a varset, computes various statistics, and outputs another varset. In the simplest case, the statistical method is an identity. We do this for scatterplots. Data points are input and the same data points are output. In other cases, such as histogram binning, a varset with n rows is input and and a varset with k rows is output, where k is the number of bins ($k < n$). With smoothers (regression or interpolation), a varset with n rows is input and and a varset with k rows is output, where k is the number of knots in a mesh over which smoothed values are computed. With point summaries (means, medians, ...), a varset with n rows is input and a varset with one row is output. With regions (confidence intervals, ranges, ...), a varset with n rows is input and and a varset with two rows is output.

Understanding how the statistics component works reveals an important reason for mapping values to cases in a varset rather than the other way around. If

$$\mathrm{A} = [\mathbb{R}, \{\langle\cdot\rangle, \langle\cdot,\cdot\rangle, \ldots\}, \{1.5 \to \langle 1\rangle, 2.7 \to \langle 2\rangle, 1.8 \to \langle 3\rangle\}] \,,$$

then

$$\mathrm{mean(A)} = [\mathbb{R}, \{\langle\cdot\rangle, \langle\cdot,\cdot\rangle, \ldots\}, \{2.0 \to \langle 1, 2, 3\rangle\}] \,.$$

Notice that the list of caseIDs that is produced by *mean*() is contained in the one row of the output varset. We do not lose case information in this mapping, the way we do when we compute results from an ordinary SQL query on a database or when we compute a data cube for an OLAP or when we pre-summarize data to produce a simple graphic. This aspect of GOG is important for dynamic graphics systems that allow *drill-down* or queries regarding metadata when the user hovers over a particular graphic element.

Figure 13.8 shows an application of a statistical method to the city data. We linearly regress 2000 population on 1980 population to see if population growth is proportional to city size. On log-log scales, the estimated values fall on a line whose slope is greater than 1, suggesting that larger cities grow faster than smaller. Ordinarily, we would draw a line to represent the regression and we would include the data points as well. We would also note that Lagos grew at an unusual rate (with a Studentized residual of 3.4). Nevertheless, our main point is to show that the statistical regression produces data points that are exchangeable with the raw data insofar as the entire GOG system is concerned. How we choose to represent the regressed values graphically is the subject of the next section.

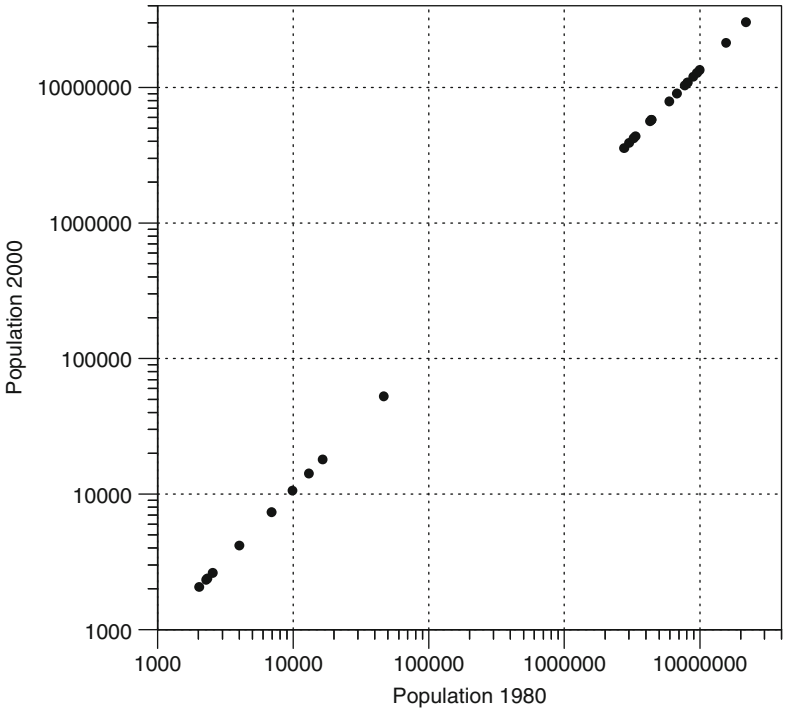


Fig. 13.8 `pop1980` ∗ *estimate*(`pop2000`), *xlog*, *ylog*

13.6 Geometry

GOG presumes no connection between a statistical method and a geometric representation. Histogram bins need not be represented by histograms. Tukey schematic plots (his original word for box plots) need not be represented by boxes and whiskers. Regressions need not be represented by lines or curves. Separating geometry from data (and from other graphical aspects such as coordinate systems) is what gives GOG its expressive power. We choose geometric representation objects independently of statistical methods, coordinate systems, or aesthetic attributes.

As Fig. 13.1 indicates, the geometry component of GOG receives a varset and outputs a geometric graph. A geometric graph is a subset of $\mathbb{R}^n$. For our purposes, we will be concerned with geometric graphs for which $1 \leq n \leq 3$. Geometric graphs are enclosed in bounded regions:

$$B^n \subset [a_1, b_1] \times \ldots \times [a_n, b_n]$$

These intervals define the edges of a bounding box or region in n-dimensional space. There are two reasons we need bounded regions. First, in order to define certain

useful geometric graphs, we need concepts like the *end* of a line or the *edge* of a rectangle. Second, we want to save ink and electricity. We don't want to take forever to compute and draw a line.

Geometric graphs are produced by graphing functions $F : B^n \rightarrow \mathbb{R}^n$ that have geometric names like *line*() or *tile*(). A geometric graph is the image of F. And a graphic, as used in the title of this chapter, is the image of a graph under one or more aesthetic functions. Geometric graphs are not visible. As Bertin (1967) points out, visible elements have features not present in their geometric counterparts.

Figures 13.9 and 13.10 illustrate the exchangeability of geometry and statistical methods. The graphics are based on UN data involving 1990 estimates of female life expectancy and birth rates for selected world countries. Figure 13.9 shows four different geometric graphs – *point*, *line*, *area*, and *bar* – used to represent a confidence interval on a linear regression. Figure 13.10 shows one geometric

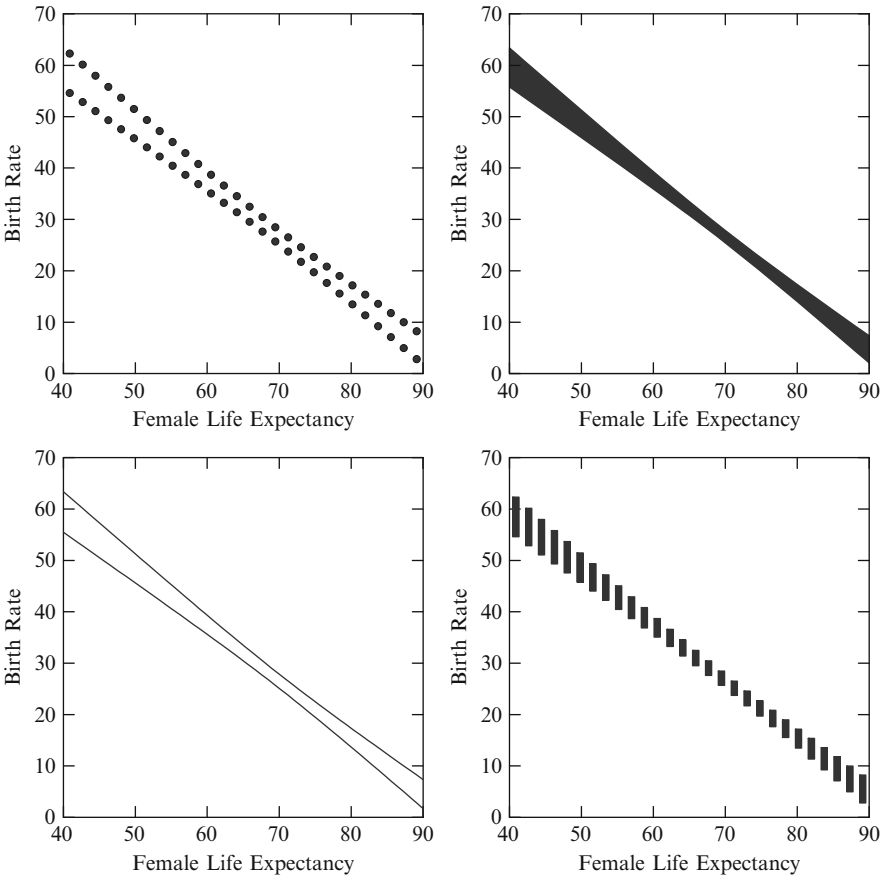


Fig. 13.9 Different graph types, same statistical method

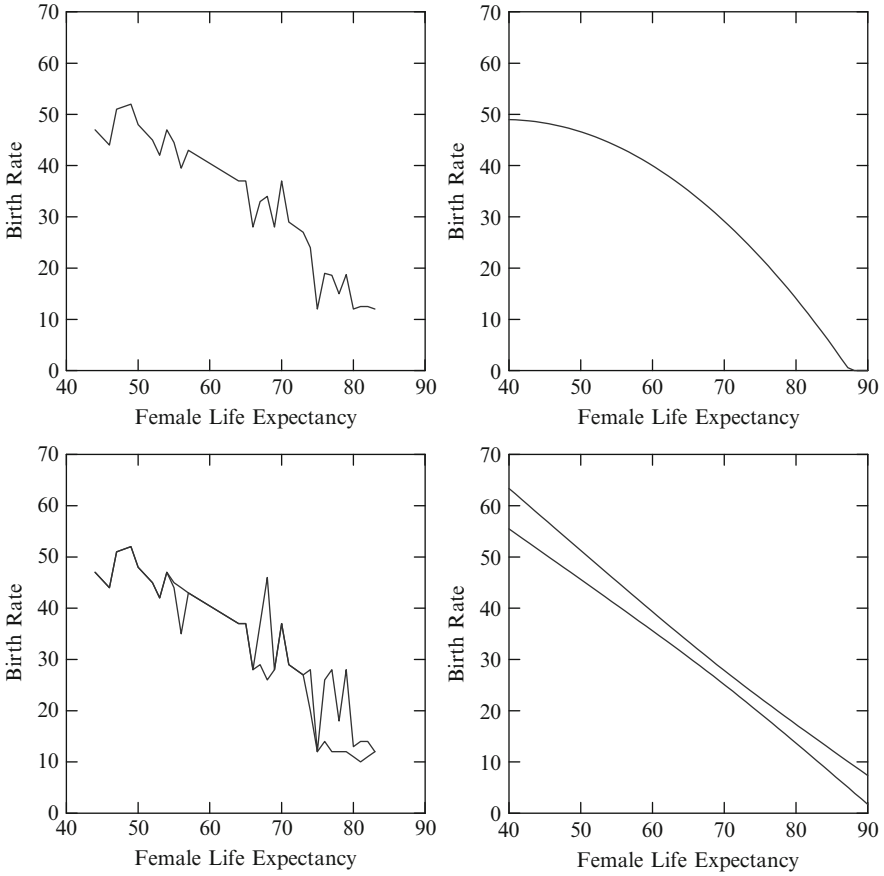


Fig. 13.10 Different statistical methods, same graph type

graph used to represent four different statistical methods – local mean, local range, quadratic regression, and linear regression confidence interval.

This exchangeability produces a rich set of graphic forms with a relatively small number of geometric graphs. Table 13.3 contains these graphing methods. The *point*() graphing function produces a geometric point, which is an n-tuple. This function can also produce a finite set of points, called a *multipoint* or a *point cloud*. The set of points produced by *point*() is called a *point* graph.

The *line*() graphing function function is a bit more complicated. Let B^m be a bounded region in $\mathbb{R}^m$. Consider the function $F : B^m \to \mathbb{R}^n$, where $n = m + 1$, with the following additional properties:

1. The image of F is bounded, and
2. $F(x) = (\mathbf{v}, f(\mathbf{v}))$, where $f : B^m \to \mathbb{R}$ and $\mathbf{v} = (x_1, \ldots, x_m) \in B^m$.

Table 13.3 Geometric graphs

Relations	Summaries	Partitions	Networks
Point	*Schema*	*Tile*	*Path*
Line (*surface*)		*Contour*	*Link*
Area (*volume*)			
Bar (*interval*)			
Histobar			

If $m = 1$, this function maps an interval to a functional curve on a bounded plane. And if $m = 2$, it maps a bounded region to a functional surface in a bounded 3D space. The *line*() graphing function produces these graphs. Like *point*(), *line*() can produce a finite set of lines. A set of lines is called a multiline. We need this capability for representing multimodal smoothers, confidence intervals on regression lines, and other multifunctional lines.

The *area*() graphing function produces a graph containing all points within the region under the *line* graph. The *bar*() graphing function produces a set of closed intervals. An interval has two ends. Ordinarily, however, bars are used to denote a single value through the location of one end. The other end is anchored at a common reference point (usually zero). The *histobar*() graphing function produces a histogram element. This element behaves like a bar except a value maps to the area of a histobar rather than to its extent. Also, histobars are glued to each other. They cover an interval or region, unlike bars.

A *schema* is a diagram that includes both general and particular features in order to represent a distribution. We have taken this usage from Tukey (1977), who invented the schematic plot, which has come to be known as the box plot because of its physical appearance. The *schema*() graphing function produces a collection of one or more points and intervals.

The *tile*() graphing function tiles a surface or space. A *tile* graph covers and partitions the bounded region defined by a frame; there can be no gaps or overlaps between tiles. The Latinate *tessellation* (for tiling) is often used to describe the appearance of the tile graphic.

A *contour*() graphing function produces contours, or level curves. A *contour* graph is used frequently in weather and topographic maps. Contours can be used to delineate any continuous surface.

The *network*() graphing function joins points with line segments (edges). Networks are representations that resemble the edges in diagrams of theoretic graphs. Although networks join points, a point graph is not needed in a frame in order for a network graphic to be visible.

Finally, the *path*() graphing function produces a *path* that connects points such that each point touches no more than two line segments. Thus, a path visits every point in a collection of points only once. If a path is closed (every point touches two line segments), we call it a circuit. Paths often look like lines. There are several important differences between the two, however. First, lines are functional; there can be only one point on a line for any value in the domain. Paths may loop, zigzag,

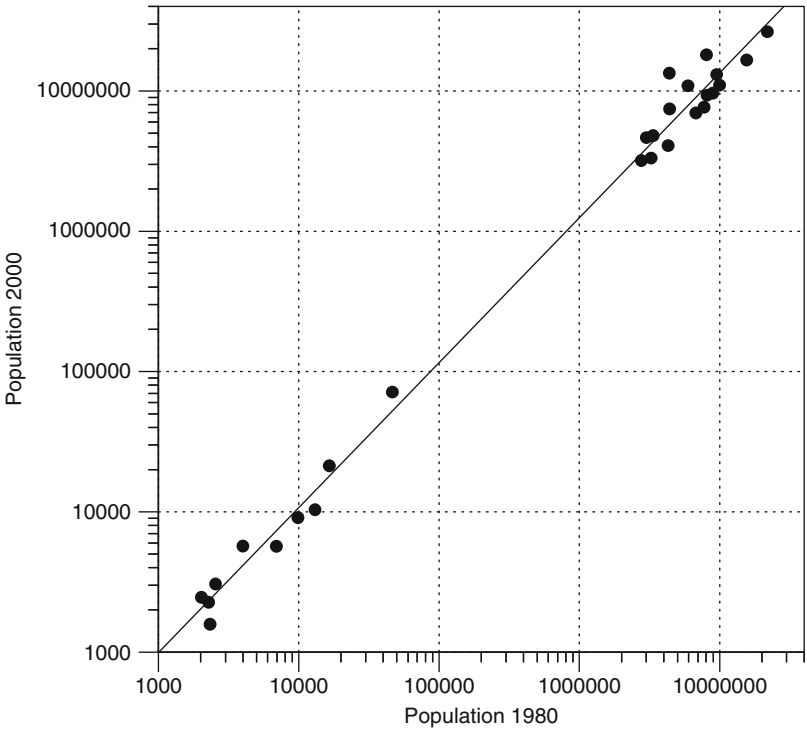


Fig. 13.11 `pop1980` $*$ {`pop2000`, $estimate$(`pop2000`)}, *xlog*, *ylog*, {*point*, *line*}

and even cross themselves inside a frame. Second, paths consist of segments that correspond to edges, or links between nodes. This means that a variable may be used to determine an attribute of every segment of a path.

Figure 13.11 contains two geometric objects for representing the regression we computed in Fig. 13.8. We use a *point* for representing the data and a *line* for representing the regression line.

13.7 Coordinates

The most popular types of charts employ Cartesian coordinates. The same real tuples in the graphs underlying these graphics can be embedded in many other coordinate systems, however. There are many reasons for displaying graphics in different coordinate systems. One reason is to simplify. For example, coordinate transformations can change some curvilinear graphics to linear. Another reason is to reshape graphics so that important variation or covariation is more salient or accurately perceived. For example, a pie chart is generally better for judging proportions of wholes than is a bar chart (Simkin and Hastie 1987). Yet another

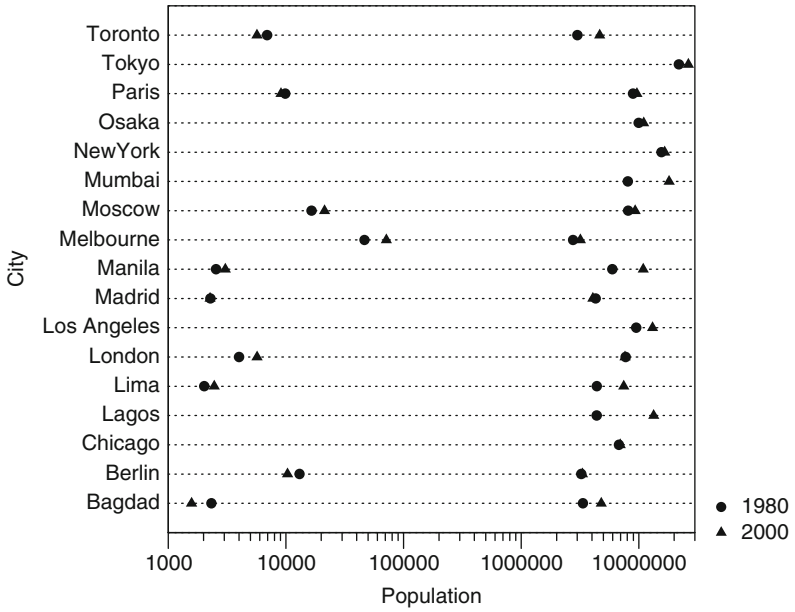


Fig. 13.12 *transpose*(`city * (pop1980 + pop2000)`), *ylog*

reason is to match the form of a graphic to theory or reality. For example, we might map a variable to the left-closed and right-open interval $[0, 1)$ on a line or to the interval $[0, 2\pi)$ on the circumference of a circle. If our variable measures defects within a track of a computer disk drive in terms of rotational angle, it is usually better to stay within the domain of a circle for our graphic. Another reason is to make detail visible. For example, we may have a cloud with many points in a local region. Viewing those points may be facilitated by zooming in (enlarging a region of the graphic) or smoothly distorting the local area so that the points are more separated in the local region.

Wilkinson (1999) contains many examples of ordinary charts rendered in different coordinate systems. A simple example suffices for the data in this chapter. Figure 13.12 shows a transposed version of Fig. 13.7. The result of this coordinate transformation (a rotation composed with a reflection) is to make the city names more readable.

13.8 Aesthetics

The term aesthetics derives from a Greek word that means perception. The derivative modern meanings of beauty, taste, and artistic criteria arose in the eighteenth century. We have chosen the name *aesthetics* to describe the class of functions that turn theoretical graphs into perceivable graphics because of its original connotations and because the modern word *perception* is subjective rather than objective;

Table 13.4 Aesthetics

Form	Surface	Motion	Sound	Text
Position	*Color*	*Direction*	*Tone*	*Label*
Size	*Texture*	*Speed*	*Volume*	
Shape	*Blur*	*Acceleration*	*Rhythm*	
Rotation	*Transparency*		*Voice*	

perception refers to the perceiver rather than the object. Aesthetics turn graphs into graphics so that they are perceivable, but they are not the perceptions themselves. A modern psychologist would most likely call aesthetics in this sense stimuli, aspects, or features, but these words are less germane to our purpose.

Table 13.4 summarizes these aesthetic attributes. We have grouped these attributes in five categories: *form*, *surface*, *motion*, *sound*, and *text*. This is not intended to be an exhaustive list; other attributes, such as *odor*, can be devised. The *color* aesthetic has three components: *hue*, *brightness*, and *saturation* (other color components are possible). The *texture* aesthetic includes components of *pattern*, *granularity*, and *orientation*.

Seven of these attributes are derived from the *visual variables* of Bertin (1967): *position* (position), *size* (taille), *shape* (forme), *orientation* (orientation), *brightness* (valeur), *color* (couleur), and *granularity* (grain). Bertin's *grain* is often translated as *texture*, but he really means granularity (as in the granularity of a photograph). Granularity in this sense is also related to the spatial frequency of a texture.

These aesthetic attributes do not represent the aspects of perception investigated by psychologists. This lack of fit often underlies the difficulty graphic designers and computer specialists have in understanding psychological research relevant to graphics and the corresponding difficulty psychologists have with questions asked by designers. Furthermore, these attributes are not ones customarily used in computer graphics to create realistic scenes. They are not even sufficient for a semblance of realism. Notice, for example, that *pattern*, *granularity*, and *orientation* are not sufficient for representing most of the textures needed for representing real objects. Instead, these attributes are chosen in a tradeoff between the psychological dimensions they elicit and the types of routines that can be implemented in a rendering system. Specifically:

- An attribute must be capable of representing both continuous and categorical variables.
- When representing a continuous variable, an attribute must vary primarily on one psychophysical dimension. In order to use multidimensional attributes such as color, we must scale them on a single dimension such as hue or brightness, or compute linear or nonlinear combinations of these components to create a unidimensional scale.
- An attribute does not imply a linear perceptual scale. In fact, few aesthetic attributes scale linearly. Some attributes such as hue scale along curvilinear segments in two- or three-dimensional space. All linear scales are unidimensional but not all unidimensional scales are linear.

- A perceiver must be able to report a value of a variable relatively accurately and effortlessly when observing an instance of that attribute representing that variable.
- A perceiver must be able to report values on each of two variables relatively accurately upon observing a graphic instantiating two attributes. This task usually, but not necessarily, requires selective attention. This criterion probably isn't achievable for all of our attributes and may not even be achievable for any pair of them. But any attribute that is clearly non-separable with another should be rejected for our system. It is too much to expect, of course, that higher order interactions among attributes be non-existent. Much of the skill in graphic design is knowing what combinations of attributes to avoid.
- Each attribute must name a distinct feature in a rendering system. We cannot implement an attribute that does not uniquely refer to a drawable (or otherwise perceivable) feature. An attribute cannot be mapped to a miscellaneous collection of widgets or controls, for example.

We have attempted to classify aesthetics so that they are orthogonal in a design sense. One must not assume that this implies they are uncorrelated in our perceptual mechanisms, however. Orthogonalization in design means making every dimension of variation that is available to one object available to another. How these variations are perceived is another matter. Many aesthetic attributes, even ones such as *size* or *position* that are usually considered visual, need not be perceived visually. There is nothing in the definition of a graphic to limit it to vision. Provided we use devices other than computer screens and printers, we can develop graphical environments for non-sighted people or for those unable to attend to a visual channel because, perhaps, they are busy, restrained, or multiprocessing. Touch, hearing, even smell can be used to convey information with as much detail and sensitivity as can vision.

Every one of the figures in this chapter incorporates several aesthetics. Without aesthetic functions, they would not be visible. Consequently, we will not add a figure to illustrate other aesthetics, particularly since we are constrained in publishing format. Note, however, that in addition to using the *position* aesthetic function in every graphic, we have employed *shape* to differentiate symbols. Note, also, that *position* aesthetics are usually referenced by axes and *shape* and other aesthetics are usually referenced by legends.

Our discussion of the seven primary GOG components ends here. But there are several important topics remaining. We will first examine issues in graphics layout, and then conclude with a discussion of the relation between graphics algebra and statistical design models.

13.9 Layout

Chart algebra does not determine the physical appearance of charts plotted on a screen or paper. It simply produces a set of tuples $(x_1, \ldots, x_p)$ that can be rendered using geometric primitives and a layout interpreter. If we have 2-tuples, then we can

render them directly on a computer screen or piece of paper. If we have 3-tuples, then we can use a perspective projection to render them on the plane. Higher-order tuples require a layout scheme to embed all dimensions in the plane. Various layout schemes are attempts to solve a graphic representation problem: how to transform a p-dimensional vector space to a 2-dimensional space so that we can perceive structures in the higher dimensional space by examining the 2-dimensional space. We will discuss several approaches in this section.

Projection

One scheme is to employ a linear or nonlinear projection from p-dimensions to two. This may cause loss of information because a projection onto a subspace is many-to-one. Also, projection is most suitable for displaying points or $\{V, E\}$ graphs. It is less suitable for many geometric chart types such as bars and pies. Nevertheless, some 2D projections have been designed to capture structures contained in subspaces, such as manifolds, simplices, or clusters (Friedman 1987). Other popular projection methods are principal components and multidimensional scaling (Hastie et al. 2001).

Sets of Functions

A second possibility is to map a set of n points in $\mathbb{R}^p$ one-to-one to a set of n functions in $\mathbb{R}^2$. A particularly useful class of functions is formed by taking the first p terms in a Fourier series as coefficients for $(x_1, \ldots, x_p)$ (Andrews 1972). Another useful class is the set of Chebysheff orthogonal polynomials. A popular class is the set of $p - 1$ piecewise linear functions with $(x_1, \ldots, x_p)$ as knots, often called *parallel coordinates* (Inselberg 1984; Wegman 1985).

An advantage of function space representations is that there is no loss of information, since the set of all possible functions for each of these types in $\mathbb{R}^2$ is infinite. Orthogonal functions (such as Fourier and Chebysheff) are useful because zero inner products are evidence of linear independence. Parallel coordinates are useful because it is relatively easy to decode values on particular variables. A disadvantage of functional representations is that manifolds, solids, and distances are difficult to discern.

Recursive Partitioning

A third possibility is recursive partitioning. We choose an interval $[u_1, u_2]$ and partition the first dimension of $\mathbb{R}^p$ into a set of connected intervals of size $(u_2 - u_1)$, in the same manner as histogram binning. This yields a set of rectangular subspaces of $\mathbb{R}^p$. We then partition the second dimension of $\mathbb{R}^p$ similarly. This second partition produces a set of rectangular subspaces within each of the previous subspaces. We continue choosing intervals and partitioning until we finish the last dimension. We then plot each subspace in an ordering that follows the ancestry of the partitioning. Recursive partitioning layout schemes have appeared in many guises: $\mathbb{R}^p \mapsto \mathbb{R}^3$

(Feiner and Beshers 1990), $\mathbb{R}^p \mapsto \mathbb{R}^2$ (Mihalisin et al. 1991), $\mathbb{R}^4 \mapsto \mathbb{R}^2$ (Becker et al. 1996).

There are several modifications we may make to this scheme. First, if a dimension is based on a categorical variable, then we assume $(u_2 - u_1) = 1$, which assures one partition per category. Second, we need not partition a dimension into equal intervals; instead, we can make $[u_1, u_2]$ adaptive to the density of the data (Wilkinson 1999, page 186). Third, we can choose a variety of layouts for displaying the nodes of the partitioning tree. We can display the cells as an n-ary tree, which is the method used by popular decision-tree programs. Or, we can alternate odd/even dimensions by plotting horizontally/vertically. This display produces a 2D nested table, which has been variously named a *mosaic* (Hartigan and Kleiner 1981) or *treemap* (Johnson and Schneiderman 1991). We use this latter scheme for the figures in this article.

This rectangular partitioning resembles a 2D rectangular fractal generator. Like simple projection, this method can cause loss of information because aggregation occurs within cells. Nevertheless, it yields an interpretable 2D plot that is familiar to readers of tables.

Because recursive partitioning works with either continuous or categorical variables, there is no display distinction between a table and a chart. This equivalence between tables and graphs has been noted in other contexts (Pedersen et al. 2002; Shoshani 1997). With recursive partitioning, we can display tables of charts and charts of tables.

13.10 Analytics

If conclusions based on statistical charts are to be useful, we must identify and interpret the statistical models underlying charts. A statistical model determines how the location of a representation element (point, line, ...) in a frame (a measurable region of representation) is computed from the values of a variable. Statistical models usually (but not necessarily) incorporate error terms and help us to articulate the domains of generalizations and inferences we make from examining a chart. Glymour et al. (1996) summarize these issues from a data mining context. Because chart algebra is based on statistical design algebras, it can be used to generate statistical models for visual data mining or predictive analytics.

This section presents the statistical model equivalents of chart algebra expressions. In each subsection, we show the chart algebra notation on the left of each equivalence expression and the statistical model notation on the right. The terms on the left comprise varsets and the terms on the right comprise variables. Note that some symbols (e.g., +) are common to both notations but have different meanings. The general linear statistical models presented in this section are due to (Fisher 1925, 1935). More recent introductions to the design notation used for statistical models are (Heiberger 1989) and (Kutner et al. 1996).

13.10.1 Statistical Model Equivalents

In the following subsections, we assume a functional model $Z = f(X, Y)$, where Z is a (possibly multivariate) variable. Z corresponds to a varset Z, which itself might be produced from a chart algebra expression. In statistical terms, we sometimes call Z a *dependent* variable and X and Y *independent* variables. In this section, we ignore Z and focus on expressions involving X and Y. These expressions are used to construct statistical models that help to predict or estimate Z.

Cross

$$\mathrm{X} * \mathrm{Y} \sim \mathcal{C} + X + Y + XY$$

The cross operator corresponds to a fully factorial experimental design specification. This design employs a product set that includes every combination of levels of a set of experimental factors or treatments. The terms on the right of the similarity represent the linear model for fitting fully factorial designs. The terms in the model are:

$\mathcal{C}$: constant term (grand mean)

X : levels of factor X (X main effect)

Y : levels of factor Y (Y main effect)

XY : product of factors X and Y(interactions)

We could use boldface for the variables on the right because the most general form of the model includes factors (multidimensional categorical variables) having more than one level. These multivariate terms consist of sets of binary categorical variables whose values denote presence or absence of each level in the factor. Alternatively, terms based on continuous variables are called *covariates*.

An example of a two-way factorial design would be the basis for a study of how teaching method and class size affect the job satisfaction of teachers. In such a design, each teaching method (factor X) is paired with each class size (factor Y) and teachers and students in a school are randomly assigned to the combinations.

Nest

$$\mathrm{X}/\mathrm{Y} \sim \mathcal{C} + Y + X(Y)$$

The terms on the right of the similarity are:

$\mathcal{C}$: constant term

Y : levels of factor Y (Y main effect)

$$X(Y) : \text{X levels nested within levels of Y}$$

The term $X(Y)$ represents the series $X \mid (Y = Y_1) + X \mid (Y = Y_2) + \ldots$

Notice that there is no interaction term involving X and Y because X is nested within Y. Not all combinations of the levels of X and Y are defined. An example of a nested design would be the basis for a study of the effectiveness of different teachers and schools in raising reading scores. Teachers are nested within schools when no teacher in the study can teach at more than one school. With nesting, two teachers with the same name in different schools are different people. With crossing, two teachers with the same name in different schools may be the same person.

Blend

$$\text{X} + \text{Y} \sim \mathcal{C} + F_{XY}$$

The terms on the right of the similarity are:

$$\mathcal{C} : \text{constant term}$$

$$F_{XY} : \text{function of } X \text{ and } Y \text{ (e.g., } X - Y)$$

The blend operator usually corresponds to a time series design. In such a design, we predict using functions of a time series. When the blend involves dependent variables, this is often called a repeated measures design. The simplest case is a prediction based on first differences of a series. Time is not the only possible dimension for ordering variables, of course. Other multivariate functional models can be used to analyze the results of blends (Ramsay and Silverman 1997).

An example of a repeated measures design would be the basis for a study of improvement in reading scores under different curricula. Students are randomly assigned to curriculum and the comparison of interest involves differences between pre-test and post-test reading scores.

It would appear that analytics have little to do with the process of building a chart. If visualization is at the end of a data-flow pipeline, then statistics is simply a form of pre-processing. In our model, however, analytics are an intrinsic part of chart construction. Through chart algebra, the structure of a graph implies a statistical model. Given this model, we can employ likelihood, information, or goodness-of-fit measures to identify parsimonious models. We will explore some graphic uses of statistical models in this section.

13.10.2 Subset Model Fitting

The factorial structure of most chart algebra expressions can produce rather complex models. We need to consider strategies for selecting subset models that are adequate

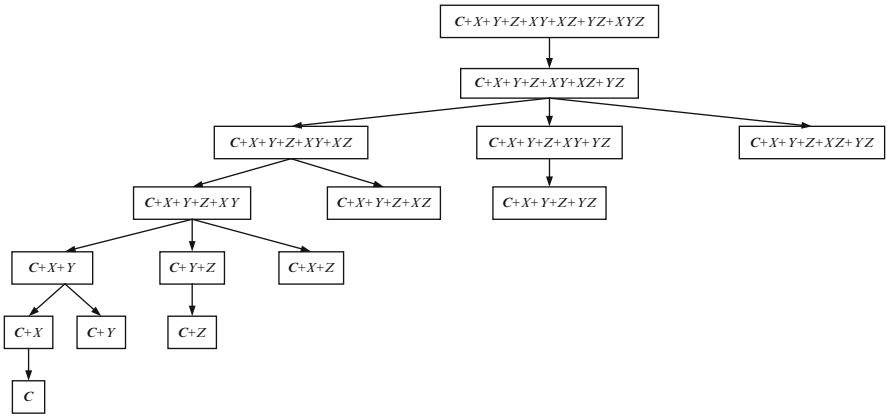


Fig. 13.13 Model subset tree

fits to the data. We will discuss one simple approach in this section. This approach involves eliminating interactions (products of factors) in factorial designs.

Interactions are often regarded as nuisances because they are difficult to interpret. Comprehending interactions requires thinking about partial derivatives. A three-way interaction XYZ, for example, means that the relation between X and Y depends on the level of Z. And the relation between X and Z depends on the level of Y. And the relation between Y and Z depends on the level of X. Without any interaction, we can speak about these simple relations unconditionally. Thus, one strategy for fitting useful subset models is to search for subsets with as few interactions as possible. In this search, we require that any variables in an interaction be present as a main-effect in the model.

Figure 13.13 shows a submodel tree for the three-way crossing $X * Y * Z$. Not all possible submodels are included in the tree, because the convention in modeling factorial designs is to include main effects for every variable appearing in an interaction. This reduces the search space for plausible submodels. By using branch-and-bound methods, we can reduce the search even further. Mosteller and Parunak (1985) and Linhart and Zucchini (1986) cover this area in more detail.

13.10.3 Lack of Fit

Statistical modeling and data mining focus on regularity: averages, summaries, smooths, and rules that account for the significant variation in a dataset. Often, however, the main interest of statistical graphics is in locating aspects that are discrepant, surprising, or unusual: under-performing sales people, aberrant voting precincts, defective parts.

An *outlier* is a case whose data value and fitted value (using some model) are highly discrepant relative to the other data-fitted discrepancies in the dataset. (Barnett and Lewis 1994). Casewise discrepancies are called *residuals* by statisticians. Outliers can be flagged in a display by highlighting (e.g., coloring) large residuals in the frame. Outliers are only one of several indicators of a poorly fit model, however. Relative badness-of-fit can occur in one or more cells of a table, for example. We can use subset modeling to highlight such cells in a display. Sarawagi et al. (1998) do this for log-linear models. Also, we can use autocorrelation and cross-correlation diagnostic methods to identify dependencies in the residuals and highlight areas in the display where this occurs.

13.10.4 Scalability

Subset design modeling is most suited for deep and narrow (many rows, few columns) data tables or low-dimensional data cubes. Other data mining methods are designed for wide data tables or high-dimensional cubes (Hand et al. 2001; Hastie et al. 2001). Subset design modeling makes sense for visualization applications because the design space in these applications does not tend to be high-dimensional. Visual data exploration works best in a few dimensions. Higher-dimensional applications work best under the guidance of other data mining algorithms.

Estimating design models requires $O(n)$ computations with regard to cases, because only one pass through the cases is needed to compute the statistics for estimating the model. Although computing design models can be worse-case $O(p^2)$ in the number of dimensions, sparse matrix methods can be used to reduce this overhead because many of the covariance terms are usually zero.

13.10.5 An Example

Smoothing data reveals systematic structure. Tukey (1977) used the word in a specific sense, by pairing the two equations

$$data = fit + residual$$

$$data = smooth + rough$$

Tukey's use of the word is different from other mathematical meanings, such as functions having many derivatives.

We smooth data in graphics to highlight selected patterns in order to make inferences. We present an example involving injury to the heads of dummies in government frontal crash tests. Figure 13.14 shows NHTSA crash test results for selected vehicles tested before 1999. The dependent variable shown on the

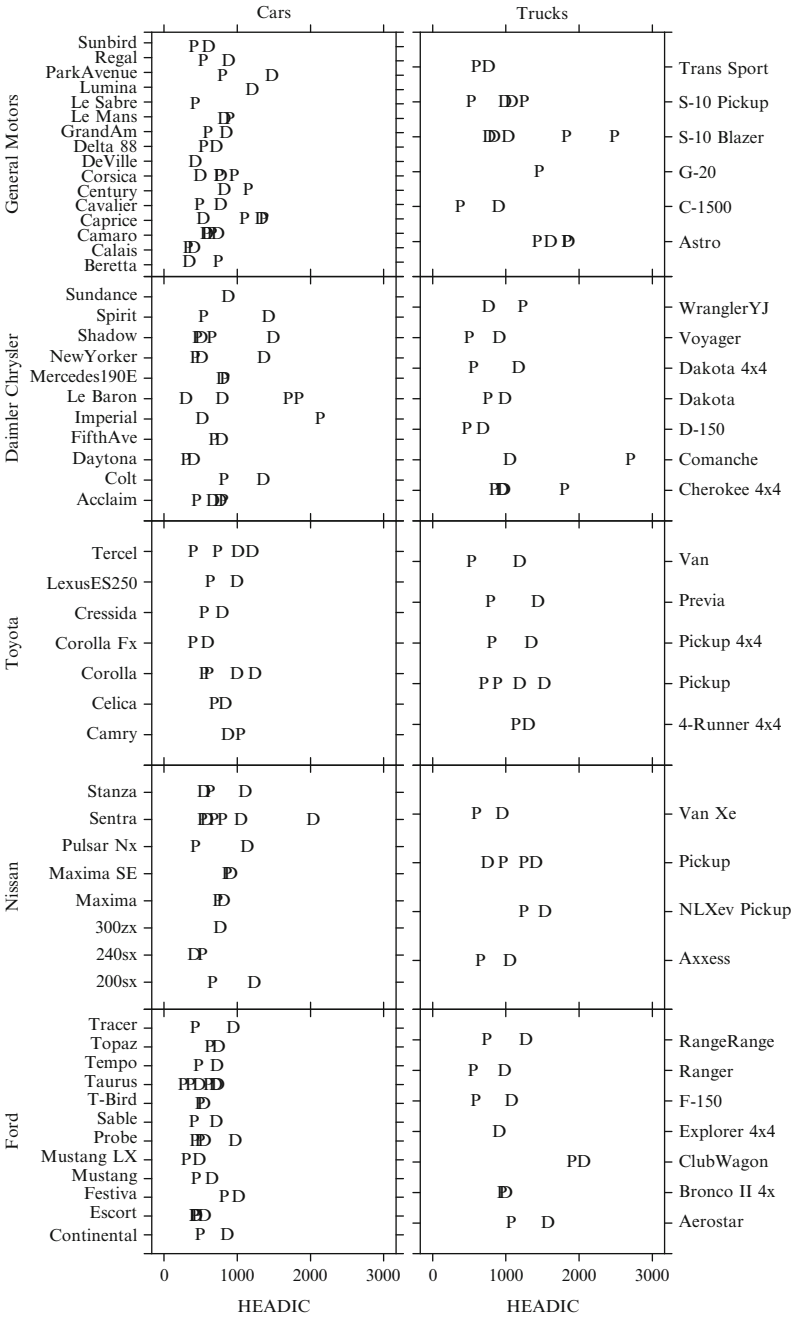


Fig. 13.14 Crash data

horizontal axis of the chart is the Head Injury Index computed by the agency. The full model is generated by the chart algebra $H * T/(M * V) * O$. This expression corresponds to the model:

$$H = \mathcal{C} + M + V + O + T(MV) + MV + MO + VO + OT(MV) + MVO$$

where the symbols are:

H : Head Injury Index

$\mathcal{C}$: constant term (grand mean)

M : Manufacturer

V : Vehicle (car/truck)

O : Occupant (driver/passenger)

T : Model

This display is difficult to interpret. We need to fit a model and order the display to reveal the results of the model fit. Figure 13.15 charts fitted values from the following subset model:

$$H = \mathcal{C} + V + O + T(MV)$$

Figure 13.15 has several notable features. First, the models are sorted according to the estimated Head Injury Index. This makes it easier to compare different cells. Second, some values have been estimated for vehicles with missing values (e.g., GM G-20 driver data). Third, the trends are smoother than the raw data. This is the result of fitting a subset model. We conclude that passengers receive more head injuries than drivers, occupants of trucks and SUVs (sports utility vehicles) receive more head injuries than occupants of cars, and occupants of some models receive more injuries than occupants of others.

13.11 Software

Four systems have so far implemented the algebra described in this chapter. Wilkinson et al. (2000) developed a system called nViZn, which used the algebra to present interactive graphics in a Web environment. Norton et al. (2001) used the algebra to structure and display data in a real-time streaming environment; their system is called Dancer. Wills (2002) developed a server-based presentation graphics system with an XML schema for GOG; this system is called. ViZml. Stolte et al. (2002) used the algebra to develop a visualization front-end for an OLAP; their system is called Polaris.

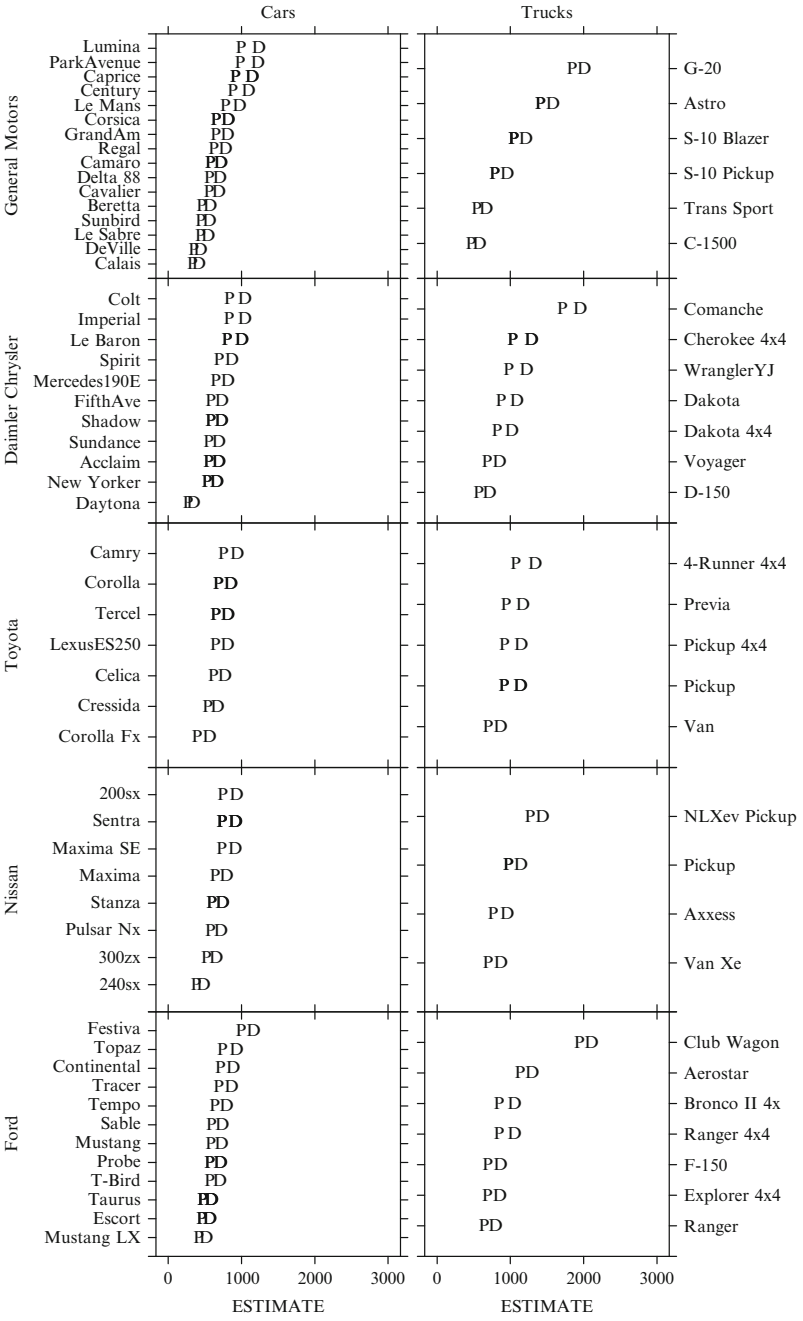


Fig. 13.15 Smoothed crash data

13.12 Conclusion

Many of the pieces that motivate graphics algebra have been lying around for decades: experimental design algebras, relational algebras, table algebras. These algebras emerged from separate disciplines, so that in most instances, researchers have been unaware of developments in the other disciplines. What is new about chart algebra is the explicit and formal equivalence between the data structures needed for statistical models and the methods for displaying them in charts. In a general sense, this equivalence allows us to think about manipulating data by manipulating statistical representation elements of a chart.

The GOG project has had several purposes. One, of course, is to develop statistical graphics systems that are exceptionally powerful and flexible. Another is to understand the steps we all use when we generate charts and graphs. This understanding leads to a formalization of the problem that helps to integrate the miscellaneous set of techniques that have comprised the field of statistical graphics over several centuries. Another purpose is to develop, ultimately, intelligent systems that can 1) graph data without human specification and 2) read already published statistical graphics to recover data and interpret relationships. Finally, we hope to define problem specification and user interaction in a way that enables graphics systems to be understood by ordinary people as well as by statisticians. By formally relating display structure and statistical models, we can produce environments in which users interact with data, receive guidance, and draw conclusions that are based on appropriate statistical inferences.

References

Abiteboul, S., Fischer, P.C., Schek, H.J.: Nested Relations and Complex Objects in Databases. Springer, New York (1989)

Agrawal, R., Gupta, A., Sarawagi, S.: Modeling multidimensional databases. In: Gray, A., Larson, P.-Å(eds.) Proceedings of 13th International Conference of Data Engineering (ICDE), IEEE Computer Society, pp. 232–243 (1997)

Andrews, D.: Plots of high dimensional data. Biometrics **28**, 125–136 (1972)

Barnett, V., Lewis, T.: Outliers in Statistical Data. Wiley, New York (1994)

Becker, R.A., Cleveland, W.S., Shyu, M.-J.: The design and control of trellis display. J. Comput. Stat. Graphics **5**, 123–155 (1996)

Bertin, J.: Smiologie Graphique. Editions Gauthier-Villars, Paris (1967)

Date, C.: What is a domain? In: Date, C.: (eds.) Relational Database Writings 1985–1989, pp. 213–313. Addison-Wesley, Reading, MA (1990)

Date, C.J., Darwen, H.: Relation-valued attributes. In: Date, C.J., Darwen, H. (eds.) Relational Database: Writings 1989-1991, pp. 75–98. Addison-Wesley, Reading, MA (1992)

Feiner, S., Beshers, C.: Worlds within worlds: Metaphors for exploring n-dimensional virtual worlds. In: Proceedings of ACM UIST '90. pp. 76–83. ACM Press, New York, NY (1990)

Fisher, R.: Statistical Methods for Research Workers. Oliver and Boyd, Edinburgh (1925)

Fisher, R.: The Design of Experiments. Oliver and Boyd, Edinburgh (1935)

Friedman, J.: Exploratory projection pursuit. J. Am. Stat. Assoc. **82**, 249–266 (1987)

Glymour, C., Madigan, D., Pregibon, D., Smyth, P.: Statistical inference and data mining. Comm. ACM **39**(11), 35–41 (1996)

Gyssens, M., Lakshmanan, L.V.S., Subramanian, I.N.: Tables as a paradigm for querying and restructuring (extended abstract). In: Proceedings of the fifteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, pp. 93–103. ACM Press, Montreal, Quebec, Canada (1996)

Hand, D.J., Mannila, H., Smyth, P.: Principles of Data Mining: Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA (2001)

Hartigan, J., Kleiner, B.: Mosaics for contingency tables. In: Computer Science and Statistics: Proceedings of the 13th Symposium on the Interface, pp. 268–273 (1981)

Hastie, T., Tibshirani, R., Friedman, J.H.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York (2001)

Heiberger, R.M.: Computation for the Analysis of Designed Experiments. Wiley, New York (1989)

Inselberg, A.: The plane with parallel coordinates. Visual Comput. **1**, 69–91 (1984)

Johnson, B., Schneiderman, B.: Treemaps: A space-filling app. roach to the visualization of hierarchical information structures. In: Proceedings of the IEEE Information Visualization '91, pp. 275–282 (1991)

Kutner, M.H., Nachtschiem, C.J., Wasserman, W., Neter, J.: App.lied Linear Statistical Models, Richard D. Irwin, Incorporation, Homewood, IL (1996)

Linhart, H., Zucchini, W.: Model Selection. Wiley, New York (1986)

Mackinlay, J.: Automating the design of graphical presentations of relational information. ACM Trans. Graph. (TOG) **5**(2), 110–141 (1986)

Makinouchi, A.: A consideration on normal form of not-necessarily-normalized relation in the relational model. In: Proceedings of the Third International Conference on Very Large Data Bases. pp. 447–453 (1977)

Mendelssohn, R.C.: The bureau of labor statistic's table producing language (TPL). In: Proceedings of the 1974 annual conference, Bureau of Labor Statistics, pp. 116–122. Washington, DC (1974)

Mihalisin, T., Timlin, J., Schwegler, J.: Visualizing multivariate functions, data, and distributions. IEEE Computer Graphics and Applications. **11**(13), 28–35 (1991)

Mosteller, F., Parunak, A.: Identifying extreme cells in a sizable contingency table: Probabilistic and exploratory app. roaches. In: Hoaglin, D.C., Mosteller, F., Tukey, J.W. (eds.) Exploring Data Tables, Trends, and Shapes, pp. 189–224. Wiley, New York (1985)

Nelder, J.A.: The analysis of randomised experiments with orthogonal block structure (Parts I and II). Proc. Roy. Soc. Lond. Series A **283**, 147–178 (1965)

Norton, A., Rubin, M., Wilkinson, L.: Streaming graphics. Stat. Comput. Graph. Newsletter **12**(1), 11–14 (2001)

Pedersen, D., Riis, K., Pedersen, T.B.: A powerful and SQL-compatible data model and query language for OLAP. In: Proceedings of the thirteenth Australasian conference on Database technologies, Australian Computer Society, Incorporation, pp. 121–130. Melbourne, Victoria, Australia (2002)

Ramsay, J., Silverman, B.: Functional Data Analysis. Springer, New York (1997)

Roth, S.F., Kolojejchick, J., Mattis, J., Chuah, M.C., Goldstein, J., Juarez, O.: SAGE tools: a knowledge-based environment for designing and perusing data visualizations. In: Proceedings of the CHI '94 conference companion on Human factors in computing systems, pp. 27–28. ACM Press, Boston, Massachusetts, USA (1994)

Sarawagi, S., Agrawal, R., Megiddo, N.: Discovery-driven exploration of OLAP data cubes. In: Proceedings of the Sixth International Conference on Extending Database Technology (EDBT) (1998)

Shoshani, A.: OLAP and statistical databases: similarities and differences. In: Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems, pp. 185–196. ACM Press, Tucson, Arizona, USA (1997)

Simkin, D., Hastie, R.: An information processing analysis of graph perception. J. Am. Stat. Assoc. **82**, 454–465 (1987)

Stevens, S.S.: On the theory of scales of measurement. Science **103**, 677–680 (1946)

Stolte, C., Tang, D., Hanrahan, P.: Polaris: A system for query, analysis, and visualization of multidimensional relational databases. IEEE Trans. Visual. Comput. Graph. 8 **1**, 52–65 (2002)
Taylor, B.N.: The international system of units (SI). In: Special Publication 330, pp. 171–182. NIST, Washington, DC (1997)
Tukey, J.W.: Exploratory Data Analysis. Addison-Wesley Publishing Company, Reading, MA (1977)
Wegman, E.J.: Hyperdimensional data analysis using parallel coordinates. J. Am. Stat. Assoc. **85**, 664–675 (1985)
Wilkinson, G.N., Rogers, C.E.: Symbolic description of factorial models for analysis of variance. J. Roy. Stat. Soc. Series C **22**, 392–399 (1973)
Wilkinson, L.: A graph algebra. In: Computing Science and Statistics: Proceedings of the 28th Symposium on the Interface, pp. 341–351 (1996)
Wilkinson, L.: The Grammar of Graphics. Springer, New York (1999)
Wilkinson, L., Rope, D., Carr, D., Rubin, M.: The language of graphics. J. Comput. Graph. Stat. **9**, 530–543 (2000)
Wills, G.: Using the java 2 platform for advanced information visualization. Unpublished paper presented at the Annual Java One conference (2002)

Chapter 14
Statistical User Interfaces

Sigbert Klinke

14.1 Introduction

A *statistical user interface* is an interface between a human user and a statistical software package. Whenever we use a statistical software package we want to solve a specific statistical problem. But very often at first it is necessary to learn specific things about the software package.

Everyone of us knows about the "religious wars" concerning the question which statistical software package/method is the best for a certain task; see Marron (1996) and Cleveland and Loader (1996) and related internet discussions. Experienced statisticians use a bunch of different statistical software packages rather than a single one; although all of the major companies (at least the marketing departments) tell us that we only need their software package.

Why do these wars, not only concerning statistical software packages, evolve? One of the reasons is that we need time to learn about the software package besides learning the statistical methodology. And the more time we need to learn to use the software package, the more we are defending "our" software package. But if we need to spend a lot of time for learning to use a statistical software package, then the question, whether this software package really has a good user interface, arises?

The basic problem is that the development of statistical software is started by experts of statistical methodology. Since they have to have a deep inside in their special fields, most of them have a very limited view to problems of other users. We generally do not consider the sex of the users, the ethnic or cultural background, the statistical knowledge or other factors when we *create* a user interface.

S. Klinke (✉)
Ladislaus von Bortkiewicz Chair of Statistics, C.A.S.E. - Center for Applied Statistics & Economics, Humboldt-Universität zu Berlin, Berlin, Germany
e-mail: sigbert@wiwi.hu-berlin.de

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3__14,

Thus the important questions we have to answer when we *design* a user interface are: What does the user want to do with this software package? And how can he do it effectively?

Fortunately, during years of development of software packages, we have collected a lot of experience about human behavior and specific differences because of sex, ethnic or cultural background and so on. In the book of Shneiderman (1998) a lot of rules have been collected which should help the software developers to avoid the worst problems. But the best way for developing a user interface is a development cycle of designing, testing, redesigning, testing, redesigning, . . . This will take a lot of time and money, but redesigning the basic components of a software package at late development will be much more expensive or even impossible.

In this chapter only a subset of all statistical software packages, namely DataDesk 6.0, GGobi 0.99, R 1.7.1, SPSS 11.0 (English version), SYSTAT 10, XploRe 4.6 and Mathematica 4.3 will be used for illustrating purposes (see also the section "Web references"). It covers a wide range of different statistical software packages.

In all statistical software packages we can find errors in the user interface design. User interface design is not a exact science as statistics, but it relies heavily on the way how we perceive information and how we react to it. That includes that in every design we will make errors before we can learn to avoid them in future. Therefore a lot of theories are available, partially explanatory, partially predicting, which should help us to design user interfaces.

14.2 The Golden Rules and the ISO Norm 9241

Complex statistical tasks require more and more complex statistical programs. In consequence more complex user interfaces are needed to be developed. Software developers recognized that common rules exist in simplifing the use of software systems. Shneiderman (1998) published a summary of these rules known as the "golden rules" of user interface design:

1. *Achieve consistency*
   The first rule is the one which is most often violated, especially when teams of people work together. Users expect that in similar situations the software behaves similarly and requires the same actions from the user.
2. *Use shortcuts*
   Beginners need a comfortable clear structured way to accomplish their task, but power users of a software package want to do their work as quickly as possible.
3. *Give informative feedback*
   Users need to have a feedback on their actions. The amount of the feedback depends on the action and the user's experience. Frequent actions require only short answers whereas rare actions require more extended answers. Beginners need more feedback whereas power users may just need acknowlegdement that the task is finished.

4. *Design closed actions*
   Actions should have a clear structure with a start and a well-defined end. This holds especially for dialogs and forms.
5. *Offer error prevention and easy error handling*
   Software should not support erroneous input from the user and provide default values. The user should be able to recover easily from errors. If a user can revert his actions easily then this will increase his trustworthiness in the software package.
6. *Support user control*
   Users prefer to initiate actions in a software package. If a user believes that he only reacts to the system he will experience a control loss.
7. *Reduce memorization*
   Humans can only remember seven plus minus two information bits in their short term memory Miller (1956). Extensive memorization to handle a software package should be avoided.

A formalization of the rules can be found, partially in very detailed instruction, in the ISO (International Standardization Organization) norm 9241. The norm itself, which distinguishes between requirements and recommendations, consists of 17 parts:

1. General introduction
2. Guidance on task requirements
3. Visual display requirements
4. Keyboard requirements
5. Workstation layout and postural requirements
6. Environmental requirements
7. Display requirements with reflections
8. Requirements for displayed colors
9. Requirements for non-keyboard input devices
10. Dialogue principles
11. Usability statements
12. Presentation of information
13. User guidance
14. Menu dialogs
15. Command dialogs
16. Direct manipulation dialogs
17. Form-filling dialogs

14.3 Development of Statistical User Interfaces

In the past we have seen the development of software according to new concepts in computer science. From the end of the 1960s/beginning of the 1970s when computers became available till now, we can distinguish several phases. In the

beginning we had non-interactive, batch oriented software packages, e.g. SAS and SPSS. The idea of incremental programming and interaction lead to systems like PRIM-9 (Tukey et al. 1973, 1974) or programming languages like BASIC. Another paradigm was that the notation used should be compact, like in languages as in APL or C. The availability of personal computers with window systems introduced graphical user interface (GUI) in contrast to command line interfaces (CLI) also to statistical software packages. As mentioned in the interview with J.E. Gentle (Hardle 2004) statistical software packages nowadays should come with both interfaces. During the last fifteen years we saw that team programming, reusability of software, network/internet computing and access to databases had their impact on programming languages (ADA, C++, Java, SQL) as well as on statistical software packages like S/S-Plus, R, XploRe, Jasp, etc.

Before we start to develop a statistical software package and a user interface (GUI or CLI), we should think about the kind of problems a user may have:

1. A user could formulate an inadequate goal, e.g. using Excel for semi-parametric modeling.
2. A user could not find the right tool/method, since the developer uses inappropriate labels, e.g. the command `paf` in XploRe should better be named `selectrows`.
3. A user could not be able to find or execute a specific method, e.g. in a statistical programming language with a lot of commands and macros, he could loose the overview. For example, languages like XploRe or R consist of a lot of commands, macros and packages.
4. The feedback from the software package to a user action could be inappropriate or misleading, e.g. the error message `syntax error`.

The first problem can not be solved with a better interface design, but so can the latter three (Franzke 1995). We have two ways to solve them: either we make a better user interface or the user has to spend a lot of time for learning about the interface. One of the most time consuming design error is a subtle inconsistency, for example if the parameter orders for nearly identical actions, either in formulas for GUIs or commands in CLIs, are different. The user will always loose time to look up these subtle differences.

Obviously we can not develop one user interface for all users. The slogan *Know your user* (Hansen 1971) in detail (statistical background knowledge, cultural background, etc.) is an important building block to the success of a (statistical) software package. We can distinguish three types of users: *novice users* who need to execute a small set of simple exercises and need an informative feedback from the software package. *Periodic users* who need support for forgotten execution sequences and need to know how to extend the current software package. But they usually need only a short feedback to an executed task. A *power user* is mainly interested in fast answers and needs only very limited feedback. Some statistical software packages, XploRe and R offer even multiple GUIs for different types of users.

However, basic guidelines for all user types are:

1. Consistency in the appearance
2. Effective information control by the user
3. Minimal memorization and minimal data entry by the user
4. Flexible adaption of the data display
5. Compatibility between data entry and output

14.3.1 Graphical User Interfaces

For novice users it is clear that they prefer software with GUIs (see Temple, Barker and Sloane, Inc. 1990), but for power users this is not quite clear, see Ulich et al. (1991). There are some general drawbacks of GUIs:

1. They need valuable screen space for menus, icons, windows etc.
2. There is no consistent set of menus, icons and windows. We have to relearn them for each software package.

A look at Fig. 14.1 shows the entry screens of different statistical software packages. Here we find all elements forming a modern GUI: menu bar, toolbar(s)

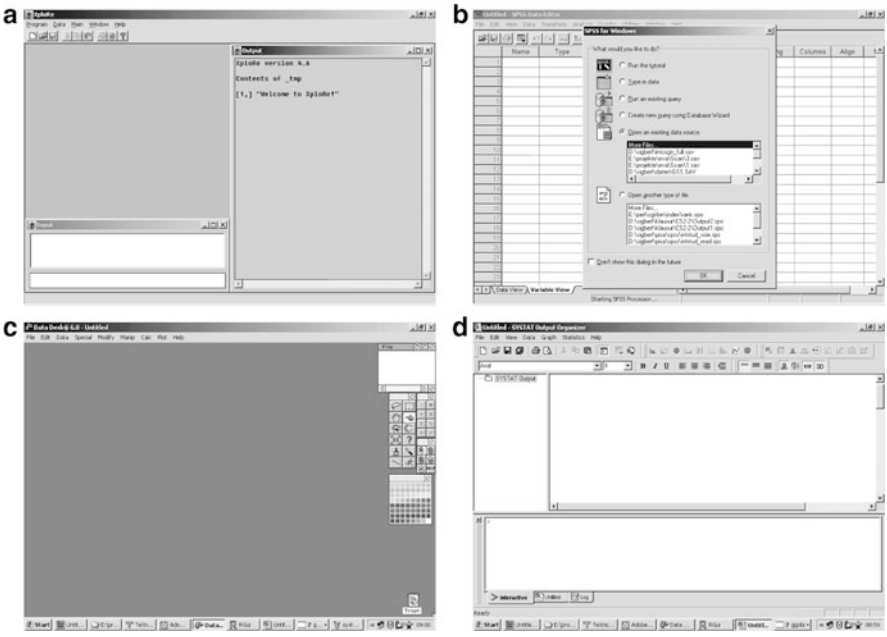


Fig. 14.1 Entry screens of the statistical software packages, (**a**) XploRe, (**b**) SPSS, (**c**) DataDesk and (**d**) SYSTAT

and multiple windows. Note that a *statistical* user interface is more than a GUI: design of the programming language, help system, basically every aspect in the interface between a user and a statistical software package.

Some packages try to help a novice user with his next task. SPSS opens, after starting the program, a dialogue box to load a dataset (see Fig. 14.1b). For R, which can load all data objects from previous sessions automatically, such feature is not necessary.

14.3.2 Toolbars

Although toolbars play a more and more important role in software nowadays, we immediately notice the sparse toolbars (see Fig. 14.2), due to the fact that we have no common set of icons. For example GGobi and DataDesk do not offer any toolbar, XploRe, SPSS and R offer only toolbars related to standard tasks, like opening, saving and printing programs, images etc. and specialized software functions. Among the considered programs only SYSTAT offers toolbars for standard statistical graphics (histogram, pie chart, boxplot, scatterplot and scatterplot matrices) and tasks (descriptive statistics, two-way-tables, two-sample t-test, ANOVA, correlations, linear regression, clustering and non-linear modeling). But to learn the meaning of the icons may take some time.

14.3.3 Menus

The first parts of a software package we use are the menu bar, the menus and the menu items. Menus can give a clear structure to the methods/actions of a software

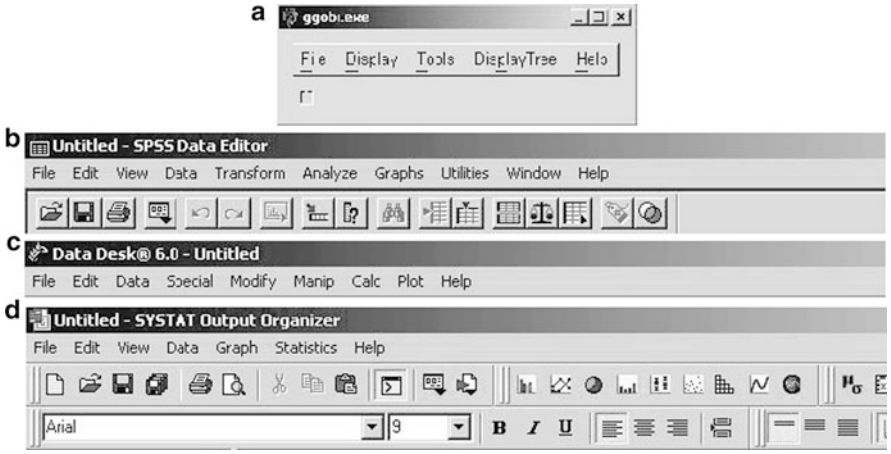


Fig. 14.2 Menu bars and toolbars of the main windows of (**a**) GGobi, (**b**) SPSS, (**c**) DataDesk and (**d**) SYSTAT

package. Liebelt et al. (1982) have found that a proper organization of the menu reduces the error rate to about 50%. Most statistical software packages have a very clear separation of the tasks in the menu bar (see Fig. 14.2).

It might be distracting if the position of the menu items changes (Mitchell and Shneiderman 1989). For unused menu items (not applicable tasks in the current situation) it is preferable if they are grayed out and do not vanish from the menu. Statistical software packages behave very differently. The menu bar in XploRe and R changes heavily depending on the active window which can be very disturbing to the user. It would have been better to attach an appropriate menu to each window. Also in GGobi the menu changes depending on the active window: compare Fig. 14.4a to Fig. 14.2a. Nevertheless this is less disturbing to the user because additional menus appear only once in the menu bar and heavy changes take place in the items of the `Display` menu which are invisible to the user. The menu `Display` is empty after starting GGobi, but filled when a dataset is loaded.

If we create a menu hierarchy we basically have two possibilities to organize them: a small, deep hierarchy or a broad, flat hierarchy. Norman and Chin (1988) found that broad, flat hierarchies lead to a better user performance. Most software packages follow this concept intuitively, none of the software packages has a menu depth larger than four.

Several orders of menu items within menus are possible: by time, by numbering, by alphabet, by importance or by function. Card (1982) found that an alphabetical order of menu items is better than a functional order. McDonald et al. (1983) showed that the advantage of the alphabetical order is lost if each menu item consists of a one line definition rather than of one meaningful known word. Nowadays all statistical software packages prefer a functional order by separating the menu items with horizontal lines into menu item groups. But within a menu item group the order is unclear.

To achieve consistency within a menu system, the same terms should be used. If we use one word items then they should be clearly separable, like “insert” and “delete”. The exact difference between “change” and “correct” is unclear. Cyclic menus, which means we can achieve the same task by different ways through the menu hierarchy, should be avoided. Users become unsure where to find a specific action; the same problem is well known from the World Wide Web.

The approach to put the most used menu items at the top and suppress the others, may irritate the user. The irritation occurs not with the most used items, but with the items which are used less often. Their position appears to be more or less randomly. Thus, Sears and Shneiderman (1994) found that bringing only the most used items to the top of the menu is an effective technique.

For power users shortcuts, e.g. through keyboard sequences or toolbars, are very important. Often used menu items should get short shortcuts, e.g. Ctrl+O for open a data set, whereas rarely used shortcuts can have longer keyboard sequences. Most statistical software packages offer only the standard shortcuts coming from the Windows operating system; only GGobi offers shortcuts for different view modes.

Unfortunately, we have no common sets of shortcuts for a (statistical) task. We have not even a common naming convention for menus, e.g. the statistical tasks are in the `Calc` menu in DataDesk, in the `Statistics` menu in SPSS and in the `Analyze` menu in SYSTAT.

For an effective menu system design it is helpful to log the user choices and to analyze them for improvements.

14.3.4 Forms and Dialog Boxes

The use of menus leads to another form of interaction with the user: forms and dialog boxes. Basically they should:

- Have a meaningful title.
- Use a consistent and for the user well known terminology.
- Group information in a consistent and meaningful way.
- Minimize mouse movement and jump from one item to another item in a useful way.
- Support error prevention.
- Allow for fast error correction.
- Provide default values, if possible.
- Indicate optional values clearly.
- Inform when the dialog or form has enough information.

Here, a very good example is SPSS. See as example in Fig. 14.3 four out of six steps for reading ASCII data into SPSS. The six dialog boxes are grouped in information about:

1. Reuse of old format.
2. Information about the arrangement of the variables.
3. Information about the arrangement of the cases.
4. Separation between single data.
5. Variable formats and names.
6. Saving the defined format.

The forms always provide default values, show the consequence of changing a value in the bottom and allow easy navigation between forms forward and backward. We can cancel the whole operation or finish it at any time. The last form gives a clear signal when we are finished. The SPSS developers have designed all these forms and dialogs very carefully.

However, we have to keep in mind that pure statistical programming languages, like R or XploRe, will have to incorporate forms and dialog boxes in their programming language. This turns out to be a difficult task.

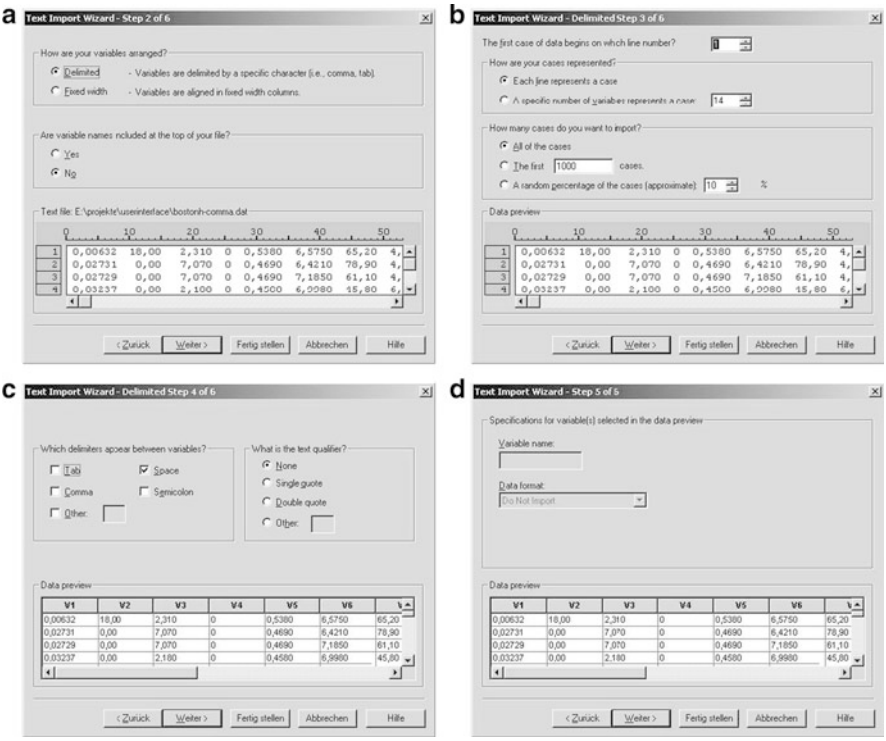


Fig. 14.3 Four out of six steps of reading ASCII data in SPSS. They provide a very clear, intuitive interface even for unexperienced users for reading data

14.3.5 Windows

Usually statistical software packages use different windows to visualize data and output. Figure 14.4 shows a scatterplot of the variables "percentage of lower status people" and "median houseprice" of the Boston Housing data (Harrison and Rubinfeld 1978). We easily find that the software packages have different methods to handle output. SPSS and SYSTAT have a special independent output window for text and graphic output. DataDesk, R (by default) and XploRe use the multiple document interface (MDI) coming with the Windows operating system. Actually R allows to switch between different types of handling windows (MDI/SDI). GGobi creates a complete set of independent windows.

In GGobi and DataDesk the data in the windows is linked (see Fig. 14.5a). Thus interactive brushing is very easy.

A problem of some statistical software packages is that the user can easily create a lot of windows, see in Fig. 14.4 and even worse in Fig. 14.5 for DataDesk. But a large number of windows can easily irritate the user. Statistical software packages have tried to overcome this problem with different approaches: having separate

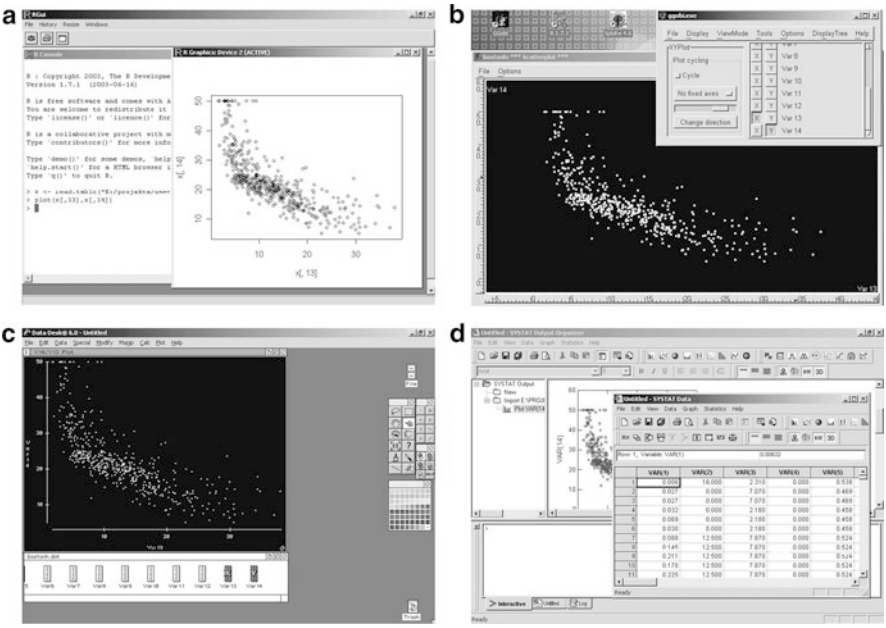


Fig. 14.4 Scatterplot of the variables "percentage of lower status people" and "median house-price" of the Boston Housing data in (**a**) R, (**b**) GGobi, (**c**) DataDesk and (**d**) SYSTAT

graphic types, for example the scatterplotmatrix in SPSS or trellis displays in R; XploRe has a datatype for a graphical display which consists of single plots. The idea is always the same: statistical information that belongs together should be in one window. Another strategy is a virtual desktops (see the software package VanGogh in Keller 2003) as we find them under Linux GUIs.

Power users prefer full-screen views (see Bury et al. 1985). Note that in Fig. 14.5 we tried to maximize the size of the graphics in R, SPSS and XploRe. SPSS and SYSTAT follow partially such a strategy with separating clearly between spreadsheet presentation of data and variables and output results. But Staggers (1993) has shown that users work faster with compact information on one screen rather than to scroll.

The grouping of information in a window plays an important role in GUI design. Fitts (1954) developed an effective forecasting model for the time T for a movement over a distance D to an object with width W

$$T = C_1 + C_2 \log 2(2D/W)$$

with device dependent constants C_1 and C_2.

May be approaches like "The CAVE" (Cruz-Neira et al. 1993), a virtual reality environment, will lead to more screen space.

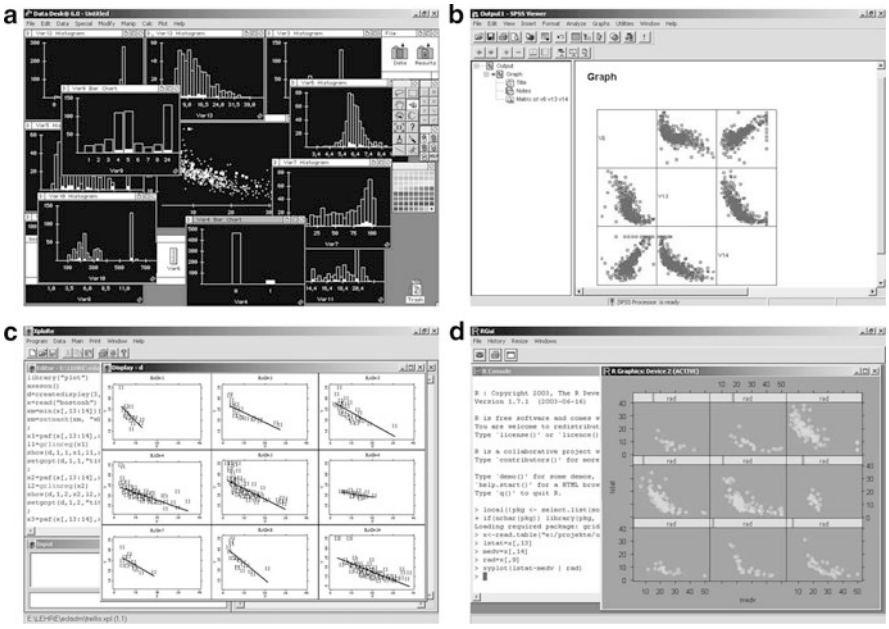


Fig. 14.5 (**a**) Linked scatterplot, histograms and barcharts in DataDesk. (**b**) Scatterplotmatrix of three variables "average number of rooms", "percentage of lower status people" and "median houseprice" in SPSS. (**c**) Trellis display of the variables "percentage of lower status people" and "median houseprice" conditioned on the variable "index of accessibility to radial highways" in XploRe. (**d**) Trellis display of the same variables in R

The question of the contents of the windows is related to showing windows. Tufte (1983, 1990) has shown proper and improper use of statistical graphics (see also Chap. II.11). Modern statistical techniques, like data mining, but also exploratory data analysis, has lead to principles of analysis like *get an overview*, *zoom*, *select* and *look to details*.

14.3.6 Response Times

The productivity of a user depends heavily on the response time of a software package to a user action. To achieve a good productivity we have to balance between the speed of working and the error rate. Fast response times (< 1 s) lead to faster user work. Fast working increases the error rate because the user does not think much about the current action since he is concentrated on the responses from the software package. Slow response times (> 15 s) lead to slower user work. Slow working decreases the error rate because the user has time to think about the next action. But if he makes a mistake he will loose time.

The right amount of the response time depends also on user experiences, e.g. if he sees that one software package reads a large dataset in 30 s whereas another software package needs 3 m for the same dataset then he will assume something has gone wrong. A power user is generally more impatient than a novice user. A partial solution to the problem of slow response times is a progress bar which shows how much time it will take till the end of the action.

Generally a user expects that simple actions, e.g. reading a small dataset, are done fast and complex actions, e.g. building a model from a large dataset, can take much longer time. The study of Martin and Corl (1986) found that the response time for a complex statistical task does not matter much for productivity, whereas the response time for a simple task (entering data) is linearly related to productivity. A variation in response times ($\pm 50\%$) does not matter much. In a set of mixed tasks the user balances out: he thinks about the task when the response time is slow and works fast if the response time is fast.

14.3.7 Catching the User Attention

In Fig. 14.4d we see that in SYSTAT the data editor stays on top, although we just created a scatterplot in the underlying output window. But the user attention is still directed to the data editor. Similar problems can be observed in other software.

Another point in GUI design we should consider is the way how we catch the attention of the user. In statistical graphics Tufte (1983, 1990) has shown how the user's attention can be redirected from the data. In the same manner a too colorful GUI may distract the user. Wickens (1992) analyzed how to catch the users attention and gave some hints:

- Use 3 different fonts with 4 different sizes in 2 intensities.
- Use up to 4 standard colors, more colors have to be used with care.
- Use soft sounds when everything is okay, use hard sounds for warnings and errors.

Nowadays operating systems offer a large variety of true-type fonts, nevertheless most people use only a few fonts in their documents.

Especially the use of colors may create special problems. First, different user may combine different reactions to the same color (cultural background); second, it is known that in Europe and North America 8% of the population have problems in recognizing a color correctly. The largest problem here is the red-green blindness, both colors appear grey to such people.

The use of sound should only be an additional option. During teaching or when working in PC-Pools it will distract other users.

14.3.8 Command Line Interfaces and Programming Languages

In the beginning of the computer age all programs only had CLIs. One of the largest statistical software packages which has survived from these times, SPSS, still has a CLI. But it is hidden by a GUI and we can reach it via SPSS syntax editor. Statistical programming languages, like R and XploRe, are more like CLIs embedded in a GUI. Only statistical software packages like GGobi and DataDesk are real GUI software, but even DataDesk has a (visual) programming language.

In the recent past we observed that statistical packages like R or XploRe have a tendency to be split up between a GUI and a CLI. In fact on the R-Project page we find more than one GUI for R.

CLI provides some advantages compared to a pure GUI. Some manipulations, for example arithmetic transformation of data, can be done much faster with the keyboard than with the mouse.

With a programming language we can achieve a precise and compact way to manipulate and analyze data. We should be able to easily learn, read and write the programming language. Some problems that can arise are:

- The design has too many objects and actions. A hierarchical approach like organizing objects and actions in libraries may help here. However, R and XploRe suffer both from an overwhelming number of packages, commands and programs.
- Sometimes the names chosen for an action are to close to computer science and not to statistics. Do we *load*, *read*, *open* or *get* a dataset (see also Table 14.1)?
- Inconsistent order of parameters for operations.

Modern statistical programming languages implement matrix algebra since we can easily transfer expressions, e.g. for computing the coefficients of a multiple linear regression, like $(X^\top X)^{-1}(X^\top Y)$ into a program (in XploRe: `inv(x'*x)*(x'*y)`). This allows for fast error correction and fast learning.

Caroll (1982) found that hierarchical (verb-object-qualifier) and symmetric command sequences, like in Table 14.2 for linear regression, lead to the best user performance and can be easily learned and remembered. The reality in software packages is shown in the Table 14.3.

Again power users prefer rather short names whereas novice users can find actions with long names more informative. It is the best to have both available, like `DoLinearRegression` and `DoLinReg` or even `dlr`. Ehrenreich and Porcu (1982) suggest rules to make (automatic) abbreviations and Schneider (1984) proposes possible abbreviation methods:

- Use a simple rule to create abbreviations
  - Truncation (most preferred by users)
  - Deletion of vocals (`DLnrRgrssn`)
  - Use last and/or first letter

Table 14.1 Reading ASCII file with the boston housing data

Software	Reading ASCII data
R	`x <- read.table("c:/data/bostonh.dat", header=FALSE)`

SPSS	`GET DATA /TYPE = TXT`
	`/FILE = 'c:databostonh.dat'`
	`/DELCASE = LINE`
	`/DELIMITERS = " "`
	`/ARRANGEMENT = DELIMITED`
	`/FIRSTCASE = 1`
	`/IMPORTCASE = ALL`
	`/VARIABLES = CRIM F7.2 ... MEDV F5.2 .`

SYSTAT	`IMPORT "c:/data/bostonh.dat.dat" / TYPE=ASCII`
XploRe	`x = read ("bostonh")`

Table 14.2 Example of a hierarchical and symmetric command sequences in the context of linear regression

```
do linear regression
do linear regression stepwise
do linear regression forward
do linear regression backward
plot linear regression line
plot linear regression residuals
```

Table 14.3 Simple linear regression with intercept between the variable "percentage of lower status people" (lstat) and the dependent variable "median houseprice" (medv) of the Boston Housing data in different statistical programming languages

Software	Linear regression commands
R	`res <- lm (medv ~ lstat)`

SPSS	`REGRESSION`
	`/MISSING LISTWISE`
	`/STATISTICS COEFF OUTS R ANOVA`
	`/CRITERIA=PIN(.05) POUT(.10)`
	`/NOORIGIN`
	`/DEPENDENT lstat`
	`/METHOD=ENTER medv .`

SYSTAT	`REGRESS`
	`USE "c:/data/bostonh.dat"`
	`MODEL MEDV = CONSTANT + LSTAT`
	`ESTIMATE`

XploRe	`res = linreg (lstat, medv)`

  - Standard abbreviation, e.g. `QTY` for `Quantity`
  - Phonetical abbreviation, e.g. `XQT` for `Execute`

- Use a (simple) second rule for conflicts
- Apply the second rule very rarely
- Abbreviations with result from the second rule should have a special symbol included
- User should know both rules
- Abbreviations should have a fixed length
- The software package should always use the long name, e.g. in error messages

Modern editors, e.g. the Visual Basic editor in Microsoft Office, support the writing of programs with semi-automatic command completion.

A future dream is that (statistical) software understands natural language. What has proven to be valuable to the user is the generation of a report of results in natural language.

Table 14.4 Error messages in different software packages

Software	Example	Error message

XploRe	`proc()=test(x)`
	`if(x=1)`
	`"true"`
	`else`
	`"false"`
	`endif`
	`endp`
	`test(0)`
		Syntax Error
		in test line: 2
		Parse Error
		on position 5 in line 2

R	`if (x=1) "true" else "false"`	Error: syntax error
SPSS	as in Table 14.3, just `/DEPENDENT` changed to `/INDEPENDENT`	
		Warning
	Unrecognized text appears on the REGRESSION command. The only recognized subcommands are: Global options: DESCRIPTIVES MATRIX MISSING WIDTH; Case selection/weight: REGWGT SELECT; Variable list: VARIABLES; Equation options: CRITERIA NOORIGIN ORIGIN STATISTICS; Dependent variable(s): DEPENDENT; Equ. methods: METHOD BACKWARD ENTER FORWARD REMOVE STEPWISE TEST; Residuals: RESIDUAL CASEWISE PARTIALPLOT SAVE SCATTERPLOT OUTFILE. Text found: INDEPENDENT This command is not executed *WARNING* REGRESSION syntax scan continues. Further diagnostics from this command may be misleading – interpret with care Misplaced REGRESSION METHOD subcommand – The METHOD subcommand must follow a DEPENDENT subcommand or another METHOD subcommand. It cannot follow any other subcommand. Check for a missing DEPENDENT subcommand. Text found: METHOD	

14.3.9 Error Messages

The most crucial response for a user is an error or warning message from the system. Error messages can be not very helpful, e.g. in XploRe or R `syntax error` in Table 14.4. A better solution would be to tell the user what the problem exactly was (use `x==1` instead `x=1`). But SPSS tells the user too much and the problem disappears behind the text. However, the ability of SPSS for abbreviating is impressive. From the linear regression example in Table 14.3 the parameter `/NOORIGIN` can be shortened to `/NOO`. Further shortening to `/NO` produces an error message.

The language in an error message and warning should be positive, constructive, meaningful and precise. Shneiderman (1982) found in a study that the error rate could be reduced by 28% with well constructed error messages.

Again it is a good idea to log the error message to see which ones are needed to be improved and which parts of the software package has to be improved.

14.3.10 Help System

Nowadays software is always accompanied with online help systems and tutorials, mostly HTML-based. A help system should give the user quick access to the information he needs. Depending on the type of users, they have different approaches to use a help system. Reference or alphabetical guides are useful for power users, but novice users learn most from a lot of examples. Consequently the help system of modern statistical software is mostly composed of several parts: reference/alphabetical guide, introductory tutorials, indices and a search engine.

In Fig. 14.6 we see the entry page of help systems. The variation in the software packages is large, from very sparse help systems upto detailed explanations how to use the help system.

Finding information in the help system is a crucial task. Thus good navigation, indices and search are essential for any help system. Help systems based on the windows help system, e.g. used by DataDesk, bring already the capabilities for an index and searching. Creating a good index, for a book or a help system is not easy. Especially since the developers of statistical algorithms mostly do not care much about the documentation. The quality of the help system depends heavily on the contributors to it. Maybe automated ways of analyzing tutorials and descriptions to create a hierarchy and an index can improve the help system quality.

One of useful help systems that we have seen is the help system of Mathematica which is an inherent part of it. At the top of Fig. 14.7d we see the detailed navigation, we know always where we are. Mathematica separates the information well: red background for Mathematica commands and programs and their descriptions, white background for explanations.

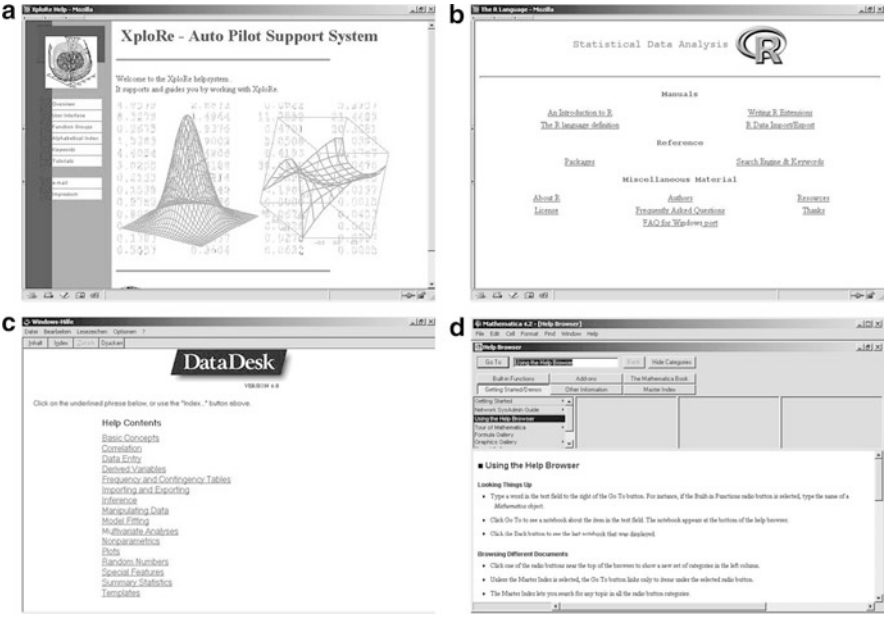


Fig. 14.6 Entry screens of the help systems in (**a**) XploRe, (**b**) R, (**c**) DataDesk and (**d**) Mathematica

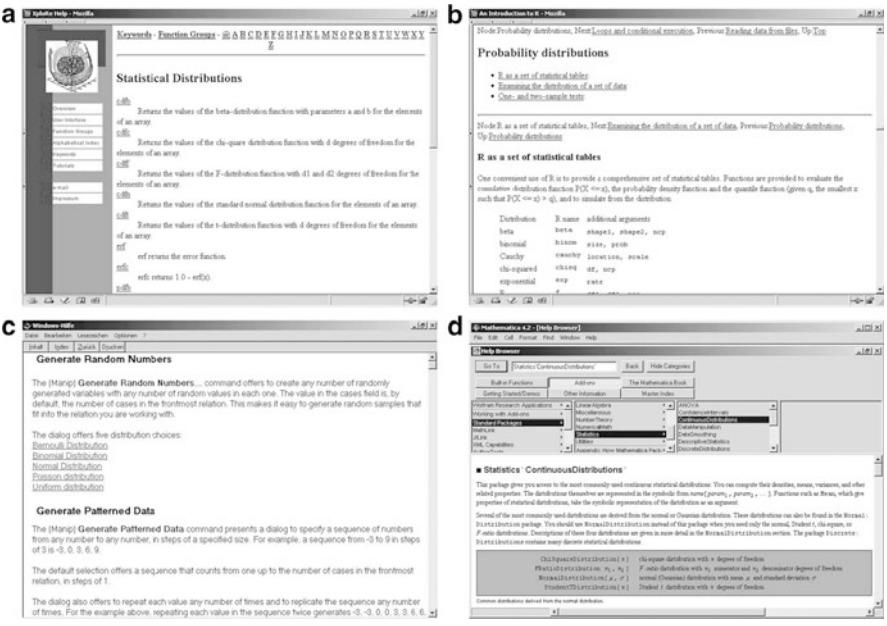


Fig. 14.7 Help system entry for statistical distributions of statistical software packages, (**a**) XploRe, (**b**) R, (**c**) DataDesk and (**d**) Mathematica

Generally, help systems and tutorials should have a simple language, short sentences (Roemer and Chapanis 1982) and a consistent terminology. This has been proven more helpful to the users and most help systems follow that suggestion. It is even more important since most help systems and tutorials are written in English and the majority of the statisticians do not speak English as native language.

14.4 Outlook

There is a wishlist for the statistical user interface of a statistical software package:

- Well-known icons for statistical tasks for useful toolbars.
- A consistent and unified terminology for menu bars and items.
- Well designed dialog boxes and forms.
- Good editors for statistical programming languages.
- A well constructed programming language.
- A well designed HTML-based help system with clear structures.
- A unique data format for exchanging data with other (statistical) software packages.

In the past we have observed that statistical software packages got various GUIs. Even SPSS offers a web-based interface now, like the R Web server or the XploRe Java client version. Currently we observe that statistical software packages are embedded via direct calls (Excel: XploRe with MD*ReX, R in the RDCOM server or DataDesk /XL) or via CORBA (R: Omega Hat project) in other software. In the future statistical software packages will use the GUI of the "host" software, but the problems we will encounter are the same.

Since the user interface design depends heavily on our perception and behavior, there is still a lot of experimental research necessary to find answers to the problems that occur.

Web References

DataDesk	http://www.datadesk.com
GGobi	http://www.ggobi.org
Jasp	http://jasp.ism.ac.jp
Mathematica	http://www.wolfram.com
PRIM-9 video	http://cm.bell-labs.com/cm/cms/departments/sia/video-library/prim9.html
R	http://www.r-project.org
– GUIs	http://www.sciviews.org/_rgui/
– Omega hat/RDCOM	http://www.omegahat.org

– Web	http://www.math.montana.edu/Rweb
S/S-Plus	http://www.insightful.com
SAS	http://www.sas.com
SPSS	http://www.spss.com
SYSTAT	http://www.systat.com
VanGogh	http://stats.math.uni-augsburg.de/VanGogh/
XploRe	http://www.xplore-stat.de
– Java client	http://www.xplore-stat.de/java/java.html
– MD*ReX	http://www.md-rex.com

References

Bury, K., Davies, S., Darnell, M.: Window management: A review of issues and some results from user testing. Technical report, IBM Human factors Center Report HFC-53, San Jose, CA (1985)

Card, S.: User perceptual mechanism in the search of computer command menus. In: Proceedings of Human Factors in Computer Systems, pp. 190–196. Washington DC (1982)

Caroll, J.: Learning, using and designing command paradigmas. Hum. Learn. **1**(1), 31–62 (1982)

Cleveland, W., Loader, C.: Smoothing by local regression: Principles and methods. In: Härdle, W., Schimek, M. (eds.) Statistical Theory and Computational Aspects of Smoothing, pp. 10–49. Physika, Heidelberg, Germany (1996)

Cruz-Neira, C., Sandin, D., DeFanti, T.: Surround-screen projection-based virtual reality: The design and implementation of the cave. In: Proceedings of SIGGRAPH'93 Conference, pp. 135–142, ACM, New York (1993)

Ehrenreich, S., Porcu, T.: Abbreviations for automated systems: Teaching operators and rules. In: Badre, A., Shneiderman, B. (eds.) Directions in Human-Computer Interaction, pp. 111–136. Ablex, Norwood, NJ (1982)

Fitts, P.: The information capacity of the human motor system in controlling amplitude of movement. J. Exp. Psychol. **47**, 381–391 (1954)

Franzke, M.: Turning research into practice: Characteristcs of display-based interaction. In: Proceedings of CHI'95 Conference: Human Factors in Computing Systems, pages 421–428. ACM, New York (1995)

Hansen, W.: User engineering principles for interactive systems. In: Proceedings of Fall Joint Computer Conference, vol. 39, pp. 523–532. AFIPS Press, Montvale, NJ (1971)

Härdle, W.: Interview with James E. Gentle. Comput. Stat. **19**(1), 1–4 (2004); Physika, Heidelberg, Germany

Harrison, D., Rubinfeld, D.L.: Hedonic prices and the demand for clean air. J. Environ. Econ. Manag. **5**, 81–102 (1978)

Keller, R.: Visualizing augsburg traffic data with vangogh. Presentation at "Workshop on Statistical Inference, Visualizing for Graphs" at Stanford University, CA., University of Augsburg, Department of Computer Oriented Statistics and Data Analysis (2003)

Liebelt, L.-S., McDonald, J., Stone, J., Karat, J.: The effect of organization on learning menu access. In: Proceedings of Human Factors Society, pp. 546–550. Twenty-Sixth Annual Meeting, CA (1982)

Marron, J.: A personal view of smoothing and statistics. In: Härdle, W., Schimek, M. (eds.) Statistical Theory and Computational Aspects of Smoothing, pp. 1–9. Physika, Heidelberg, Germany (1996)

Martin, G., Corl, K.: System response time effects on user productivity. Behav. Inform. Tech. **5**(1), 3–13 (1986)

McDonald, J., Stone, J., Liebelt, L.: Searching for items in menus: the effects of organization and type of target. In: Proceedings of Human Factors Society, Twenty-Seventh Annual Meeting, pp. 834–837. Santa Monica, CA (1983)
Miller, G.: The magical number seven, plus or minus two: some limits on our capacity for processing information. Psychol. Sci. **63**, 81–97 (1956)
Mitchell, J., Shneiderman, B.: Dynamic vs. static menus: An experimental comparison. ACM SIGCHI Bull. **20**(4), 33–36 (1989)
Norman, K., Chin, J.: The effect of tree structure on search in a hierarchical menu selection system. Behav. Inform. Tech. **8**(2), 25–134 (1988)
Roemer, J., Chapanis, A.: Learning performance and attitudes as a function of the reading grade level of a computer-presented tutorial. In: Proceedings of Conference on Human Factors in Computer Systems, pp. 239–244. ACM, Washington DC (1982)
Schneider, M.: Ergonomic considerations in the design of text editors. In: Vassiliou, Y. (eds.) Human Factors and Interactive Computer Systems, pp. 141–161. Ablex, Norword, NJ (1984)
Sears, A., Shneiderman, B.: Split menus: effectively using selection frequency organize menus. ACM Trans. Comput. Hum. Interact. **1**(1), 27–51 (1994)
Shneiderman, B.: System message design: Guidelines and experimental results. In: Badre, A., Shneiderman, B. (eds.) Directions in Human/Computer Interactions, pp. 55–78. Ablex, Norword, NJ (1982)
Shneiderman, B.: Designing the User Interface. Addison Wesley Longman, Inc (1998)
Staggers, N.: Impact of screen density on clinical nurses' computer task performance and subjective screen statisfaction. Int. J. Man Mach. Stud. **39**(5), 775–792 (1993)
Temple, Barker and Sloane, Inc.: The benefits of the graphical user interface. Multimedia Review, pp. 10–17 (1990)
Tufte, E.: The Visual Display of Quantitative Information. Graphics Press, Cheshire, CT (1983)
Tufte, E.: Envisioning Information. Graphics Press, Cheshire, CT (1990)
Tukey, J., Friedman, J., Fishkeller, M.: Prim–9: An interactive multidimensional data display and analysis system. Video. ASA Statistical Graphics Video Lending Library (1973)
Tukey, J., Friedman, J., Fishkeller, M.: Prim–9: An interactive multidimensional data display and analysis system. Technical Report SLAC-PUB-1408, Stanford Linear Accelerator Center, Stanford, CA (1974)
Ulich, E., Rautenberg, M., Moll, T., Greutmann, T., Strohm, O.: Task orientation and user-oriented dialogue design. Int. J. Hum. Comput. Interact. **3**(2), 117–144 (1991)
Wickens, C.: Engineering psychology and human performance. Harpercollins, New York (1992)

Chapter 15
Object Oriented Computing

Miroslav Virius

15.1 Introduction

Object Oriented Programming (OOP) is a preferred methodology in contemporary software development. OOP may be considered as a continuation of the well known ideas of Structured Programming and Modular Programming. If properly used, it leads to well structured code which is easy to understand for human reader, easy to debug and easy to maintain.

15.1.1 First Approach to Objects

Every computer program may be considered as a software model of a real problem. It follows, that two basic domains should be taken into account during the analysis of the problem and design of the program: the *problem* domain, which is a part of the real world, and the *model* domain, which is a mapping of the problem domain to the computer program.

The problem domain consists of a set of interacting objects. Selected objects of the problem domain must of course correspond to data structures in model domain and the interactions of objects in the problem domain must correspond to the operations with these data structures. That is, the interactions of objects in the problem domain will be represented by procedures and functions dealing with these data structures.

Example 1. Consider the program modeling the interactions of elementary particles in a detector using the Monte Carlo method. (Design and analysis of Monte

M. Virius (✉)
Czech Technical University in Prague, Faculty of Nuclear Sciences and Physical Engineering, Prague, Czech Republic
e-mail: miroslav.virius@fjfi.cvut.cz

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_15,

Carlo experiments is discussed in depth in Chap. 3.) The problem domain of this experiment simulation consists of the detector, of the particle source, of the air surrounding the experimental apparatus, of many elementary particles and, of course, of a statistical file containing the simulation results. It follows, that the model of the experiment should contain a suitable representation of the detector, a suitable representation of the particle source, statistical file, etc.

The representation of the elementary particle source may consist of the data representing its coordinates in a given coordinate system, of a description of the spectrum of the source (i.e. of probability distributions describing the emission of various types of particles, their directions, energies and other characteristics of emitted particles) etc.

15.1.2 Note on Unified Modeling Language

To formalize object-oriented analysis and design, the Unified Modeling Language (UML) is widely used. UML consists of a set of diagrams that describe various aspects of the problem solved. We use some UML diagrams in this chapter. A short introduction to the UML is given in Sect. 15.3; full description of the UML may be found in Booch (2005).

Note that the Unified Modeling Language is described by the international standard ISO/IEC 19501:2005.

15.2 Objects and Encapsulation

In the model domain, the term *object* denotes the data representation of the objects of the problem domain, together with the operations defined on this data.

This means that we define a data structure together with the operations with it. These operations are usually called *methods*. The object's data are denoted as *attributes*; the data and the methods together are denoted as *members* of the object.

A basic rule of the OOP requires that the methods should be used for all the manipulations with object's data. Only the methods of the object are allowed to access the data; no other access to the data is permitted. (We shall see later that it is acceptable to violate this rule under some special circumstances.) This principle is called *encapsulation* and is sometimes presented by the "wall of code around each piece of data" metaphor.

Note.

Methods that return the value of the attribute (data member) `X` have usually the `GetX()` identifier; methods that set the value of the attribute `X` usually have the `SetX()` identifier. In some programming environments, these identifiers may be

required. These methods are called *getters* and *setters*, respectively; common name for both getters and setters is *access methods* or *accessors*.

15.2.1 Benefits of Encapsulation

So far, there is nothing new in encapsulation: This is implementation hiding, well known from modular programming. The object may be considered as a module and the set of the methods as its interface.

The main benefit of the encapsulation is that the programmer may change the implementation of the object without affecting the whole program, if he or she preserves the interface of the object. Any change of the data representation will affect only the implementation of the object's methods.

Example 2. Let's continue with the Monte Carlo simulation of the experiment with elementary particles. The object representing the detector will, of course, contain the coordinates of some important points of the detector. The first idea could be to use Cartesian coordinates; in later stage of the program development, it will be found that the spherical coordinates will suit better – e.g., because of the detector shape and program performance.

If it were allowed to manipulate the detector data directly by any part of the program, all the parts of the program that use this data should be changed. But if the encapsulation is properly applied and the data is manipulated only by the methods of the detector, all what will have to be changed is the implementation of some detector methods.

15.2.2 Objects and Messages

The OOP program is considered as the program consisting only of objects that collaborate by means of the *messages*. This may seem a little strange, but in this context, *to send a message to an object* means *to call a method of this object*. A message means a request for an operation on the object's data, i.e. a request to perform a method.

The object may receive only those messages for which it has corresponding methods. Sending a message that the object does not recognize causes an error. It depends on the programming language whether this error is detected in the compile time or in the run time. (In C++, it is detected in the compile time.)

15.2.3 Class

Objects of the problem domain may often be grouped into *classes*; one class contains objects that differ only in the values of some properties. The same holds

for the objects in the model domain. The classes of objects in the problem domain are represented by user-defined data types called *object types* or *classes* in OOP programs.

The term *instance* is used to denote a variable, constant, or parameter of an object type. It is equivalent to the term *object*.

Class Members

Up to now, we have considered the class as a data type only; it serves as a template for the creation of instances. But in the OOP, the class is an object, too. It follows it may have its own data and its own methods and may receive messages.

Data members that are part of the class, not of particular instances, are called *class data members* or *class attributes* and the methods that correspond to the messages sent to the class are called *class methods*. Non-class members, attributes, as well as methods are, if necessary, designated as *instance members*.

Class data members contain data of the class as a whole, i.e. data shared among all the instances of the class; class methods operate on class attributes. (From the non-OOP point of view, the class data members are global variables hidden in the class, and the class methods are global functions or procedures hidden in the class.)

Note.

Class data members are often called *static data members* and class methods are called *static methods* in C++, Java, and some other programming languages, because they are declared using the `static` keyword in these languages.

Note.

The class in C++, Java and many other OOP languages may contain definitions of other types, including other classes, as class members. Even though the so called nested classes are sometimes very useful, we will not discuss them in this article.

Note.

We have already mentioned that the class in the OOP may be considered as an object, i.e. as an instance of another class; this leads to the concept of *metaclass*. The metaclass is a class that has only one instance – a class. You can find metaclasses in pure OOP languages like Smalltalk. We will not discuss the metaclass concept here.

Example 3. We may suppose – at some level of abstraction – that the representation of all the particles in the Monte Carlo simulation of a particle experiment is essentially the same. Thus, every individual particle belongs to the *class of particles*. It follows that the model will contain the `Particle` class, and the program will contain the definition of the corresponding data type – and of course some instances of this type.

Because every particle has its own mass and velocity, the `Particle` class will contain the declaration of four data items representing the particle mass and the three components of the particle velocity vector. The `Particle` class should also contain methods to set and to get the values of these data items. Later on, we will see that even other methods are necessary – e.g., a method for the interaction with the detector.

It is also necessary to know the total number of generated particles and the actual number of existing particles in the simulation program. These numbers of the particles do not describe an individual particle. It follows that they cannot be data members of any `Particle` instance; to hold, and of course to maintain these data is the task of the whole `Particle` class. These data will be stored in the class attributes – because we use the C++, we may say in static attributes – of type `int`, and they will be accessed by class methods (static methods).

Definition of the `Particle` class in C++ will be as follows:

```
// Particle class definition in C++, first approach
class Particle
{
public:
   // Constructor
   Particle(double _mass, double vX,
        double vY, double vZ);
   // Instance methods
   ~Particle() { --actual; }          // Destructor
   double GetMass() { return mass; }
   void SetMass(double m){ mass = m; }
   void SetVelocity(double vX, double vY, double vZ);
   double GetVelocityX() { return velocityX; }
   // Performs the interaction with the detector
   virtual void Interact(Detector *aDetector);
   // ... and other methods
   // Class methods
   static int GetActual() { return actual; }
   static int GetTotal() {}
private:
   // Instance data members
   double mass;
   double velocityX, velocityY, velocityZ;
   // Class data members
   static int actual;
   static int total;
};                  // End of the class declaration

// Definition of the static attributes
int Particle::actual = 0;
```

```
int Particle::total = 0;

// Definition of the constructor
Particle::Particle(double _mass, double vX,
                   double vY, double vZ)
: mass(_mass), velocityX(vX), velocityY(vY),
    velocityZ(vZ)
{
 ++actual; ++total;
}
// And other method definitions
```

We will not discuss the syntactic rules of the class declaration in the C++ here – those rules can be found in any textbook devoted to this programming language, e.g., in Stroustrup (2000). We only note a few points.

This class contains the `mass`, `velocityX`, `velocityY`, and `velocityZ` instance attributes, and the `actual` and `total` class attributes (note the `static` keyword in their declarations). It follows that every instance of the `Particle` class will have its own `mass`, `velocityX`, etc. data members. On the other hand, no instance will contain the `total` or `actual` data members. These are global variables shared among all instances and they exist even before the first instance of the `Particle` class is created and after the last one is destroyed.

The `Particle()` method is a special method called *constructor* and it serves to the construction of new instances. It is invoked as a response to the message requesting the creation of a new instance of the class. (Even though it is a class method, its declaration in C++ does not contain the `static` keyword.) Its task is to create an instance and to initialize its instance attributes. In our example, it also actualizes the values of the two class attributes.

The `~Particle()` method is another special method called *destructor*. The destructor prepares the instance for final destruction. In our example, it decreases the number of existing particles, because the instance for which the destructor is called will be immediately destroyed. (This is – unlike the constructor – the instance method. Note that in garbage collected OOP languages, e.g., in Java, destructors are not used.)

15.2.4 Object Composition

One object in a program may exploit the services of another object. It may call the methods of any other independent object, or it may contain another object as a data member. The second approach is usually called *object composition*, even though it is typically implemented as a composition of classes.

Note.

An object may not contain another object of the same class, of any class containing an object of the same class or of any derived class as data member. It may, of course, contain the pointers or the references to objects of any of these classes.

Example 4. Consider the particle source in the Monte Carlo simulation. It will be an instance of the `Source` class. For the simulation of the random processes of the particle emission, we will need a random number generator. The random number generator will be implemented in the program as an instance of the `Generator` class and will be based on the theory discussed in Chap. 2. (The `Generator` class is an example of a class that has been found during the design of the `Source` class. It does not appear in the original formulation of the problem.)

This means that the `Source` class will contain an instance of the `Generator` class or a pointer to an instance of that class:

```
class Source
{
public:
   Source();
   Particle* Generate(); // Returns pointer to
   new particle
   // ... and other methods
private:
   Generator *gen;       // Pointer to Generator
   instance
   // ... and other private members
};
```

15.2.5 Access Control

Note the `private` and `public` *access specifiers* in the class declarations above. The `public` specifier declares that all the subsequent members of the class are public, i.e., they are accessible from any part of the program. The public members of the class constitute the *class interface*. The class interface usually contains only some methods and constant attributes. (Constant attributes may be accessed directly. This does not violate the encapsulation, because constant attributes cannot be changed.)

The `private` specifier means that the following members of the class are private, i.e., accessible only from the methods of the class. In other words, private members are *implementation details* of the class that can not be used by other parts of the program. Changes of private parts of the class do not change the class interface and do not affect other parts of the program.

Later on, we will see the third access specifier, `protected`. Protected members are accessible only from the methods of this class and from all the classes derived by inheritance. Thus, they constitute the *class interface for derivation*, that may be

wider than the interface of the class for common use. We will discuss the inheritance in Sect. 15.4.

The access specifiers help to implement the encapsulation. Note that in C++, as well as in many other object oriented languages, the subject of access control is the class, not the individual objects (instances). Thus any method called for an instance of the given class may use all private members of another instance of the same class.

15.3 Short Introduction to the UML

In Sect. 15.1.2 we have mentioned the UML. This is a modeling language based on a set of diagrams describing various aspects of the problem solved:

- The *class diagram* describes the classes used in the problem and their mutual dependencies.
- The *object diagram* describes all objects (instances) in the problem and their mutual dependencies.
- The *activity diagram* describes activities of the objects.
- The *state diagram* describes the states of objects and their possible changes and transitions.
- etc.

We will use only class diagrams in this chapter.

The class in the class diagram is presented as a rectangle containing the name of the class. It usually contains also the names of the methods and the names of the attributes; both may be prefixed by symbols representing their access specification (the + sign for public members, the - sign for private members and the # for protected ones). The data type of the attributes and the return type of the methods may be shown, too.

The class name, the attributes and the methods are separated by horizontal lines in the class icon. If not necessary, attributes and methods may be omitted. Figure 15.1 shows the icon of the `Source` class as we have designed it in Sect. 15.2.4.

Associations (i.e., any relations) among the classes in UML class diagrams are represented by lines connecting the class icons; as a description, the meaning and the multiplicity of the relation may be given. For example, the numbers appended to the line connecting the `Source` and the `Particle` classes in Fig. 15.2 express the fact that one particle source may emit any number of particles and that any particle is emitted by one source. The numbers appended to the line connecting the `Source`

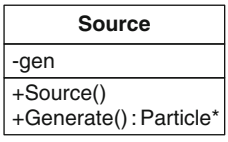


Fig. 15.1 The full UML icon of the `Source` class

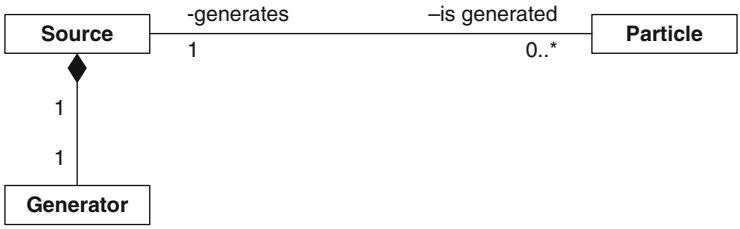


Fig. 15.2 Relations among the `Source`, `Particle` and `Generator` classes

and the `Generator` class express the fact that one source uses only one random number generator.

Object composition is expressed by an arrow ending with a filled diamond on the side of the containing class. Figure 15.2 shows relations among the `Source`, `Particle`, and `Generator` classes. Simplified class icons are used.

15.4 Inheritance

Inheritance is a very powerful tool used in OOP to derive new classes from existing ones. First, look at an example.

Example 5. Investigating our Monte Carlo simulation more deeply, we find, that various types of elementary particles can be involved: photons, neutrons, neutrinos, etc.

On the one hand, we may conclude that one common data type, the `Particle` class, is sufficient for the representation of all the different particles, because they have many common features:

- Every particle has a velocity vector.
- Every particle has a mass, a spin, and electrical charge.
- Every particle has its halftime of decay.
- Every particle may interact with the detector.
- etc.

On the other hand, the way of the interaction with the detector is substantially different for different types of the particles. In some cases, it is described by mathematical formulae, in other cases it is described by measured data only. It follows that the operation representing the interaction of the particle in the detector must be implemented in a different way for different types of the particles, and this leads to the conclusion that different types of simulated particles have to be represented by different classes in the program; but these class types share many common properties.

This situation – closely related, but different classes – can be expressed in the program model: OOP offers the mechanism of *inheritance*, which is the way of deriving one class from some other one (or other ones).

The class a new type is derived from is usually called the *base class*.

15.4.1 Base Class and Derived Class

The derived class *inherits* all the public and protected members of its base class or classes. This means that the derived class contains these members and may access them without any constrains. Private members are not inherited. They are not directly accessible in the derived class; they may be accessed only by the access methods inherited from the base class.

The derived class may add its own data members and methods to the inherited ones. The derived class may also redefine (*override*) some of the methods defined in the base class. (To override a method in a derived class means to implement a different response to the same message in the derived class.) In this case, the signature, i.e., the identifier, the return type, the number, and the types of the parameters of the overriding method in the derived class should be the same as the signature of the overridden method in the base class.

No members of the base class may be deleted in the inheritance process.

The set of all the classes connected by inheritance is usually called the *class hierarchy*.

Note that in some programming languages there are exceptions to the above rules. For example, the constructors, destructors, and overloaded assignment operators are not inherited in C++. Instead, the constructor of the derived class always calls the base class constructor and the destructor of the derived class always calls the base class destructor. The same holds for the default assignment operator of the derived class. This feature may be considered as a generalized form of inheritance, of course. Another example is the “deleted" functions in derived classes introduced in C++11 – the new standard has been issued in August 2011 the C++ language; see Stroustrup (2009).

15.4.2 Generalization and Specialization

The base class always represents a concept that is more general and more abstract than the concept represented by the derived class; it follows that the derived class represents a more specialized concept than the base class. Thus, the derived class always represents a *subclass* – or a subtype – of its base class. *Any instance of the derived class is also considered to be an instance of the base class.*

The interface of the base class is a subset of the interface of the derived class.

Consequently, an instance of the derived class may be used everywhere where an instance of the base class is expected. This rule may significantly simplify the operation with instances of many similar classes.

In the UML class diagram, the inheritance is represented by an arrow ending with triangle (not filled). The arrow points from the derived class to the base class.

Example 6. Consider the `Particle` class in our Monte Carlo simulation. This is a general concept that can be used to describe common features of all the particles involved. But in the simulation, concrete types of particles – e.g., protons, electrons, etc. – will be used.

Consequently, we will use the `Particle` class as the base class of the particles hierarchy that will contain the common data members and the common methods of all the particles. All the classes representing concrete particle types will be derived from the `Particle` class – see Fig. 15.3.

We will show only the declaration of the `Electron` class here.

```
class Electron: public Particle
{
public:
   Electron();
   void SetCharge(double _charge) { charge =
        _charge; }
   virtual void Interact(Detector *aDetector);
private:
   double charge;
};
```

The declaration of the `Electron` class contains only the declaration of the constructor, two access methods and one data member. Nevertheless, the methods `GetVelocityX()`, `SetVelocityX()`, `GetActual()` etc., inherited from the base class, may be called for any instance of this class. This class changes – overrides – the implementation of the `Interact()` method.

On the other hand, data members `mass`, `velocityX`, `actual`, etc. are not directly accessible in the `Electron` class. These data members are in the base

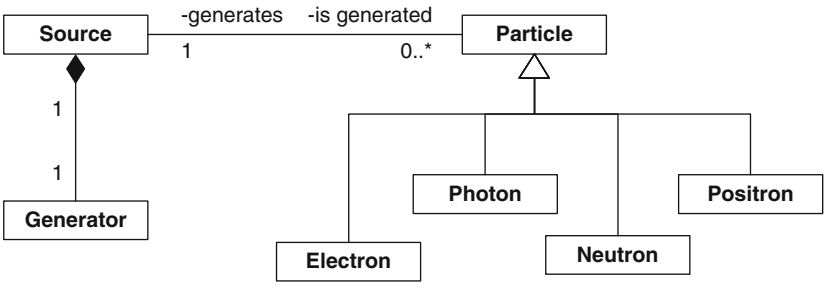


Fig. 15.3 The `Particle` class as a base class of concrete particle types

class declared as private and the derived class must manipulate them using the public access methods only. Thus the following fragment of the definition of the `Electron::Interact()` method is incorrect:

```
// Error: velocityX, velocityY and velocityZ
// are inaccessible in the Electron class.
void Electron::Interact(Detector *aDetector)
{
   double velocity = sqrt(velocityX*velocityX +
             velocityY*velocityY + velocityZ
                *velocityZ);
   // ... and so on ...
}
```

The correct form uses the access methods inherited from the base class:

```
// Correct form of the previous code fragment
void Electron::Interact(Detector *aDetector)
{
   double velocity = sqrt(GetVelocityX()
        *GetVelocityX() +
             GetVelocityY()*GetVelocityY() +
             GetVelocityZ()*GetVelocityZ());
   // ... and so on ...
}
```

Of course, this is sometimes inconvenient. If we change the access specifiers of these data members to `protected` in the base class,

```
// Particle class definition revised
class Particle
{
public:
   // Public members as before
protected:
   // Instance data members
   double mass;
   double velocityX, velocityY, velocityZ;
   // Class data members
   static int actual;
   static int total;
};
```

the problems with access to data members will not appear. On the other hand, this violates the encapsulation of the base class and it may have a negative impact on the clarity and maintainability of the program code.

15.4.3 Using Base Class as Common Interface

As stated above, instances of derived classes may be used anywhere instances of the base class are expected. This gives us very powerful tool to deal with the objects of the classes derived from the same base class in a uniform manner.

Example 7. In the Monte Carlo simulation of the particle experiment, we may store all the emitted particles in a suitable container first, then exclude particles, that do not hit the detector etc., and after that preprocessing, let the remaining particles interact with the detector. Consider the following fragment of code:

```
const int N = 1000000;      // Number of particles to
                            // emit
vector<Particle*> store;    // Store for particles
Source source1;             // Particle source
Detector detector;          // Detector in the
                            // experiment
for(int i = 0; i < N; i++)
   store.push_back(source1.Generate());
// ... after some preprocessing of the set of the
// particles:
for(int i = 0; i < store.size(); i++)
   store[i] -> Interact(detector);
```

The `store` variable is a vector (dynamically sized array) of pointers to `Particle`, the base class of all the elementary particles involved. This allows us to store pointers to instances of any class derived from the `Particle` class in this container.

The expression

```
store[i] -> Interact(detector);
```

represents the call of the `Interact()` method of the particle pointed to by `store[i]`. (In fact, this statement calls the method of the `Particle` class. To ensure that the method of the actual class of the particle is called, the method needs to be declared with the `virtual` specifier. This will be discussed later in Sect. 15.5, *Polymorphism.*)

This example demonstrates that the base class defines the common interface for all the derived classes. This common interface allows us to treat all the particles in an uniform way no matter what the actual type of the particle is.

Technical Note

The conversion from the derived class to the base class is automatic, but it deserves special attention in some situations. Consider the following assignment:

```
Electron e;                 // Non-dynamical instances
Particle p;
p = e;
```

After this statement has been executed, the `p` variable will contain an instance of the `Particle` class, not an instance of the `Electron` class! The reason is simple: The declaration

```
Particle p;
```

reserves memory only for the `Particle` instance, thus there is no place for the additional data members declared in the `Electron` class. The only way how to execute the assignment is to convert the derived class instance `e` to the base class first, i.e., to truncate the data members added in the derived class.

The same problem arises in the case of passing function arguments by value. Having the function

```
void process(Particle p);        // Pass by value
```

it is syntactically correct to write

```
process(e);                      // e is Electron
```

but the instance `e` of type `Electron` will be first cast (converted) to the `Particle` type. *This cast leads to the loss of information.*

These problems may be avoided if we use dynamical instances only. If we rewrite the preceding declarations into the form

```
Electron *ep = new Electron;  // Dynamical instances
Particle *pp;
pp = ep;                      // OK
```

the `pp` variable will still contain the pointer to the instance of the `Electron` class. (The type of the pointer is converted, not the type of the instance.)

The parameter of the `process()` function should be passed by the pointer or by the reference. In both cases, the original instance is accessible in the function body and no information is lost.

In Java, C# and other OOP languages, that use dynamical instances of the object types only and manipulate them by references, these problems do not appear.

15.4.4 Inheritance, or Composition?

It is not clear in some cases, whether a new class should be derived from some suitable class by inheritance or whether the object composition should be used.

There are two questions that should be answered in this case:

- *Is* the new class a special case of the base class proposed?
- *Has* the new class a data member of the base class proposed?

This is known as the *IS A – HAS A* test. Only if the answer to the first question is *yes*, the inheritance may be considered, otherwise the composition should be used.

Example 8. Consider the `Source` class, representing the source of elementary particles in the Monte Carlo simulation. It will be based on some generator of random numbers represented in our program by the `Generator` class. Thus, the `Source` seems to be the `Generator` class with some added functionality. Should we derive the `Source` class from the `Generator` class by inheritance?

If we apply the IS A – HAS A test, we find that the particle source is not a special case – a subclass – of the random number generator. It uses the random number generator, thus it may contain it as a data member, however, the inheritance should be avoided in this case.

Consider for an instant that we use the inheritance to derive the `Source` class from the `Generator` class,

```
// Bad use of the inheritance
class Source: public Generator
{
   // Body of the class
}
```

This would mean that we could use a `Source` instance everywhere the `Generator` instance is expected. But the random number generator is necessary even in other parts of the Monte Carlo simulation program, e.g., for the determination of the interaction type, for the determination of the features of the secondary particles resulting from the interaction (if any) etc. However, the `Source` class may not serve as the random number generator in these parts of the program.

Such a design of the `Source` class may cause that some typing errors in the program will not be properly detected and some mysterious error messages during the compilation will be reported; or even worse – it may lead to runtime errors hard to discover.

15.4.5 Multiple Inheritance

The class may have more than one base class; this is called *multiple inheritance*. The class derived from multiple base classes is considered to be the subclass of all its base classes.

Multiple inheritance may be used as a means of the class composition.

This feature is supported only in a few programming languages – e.g., in C++ (see Stroustrup 2000) or in Eiffel (see Meyer 1988). In Java, C#, Object Pascal and some other languages it is not supported. Multiple inheritance poses special problems, that will not be discussed here.

15.5 Polymorphism

We have seen at the end of the previous section, that instances of many different classes were dealt with in the same way. We did not know the exact type of the instances stored in the `store` container; it was sufficient that they were instances of any class derived from the `Particle` class.

This feature of the OOP is called *polymorphism*. Polymorphism means that instances of various classes may be used in the same way – they accept the same messages, thus their methods may be called without any regard to the exact type of the instance. In some programming languages, e.g., in Java, this is automatic behavior of the objects (or of their methods), in some other programming languages, e.g., in C++, this behavior must be explicitly declared.

There are at least two ways to achieve polymorphic behavior: The use of the inheritance and the use of the interfaces. The interfaces will be discussed in 15.5.4.

Example 9. Let's consider the example given at the end of the *Inheritance* section once again. The expression `store[i]` is of type "pointer to the `Particle` class", even though it in fact points to an instance of the `Electron`, `Photon`, or some other derived class.

It follows that the statement

```
store[i] -> Interact(detector);// Which method is
                               // called?
```

might be interpreted as the call of the `Particle::Interact()` method, even though it should be interpreted as the call of the `Interact()` method of some derived class.

15.5.1 Early and Late Binding

The previous example shows that there are two possible approaches to the resolution of the type of the instance for which the method is called, if the pointer (or reference) to the instance is used:

- *Early binding.* The type of the instance is determined in the compile time. It follows that the static (declared) type of the pointer or reference is used. This is the default for all methods in C++, C#, or Object Pascal.
- *Late binding.* The type of the instance is determined in the run time. It follows that the actual type of the instance is used and the method of this type is called. This is always used for the methods in Java. In C++, the `virtual` keyword denotes the methods using the late binding.

The late binding gives the class the polymorphic behavior. On the other hand, late binding is less effective than early binding, even though the difference may be

negligible. (In C++ on PCs, the difference between the late and the early binding is usually one machine instruction per method call.)

Any method that might be overridden in any of the derived classes should use the late binding.

Note.

In C++ and other object oriented languages in which the late binding must be declared, the classes containing at least one virtual method are called *polymorphic classes*. Classes without any virtual method are called *non-polymorphic classes*. In languages like Java, where all the methods use late binding by default, all the classes are polymorphic.

15.5.2 Algorithm of the Late Binding

In this subsection, some low level concepts will be discussed. They are not necessary for understanding the basic concepts of the OOP, but they can give you a better insight in it.

We will explore one the common way of the late binding implementation, i.e., of the determination of the actual type of the instance for which the method is called.

This is based on the so called *virtual method tables*. The virtual method table (VMT) is the hidden class data member that is part of any polymorphic class. Any polymorphic class contains exactly one VMT. (The hidden class member is a class member the programmer does not declare – the compiler adds it automatically.)

The VMT is an array containing pointers to all the virtual method implementations of the class. Any derived class has its own VMT that contains pointers to all the virtual methods (even those that are not overridden in this class). The pointers to the virtual methods in the VMT of the derived class are stored in the same order as the pointers to corresponding methods in the base class.

Any instance of the polymorphic class contains another hidden data member – the pointer to the VMT of the class. This data member is stored in all the instances at the same place – e.g., at the beginning.

The method call is performed in the following way:

1. The program takes the instance, for which the method is called.
2. In the instance, it finds the pointer to the VMT. It's position does not depend on the type of the instance.
3. In the VMT, it finds the pointer to the method called. In all the VMTs, this pointer is stored in the same entry; e.g., the pointer to the `Interact()` method might be in the VMT of the `Particle` class and in the VMTs of all the classes derived from the `Particle` class in the first entry.
4. The program uses this pointer to call the method of the actual class of the instance.

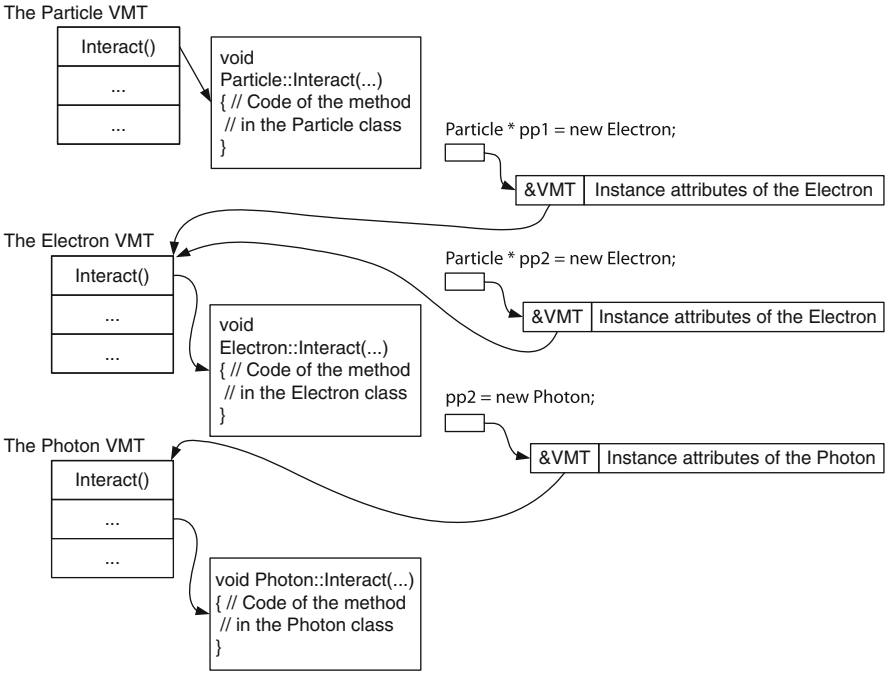


Fig. 15.4 Typical implementation of the late binding

Figure 15.4 illustrates this process for the `Particle` base class and two derived classes. The values stored in the VMT are set usually at the start of the program or when the class is loaded to the memory. The values of the pointer to the VMT in the instances are set automatically by the constructor.

15.5.3 Abstract Class

In some cases, the base class represents such an abstract concept that some operations with instances of this class cannot be implemented. Nevertheless, at least the stub of the corresponding method should be present in the class, because this class is used as a base class and determines the common interface for all the derived classes.

Such a class is called the *abstract class* and such an operation is called the *abstract method* in OOP. Abstract methods have usually no implementation (no method body).

It is not allowed to create instances of the abstract classes and it is not allowed to call the abstract methods. It is of course possible to define pointers or references to abstract classes.

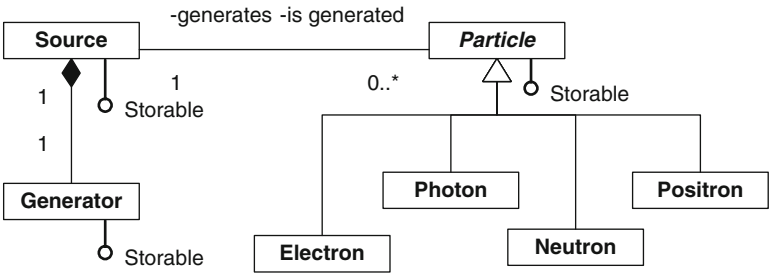


Fig. 15.5 The classes implementing the `Storable` interface

The abstract classes serve as base classes. If the derived class does not implement any of the inherited abstract methods, it will be abstract like the base class. The abstract class:

- Defines the interface of the derived classes.
- Provides the implementation of non-polymorphic methods for the derived classes.
- Offers a default implementation of non-abstract polymorphic (virtual) methods.

Note that the abstract class names are italicized in the UML class diagrams – see e.g., the `Particle` class in Fig. 15.5.

Example 10. Consider the `Particle` class defined above. How could the `Interact()` method be implemented?

For the derived classes, the situation is clear: If it is, e.g., the `Photon`, the interaction could be the photoelectric effect, the Compton scattering, or some other interaction known to particle physicists; probabilities of these phenomena are determined according to their total effective cross sections. For the other derived classes, there are other well defined possibilities that can be expressed in the program code.

However, there is no general interaction, that could be used to implement the `Interact()` method of the general `Particle`. On the other hand, this method must be declared in the `Particle` class as the part of the common interface of derived classes. If we omit it, the statement

```
store[i] -> Interact(detector);
```

will not compile, because `store[i]` is the pointer to the `Particle` class that does not contain such a method.

Thus the `Interact()` method should be declared abstract (in C++, the abstract methods are called *pure virtual methods*). In the following revision of the `Particle` class, we omit all parts of that class that are unchanged.

```
// Particle as an abstract class
class Particle
{
public:
```

```
    // Pure virtual method (abstract method)
    virtual void Interact(Detector *aDetector) = 0;
    // All other public members as before
protected:
    // All data members as before
};
```

15.5.4 Interfaces

The term *interface* is used in two different meanings in OOP:

1. It is the set of the public methods and attributes of the object. This is similar to the module interface in the modular programming.
2. It denotes a named set of methods and constants. This set may be empty and is defined independently on any class.

We will discuss the second meaning of this term in this subsection.

Interface as a Set of Methods

We have defined the interface as a named set of methods and constants. The interface represents a way to achieve the polymorphic behavior; it is an alternative to inheritance. This concept is relatively new in OOP, it was first widely used in the Java language.

In languages that support interfaces, any class may declare that it *implements* the given interface. This means that the class will supply the implementations (bodies) of the methods listed in the interface.

Note the terminological difference: Even though the interfaces are syntactically similar to the classes that contain only public abstract methods, they are not *inherited*, they are *implemented*. In programming languages, that support interfaces, any class may implement many interfaces, even if the language does not support multiple inheritance.

The interface represents the type. If the class `C` implements the interfaces `I1` and `I2`, any instance of this class is an instance of the type `C` as well as it is an instance of the type `I1` and of the type `I2`.

The interface is usually represented by a small circle connected to the class icon in the UML class diagrams (see Fig. 15.5). It may also be represented by a class-like icon marked by the "`interface`" label ("stereotype"). Implementation of the interface is represented by a dashed arrow pointing to the implementing class (see Fig. 15.7).

Interfaces in C++

We have chosen C++ as the language of examples in this chapter, thus it is necessary to cover briefly the interfaces in this language. C++ does not support interfaces directly; nevertheless, interfaces may be fully simulated by abstract classes that contain only public abstract (pure virtual) methods, and the interface implementation may be substituted by the inheritance. This will be demonstrated by the following example.

Example 11. The Monte Carlo simulation may be time-consuming and it would be convenient to have the possibility to store the status of the simulation into a file, so that the computation might be interrupted and continued later.

It is clear that all the generated but not yet processed particles should be stored. The status of the particle source, and consequently the status of the random number generator, should be stored, too. This is necessary especially for debugging, because it ensures that we could get the same sequence of random numbers in repeated runs of the program, even if they are interrupted.

It follows that we have at least two different object types belonging to different class hierarchies that have a common feature – they will be stored in a file and later will be restored into their original state. It follows that all the classes involved should have suitable methods, e.g., `store()` and `load()`.

The simulated experiment is represented by the `Experiment` class in the program and to store the experiment status is the task of this class; it follows we would like to implement the `storeObject()` method to store objects passed as arguments in this class. Thus all the parameters – all the objects stored – should be of the same type.

The solution of this dilemma – the method requires objects of the same type as parameters, but we have objects of at least two distinct types belonging to different class hierarchies – is to use a suitable interface that contains the `store()` and `load()` methods. We will use the `Storable` identifier for this interface. The `Source`, `Generator` and `Particle` classes should be modified as follows:

```
// Interface simulation in C++
class Storable
{
public:
   virtual void store(ostream&) = 0;
   virtual void load(istream&) = 0;
};

class Generator: public Storable   // Interface
                                      implementation
{
public:
   virtual void store(ostream& out)
   {/* Store the generator */}
```

```
    virtual void load(istream& in)
    {/* Read the generator and reconstruct it */}
    // ... Other methods and attributes as before
};

class Source: public Storable       // Interface
                                       implementation
{
public:
    virtual void store(ostream& out)
    {/* Store the source */}
    virtual void load(istream& in)
    {/* Read the source from the file and reconstruct
        it*/}
    // ... Other methods and attributes as before
};

class Particle: public Storable  // Interface
                                    implementation
{
public:
    virtual void store(ostream& out)
    {/* Store the particle */}
    virtual void load(istream& in)
    {/* Read the particle from the file and
        reconstruct it */}
    // ... Other methods and attributes as before
};
```

(`ostream` and `istream` are base classes for output and input data streams in the standard C++ library.) Figure 15.5 shows the revised UML class diagram of these classes.

Note that the `Particle` class is abstract, thus it needs not override the methods of the `Storable` interface. The classes representing the concrete particle types, `Electron` etc., inherit the implementation of the `Storable` interface; thus it is not necessary to declare this fact. Of course, if a derived class is not abstract, it must inherit or override the implementation of the methods of this interface.

Implementation of the `Storable` interface allows us to define the method `Experiment::storeObject()` as follows:

```
void Experiment::storeObject(Storable& obj, ostream&
    out)
{
    obj.store(out)
}
```

`Storable` interface serves as the common type of all the storable objects – particles as well as random number generators – in the program. This gives us the possibility to treat all these objects in our program in a uniform way, even if they do not belong to the same hierarchy of inheritance.

15.6 More About Inheritance

Inheritance can be easily misused and this is often counterproductive. Poorly designed inheritance hierarchies lead to programs that are difficult to understand, contain hard-to-find errors, and are difficult to maintain. In this section, we give some typical examples.

15.6.1 Substitution Principle

In Sect. 15.4, *Inheritance*, we have seen that any instance of any derived class may be used in place of an instance of the base class. This is sometimes called *the substitution principle*.

As far as we have seen, this is a syntactic rule: If you follow it, the program compiles. But we already know that the derived class should be a specialization of the base class, if we want to use the inheritance reasonably. Let's investigate more deeply, what it means.

"Technical" Inheritance

This problem is similar to the problem we have seen in Sect. 15.4.4. We have two related classes, say `A` and `B`, and class `B` contains all the members of `A` and some additional ones. Is it reasonable to use `A` as a base class of `B`?

Of course, this situation indicates, that `B` *might be* really derived from `A`. But this is only an indication that cannot replace the IS A – HAS A test. In 15.4.4, we have seen an example that leads to object composition. Here we give another example that will be solved by inheritance.

Example 12. Consider the particles in our Monte Carlo simulation. The interaction of electrically charged particles in the detector substantially differs form the interaction of uncharged particles, thus it would be convenient to split the class of all the particles into two subclasses, one for the uncharged particles and the other for the charged ones.

The class representing the charged particles contains the same data members as the class representing the uncharged particles plus the `charge` attribute and the access methods `setCharge()` and `getCharge()` to manipulate the charge.

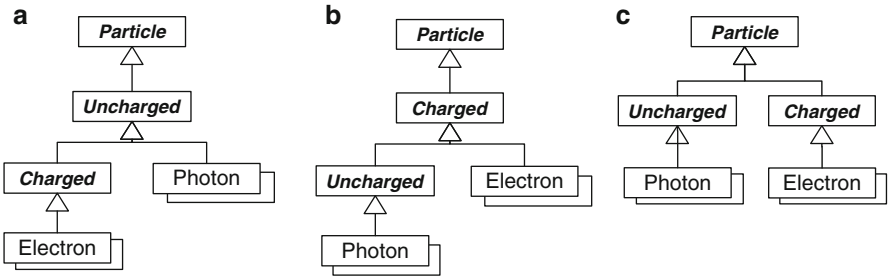


Fig. 15.6 Class hierarchies discussed in subsections 15.6.1 and 15.6.2. Only (c) is correct

This might lead to the idea to define the `Uncharged` class representing the uncharged particles and use it as a base for the `Charged` class representing the charged particles. These two classes will serve as base classes for the classes representing concrete particle types (Fig. 15.6(a)).

This class hierarchy design is incorrect and leads to problems in the program. Suppose the following two declarations:

```
list<Uncharged*> listOfUncharged;
Electron e;    //  Electron is charged particle
```

The `listOfUncharged` variable is a double-linked list of pointers to the `Uncharged` instances. If the `Charged` class were derived from the `Uncharged` class, it would be possible to insert any charged particle into this container of uncharged ones. The following statement would compile and run (of course, the results would be unpredictable):

```
ListOfUncharged.push_back(&e); // It compiles...
```

The cause of this problem is evident – *the charged particle is not a special case of the uncharged particle* (the IS A test), so this inheritance is not applicable.

"Logical" Inheritance

Now we will see that the IS A – HAS A test may be insufficient in some cases. First, consider the following example.

Example 13. We will continue with the analysis of the Monte Carlo simulation of the charged and uncharged particles. The uncharged particles may be considered as a special case of the charged particles with the electric charge set to zero. Consequently, it seems to be logical to derive the `Uncharged` class from the `Charged` class (Fig. 15.6(b)).

However, no member of the base class may be excluded from the derived class in the inheritance process. Thus the derived class, `Uncharged`, will contain the `charge` attribute and both the access methods. In order to ensure that the charge

is zero, we have to override the `setCharge()` method so that it always sets the `charge` value to zero,

```
void Uncharged::setCharge(double ch) {
  charge = 0.0;          // Parameter value not used
}
```

Nevertheless, this construction may fail. Consider the following function:

```
void process(Charged& cp){
    const double chargeValue = 1e-23;
    cp.setCharge(chargeValue);
    assert(cp.getCharge() == chargeValue);
    // And some other code...
}
```

This is correct behavior of the `process()` function: It expects a charged particle, changes its charge to some predefined value and tests whether or not this change succeeded. If the argument is really a charged particle, it works.

However, the classes representing the uncharged particles, e.g., `Photon`, are derived from the `Uncharged` class and this class is derived from the `Charged` class, so the following code fragment compiles, but fails in the run time:

```
Photon p;        // Uncharged particle
process(p);      // Assertion fails...
```

This example shows, that even if the IS A test succeeds, it does not follow that the inheritance is necessarily the right choice. In this case, the overridden `setCharge()` method violates the contract of the base class method – it does not change the `charge` value.

15.6.2 Substitution Principle Revised

The preceding example demonstrates that in some situations the `Uncharged` class has significantly different behavior than the base class, and this leads to problems – even to run time errors.

This is the rule: Given the pointer or reference to the base class, if it is possible to distinguish, whether it points to an instance of the base class or of the derived class, the base class cannot be substituted by the derived class.

The conclusion is, that the substitution principle is more than a syntactic rule. This is a constraint imposed on derived classes, that requires, that the derived class instances must be programmatically indistinguishable from the base class instances, otherwise the derived class does not represent a subtype of the base class.

This conclusion has been originally formulated by Liskov (1988), Martin (1996) as follows:

> What is wanted here is something like the following substitution property: If for each object o_1 of type S there is an object o_2 of type T such that for all programs P defined in terms of T, the behavior of P is unchanged when o_1 is substituted for o_2, then S is subtype of T.

Example 14. Let's finish the analysis of the charged and uncharged particles problem. We have seen that the `Charged` and `Uncharged` classes may not be derived one from the other. To avoid both kinds of problems, it is necessary to split the hierarchy and to derive both classes directly from the `Particle` class:

```
// Proper Particle hierarchy
class Charged: public Particle { /* ... */ };
class Uncharged: public Particle { /* ... */ };
```

This class hierarchy is shown in the Fig. 15.6(c).

15.6.3 Inheritance and Encapsulation

In this subsection we demonstrate that the inheritance may lead to significant violation of the encapsulation, which may cause problems in the implementation of derived classes. We start with an example.

Example 15. The particles interact in the detector in different ways. Some of the interactions represent events that are subject to our investigation and need to be logged in the result file and further processed. (This means to record the particle type, energy, coordinates of the interaction etc.) But the events may appear in groups, so the `ResultFile` class will contain the methods `LogEvent()` and `LogEventGroup()`. The latter will get the vector containing data of several events as an argument. Suppose that both these methods are polymorphic.

At some later stage of the program development, we find that it is necessary to be aware of the total count of recorded events. The actual implementation of the `ResultFile` class does not support this feature and we cannot change it, e.g., because it is the part of some program library.

The solution seems to be easy – we derive a new class, `CountedResultFile`, based on the `ResultFile`. The implementation could be as follows:

```
class CountedResultFile: public ResultFile
{
public:
    virtual void LogEvent(Event *e)
    {
        ResultFile::LogEvent(e);
        count++;
    }
    virtual void LogEventGroup(vector<Event*> eg)
    {
```

```
            ResultFile::LogEventGroup(eg);
            count += eg.size();
        }
    private:
        int count;
    };
```

The overridden methods simply call the base class methods to log the events and then increase the `count` of the recorded events.

It may happen that we find that the `LogEventGroup()` method increases the count of recorded events incorrectly: After the call

```
LogFile *clf = new CountedLogFile;
clf -> LogEventGroup(eg);   // (*)
```

the `count` value increases by twice the number of the events in `eg`.

The reason might be that the implementation of the `LogEventGroup()` method internally calls the `LogEvent()` method in a loop. This is what happens:

1. The (`*`) statement calls the `LogEventGroup()` method. This is a polymorphic method, so the `CountedResultFile::LogEventGroup()` method is called.
2. This method calls the base class `LogEventGroup()` method.
3. The base class method calls the `LogEvent()` method in a loop. *But because these methods are polymorphic, the method of the actual type, i.e. the* `ComputedResultFile::LogEvent()` *method is called.*
4. This method calls the base class method to record the event and increases the count of events. After that it returns to the `CountedResultFile::LogEventGroup()` method. This method increases the event count once again.

To implement the derived class properly, we need to know that the `ResultFile::LogEventGroup()` method internally calls the `ResultFile::LogEvent()` method. *But this is an implementation detail, not the part of the contract of the methods of the* `ResultFile` *class* – it will probably not be mentioned in the documentation to the library containing the `ResultFile` class.

Solution

This problem may easily be avoided by using interfaces and object composition (cf. class diagram in Fig. 15.7); it is necessary to use another design of the `ResultFile` class, as well as another design of the `CountedResultFile` class.

First we design the `ResultFileInterface` interface as follows:

```
class ResultFileInterface {
public:
     virtual void LogEvent(Event *e) = 0;
```

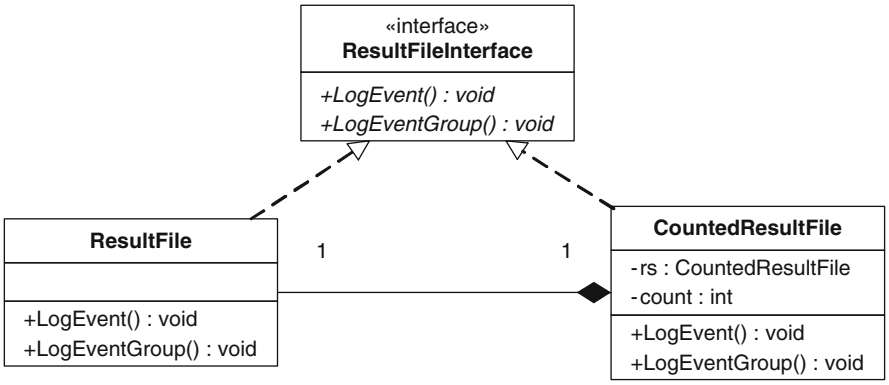


Fig. 15.7 Class diagram of the correct design. Only interface methods and corresponding attributes are shown

```
    virtual void LogEventGroup(vector<Event*> eg) = 0;
};
```

The class `ResultFile` will implement this interface:

```
class ResultFile: public ResultFileInterface {
 // Implementation as before
};
```

Now, `CountedResultFile` may be designed as an independent class that implements the `ResultFileInterface` and uses the `ResultFile` as an attribute:

```
class CountedResultFile: public ResultFileInterface {
public:
    virtual void LogEvent(Event *e)
    {
        rs.LogEvent(e);
        count++;
    }
    virtual void LogEventGroup(vector<Event*> eg)
    {
        rs.LogEventGroup(eg);
        count += eg.size();
    }
private:
    int count;
    ResultSet rs;
};
```

The problem may not appear, because the `CountedResultFile` is not derived from the `ResultFile` now. Nevertheless, the classes may be treated polymorphically, i.e. instances of the `CountedResultFile` may be used instead of instances of the `ResultFile`, if they are used as instances of the `ResultFileInterface` interface.

15.7 Structure of the Object Oriented Program

We conclude this chapter by describing shortly the typical structure of the OOP program.

As we have seen in previous sections, an OOP program consists of objects that collaborate by messages. In this structure, one object must play the role of a *starting object*. This means that one of the methods of this object will be called as the program starts. The starting object typically creates other objects in the program and manages the their lifetime.

All the other objects represent various parts of the problem solved and are responsible for the associated resource management, computation, etc.

Example 16. Here we finish the Monte Carlo simulation of an experiment with particles. We have already mentioned the `Experiment` class covering the application as a whole. The only instance of this class in the program will be the starting object. The `Experiment` class will contain the public method `run()` that will represent the run of the experiment.

As we are in C++, our program must have the `main()` function where the program starts. It will create the starting object and let it run:

```
int main() {              // Create the starting object
    Experiment().Run(); // and run it
}
```

The `Experiment` class could be defined as follows:

```
class Experiment{
public:
    Experiment();         // Constructor
    void Run();           // Run the experiment
private:
    Detector *detector; // Detector in this experiment
    Source *source;     // Particle source
    ResultFile *rfile;  // Result file
 };
```

The `Experiment::Experiment()` constructor reads the input data (e.g., cross sections describing the probabilities of the interactions) and creates and initializes the attributes (particle source, result file etc.).

The `Experiment::Run()` method does the work – it starts particle emission in the source using the appropriate method of the `Source` class, determines, whether the given particle hits the detector using the appropriate `Detector` method, records the results of the detection into the `ResultFile` instance and in the end processes the results using the appropriate `ResultFile` methods.

Class diagram of the program at the level we have given here is presented in Fig. 15.8.

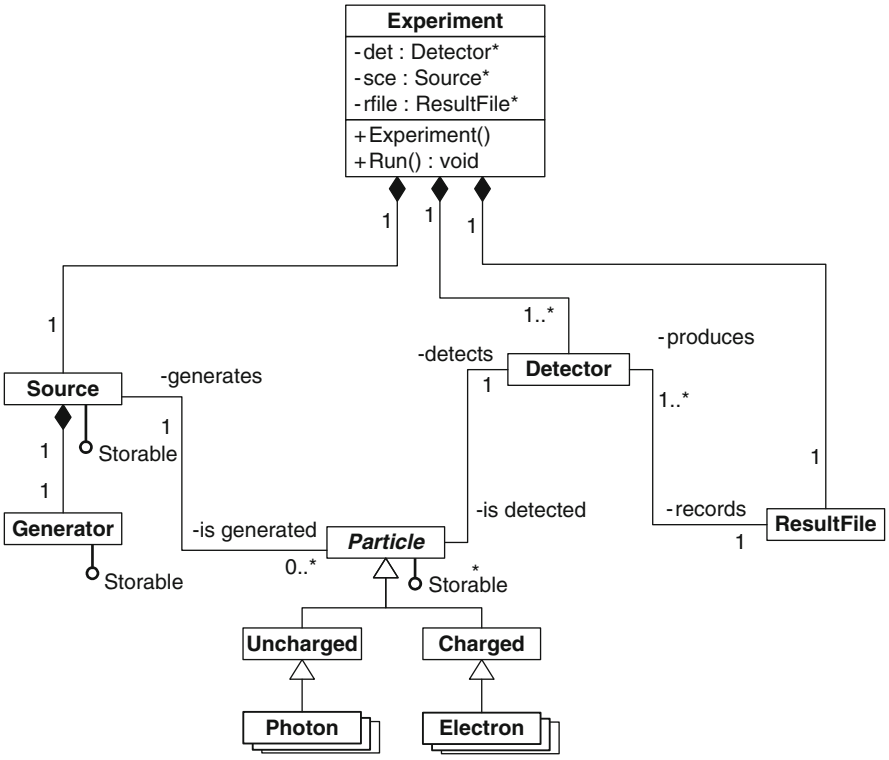


Fig. 15.8 Basic structure of the simulation program

Note that this is top-level design only. The extent of this chapter allows us to demonstrate only the beginning of the object oriented approach to the example.

15.8 Conclusion

Attempts to formalize the process of object oriented analysis and design have been made since the beginning of the OOP. A widely used approach is described in Booch (1993).

In object oriented design some common problems – or tasks – may appear in many different situations. Reusable solutions of these problems are called *Design Patterns*. A well known example of design pattern is the *Singleton* – the class that may have at most one instance. Another example is the *Responsibility Chain*; this design pattern solves the problem how to find the proper processing of varying data, even if the way of processing may dynamically change. The *Observer* design pattern

is used, if some objects in the program need to respond to some events arising in other objects.

The idea of design patterns and the 23 most common design patterns in OOP are described in Gamma (1995).

References

Booch, G.: Object-Oriented Analysis and Design with Applications. Wesley, New York (1993)
Booch, G., Rumbaugh, J., Jacobson, I.: The Unified Modeling Language User Guide. Wesley Professional, New York (2005)
Gamma, E., Helm, R., Johnson R., Vlissides, J.: Design Patterns. Wesley, New York (1995)
Liskov, B.: Data Abstraction and Hierarchy. SIGPLAN Notices **23**, 5 (1988)
Martin, R.C.: The Liskov Substitution Principle. The C++ Report (1996)
Meyer, B.: Object-Oriented Software Construction. Prentice Hall, New York (1997)
Stroustrup, B.: The C++ Programming Language. Special edition. Wesley, New York (2000)
Stroustrup, B.: What is C++0x? Available at: http://www2.research.att.com/~bs/papers.html (citation July 2010)

Part III
Statistical Methodology

Chapter 16
Model Selection

Yuedong Wang

16.1 Introduction

The need for model selection arises when a data-based choice among competing models has to be made. For example, for fitting parametric regression (linear, non-linear and generalized linear) models with multiple independent variables, one needs to decide which variables to include in the model (Chaps. III.7, III.8 and III.12); for fitting non-parametric regression (spline, kernel, local polynomial) models, one needs to decide the amount of smoothing (Chaps. III.5 and III.10); for unsupervised learning, one needs to decide the number of clusters (Chaps. III.13 and III.16); and for tree-based regression and classification, one needs to decide the size of a tree (Chap. III.14).

Model choice is an integral and critical part of data analysis, an activity which has become increasingly more important as the ever increasing computing power makes it possible to fit more realistic, flexible and complex models. There is a huge literature concerning this subject (Burnham and Anderson 2002; George 2000; Linhart and Zucchini 1986; Miller 2002) and we shall restrict this chapter to basic concepts and principles. We will illustrate these basics using a climate data, a simulation, and two regression models: parametric trigonometric regression and non-parametric periodic splines. We will discuss some commonly used model selection methods such as Akaike's AIC (Akaike 1973), Schwarz's BIC (Schwarz 1978), Mallow's C_p (Mallows 1973), cross-validation (CV) (Stone 1974), generalized cross-validation (GCV) (Craven and Wahba 1979) and Bayes factors (Kass and Raftery 1995). We do not intend to provide a comprehensive review. Readers may find additional model selection methods in the following chapters.

Y. Wang (✉)
Department of Statistics and Applied Probability, University of California, Santa Barbara, CA, USA
e-mail: yuedong@pstat.ucsb.edu

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_16,

Let $\mathcal{M} = \{M_\lambda,\ \lambda \in \Lambda\}$ be a collection of candidate models from which one will select a model for the observed data. λ is the model index belonging to a set Λ which may be finite, countable or uncountable.

Variable selection in multiple regression is perhaps the most common form of model selection in practice. Consider the following linear model

$$y_i = \mathbf{x}_i^\top \beta + \epsilon_i, \qquad i = 1, 2, \cdots, n, \tag{16.1}$$

where $\mathbf{x}_i$ are vectors of m independent variables, β is a m-vector of parameters, and ϵ_i are random errors. Often a large number of independent variables are investigated in the model (16.1) and it is likely that not all m variable are important. Statistical inferences can be carried out more efficiently with smaller models. The goal of the variable selection is to find a subset of these m independent variables which is optimal in a certain sense. In this case, Λ is the collection of all 2^m subsets and λ is any particular subset.

For illustration, we will use part of a climate data set downloaded from the Carbon Dioxide Information Analysis Center at http://cdiac.ornl.gov/ftp/ndp070. The data consists of daily maximum temperatures and other climatological variables from 1,062 stations across the contiguous United States. We choose daily maximum temperatures from the station in Charleston, South Carolina, which has the longest records from 1871 to 1997. We use records in the year 1990 as observations. Records from other years provide information about the population. To avoid correlation (see Sect. 16.6) and simplify the presentation, we divided 365 days in 1990 into 73 five-day periods. The measurements on the third day in each period is selected as observations. Thus the data we use in our analyses is a subset consisting of every fifth day records in 1990 and the total number of observations $n = 73$. For simplicity, we transform the time variable t into the interval [0, 1]. The data is shown in the left panel of Fig. 16.1.

Our goal is to investigate how maximum temperature changes over time in a year. Consider a regression model

$$y_i = f(t_i) + \epsilon_i, \qquad t_i = i/n, \qquad i = 1, \cdots, n, \tag{16.2}$$

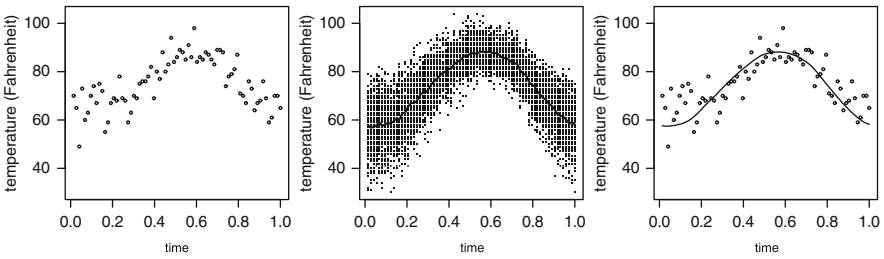


Fig. 16.1 *Left*: 73 observations in the year 1990. *Middle*: observations on the same 73 days from 1871–1997. Averages are marked as the *solid line*. *Right*: 73 observations in the year 1990 (*points*) and a periodic spline fit (*line*) to the average temperatures in the middle panel

where y_i is the observed maximum temperature at time t_i in Fahrenheit, f is the mean temperature function and ϵ_i's are random fluctuations in the year 1990. We assume that ϵ_i's are independent and identically distributed with mean zero and variance σ^2. Note that even though model (16.2) is very general, certain model assumptions (choices) have already been made implicitly. For example, the random fluctuations are assumed to be additive, independent and homogeneous. Violations of these assumptions such as independence may have severe impacts on model selection procedures (Sect. 16.6).

In the middle panel of Fig. 16.1, we plot observations on the same selected 73 days from 1871 to 1997. Assuming model (16.2) is appropriate for all years, the points represent 127 realizations from model (16.2). The averages reflect the true mean function f and the ranges reflect fluctuations. In the right panel, a smoothed version of the averages is shown, together with the observations in 1990. One may imagine that these observations were generated from the smoothed curve plus random errors. Our goal is to recover f from the noisy data. Before proceeding to estimation, one needs to decide a model space for the function f. Intuitively, a larger space provides greater potential to recover or approximate f. At the same time, a larger space makes model identification and estimation more difficult (Yang 1999). Thus the greater potential provided by the larger space is more difficult to reach. One should use as much prior knowledge as possible to narrow down the choice of model spaces. Since f represents mean maximum temperature in a year, we will assume that f is a periodic function.

Trigonometric Regression Model. It is a common practice to fit the periodic function f using a trigonometric model up to a certain frequency, say λ, where $0 \le \lambda \le K$ and $K = (n-1)/2 = 36$. Then the model space is

$$M_\lambda = \text{span}\left\{1, \sqrt{2}\sin 2\pi\nu t, \sqrt{2}\cos 2\pi\nu t, \ \ \nu = 1, \cdots, \lambda\right\}. \tag{16.3}$$

The order λ is unknown in most applications. Thus one needs to select λ among $\Lambda = \{0, 1, \cdots, K\}$ where $M_0 = \text{span}\{1\}$. For a fixed λ, we write the model M_λ as

$$y_i = \beta_1 + \sum_{\nu=1}^{\lambda}\left(\beta_{2\nu}\sqrt{2}\sin 2\pi\nu t + \beta_{2\nu+1}\sqrt{2}\cos 2\pi\nu t\right) + \epsilon_i, \quad i = 1, \cdots, n, \tag{16.4}$$

or in a matrix form

$$\mathbf{y} = X_{\lambda\,\lambda} + \text{›},$$

where $\mathbf{y} = (y_1, \cdots, y_n)^\top$,

$$X_\lambda = \begin{pmatrix} 1 & \sqrt{2}\sin 2\pi t_1 & \sqrt{2}\cos 2\pi t_1 & \cdots & \sqrt{2}\sin 2\pi\lambda t_1 & \sqrt{2}\cos 2\pi\lambda t_1 \\ 1 & \sqrt{2}\sin 2\pi t_2 & \sqrt{2}\cos 2\pi t_2 & \cdots & \sqrt{2}\sin 2\pi\lambda t_2 & \sqrt{2}\cos 2\pi\lambda t_2 \\ \vdots & \vdots & \vdots & \cdots & \vdots & \vdots \\ 1 & \sqrt{2}\sin 2\pi t_n & \sqrt{2}\cos 2\pi t_n & \cdots & \sqrt{2}\sin 2\pi\lambda t_n & \sqrt{2}\cos 2\pi\lambda t_n \end{pmatrix}$$

is the design matrix, $\beta = (\beta_1, \cdots, \beta_{2\lambda+1})^\top$ and $\epsilon = (\epsilon_1, \cdots, \epsilon_n)^\top$. The coefficients β are estimated by minimizing the least squares (LS)

$$\min_{\beta} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(y_i - \beta_1 - \sum_{\nu=1}^{\lambda} \left(\beta_{2\nu} \sqrt{2} \sin 2\pi \nu t_i + \beta_{2\nu+1} \sqrt{2} \cos 2\pi \nu t_i \right) \right)^2 \right\} \tag{16.5}$$

Since design points are equally spaced, we have the following orthogonality relations:

$$\begin{aligned} \frac{2}{n} \sum_{i=1}^{n} \cos 2\pi \nu t_i \cos 2\pi \mu t_i &= \delta_{\nu,\mu}, \quad 1 \le \nu, \ \mu \le K, \\ \frac{2}{n} \sum_{i=1}^{n} \sin 2\pi \nu t_i \sin 2\pi \mu t_i &= \delta_{\nu,\mu}, \quad 1 \le \nu, \ \mu \le K, \\ \frac{2}{n} \sum_{i=1}^{n} \cos 2\pi \nu t_i \sin 2\pi \mu t_i &= 0, \qquad 1 \le \nu, \ \mu \le K, \end{aligned} \tag{16.6}$$

where $\delta_{\nu,\mu}$ is the Kronecker delta. Thus columns of the design matrix are orthogonal. That is, $X_\lambda^\top X_\lambda = n I_{2\lambda+1}$ where $I_{2\lambda+1}$ is an identity matrix of size $2\lambda + 1$. Let X_K be the design matrix of the largest model M_K. Then $X_K/\sqrt{n}$ is an orthonormal matrix. Define the discrete Fourier transformation $\tilde{\mathbf{y}} = X_K^\top \mathbf{y}/n$. The LS estimate of β is $\widehat{\beta} = (X_\lambda^\top X_\lambda)^{-1} X_\lambda^\top \mathbf{y} = X_\lambda^\top \mathbf{y}/n = \tilde{\mathbf{y}}_\lambda$, where $\tilde{\mathbf{y}}_\lambda$ consists of the first $2\lambda + 1$ elements of $\tilde{\mathbf{y}}$. More explicitly,

$$\begin{aligned} \widehat{\beta}_0 &= \frac{1}{n} \sum_{i=1}^{n} y_i = \tilde{y}_1, \\ \widehat{\beta}_{2\nu} &= \frac{\sqrt{2}}{n} \sum_{i=1}^{n} y_i \sin 2\pi \nu t_i = \tilde{y}_{2\nu}, \qquad 1 \le \nu \le \lambda, \\ \widehat{\beta}_{2\nu+1} &= \frac{\sqrt{2}}{n} \sum_{i=1}^{n} y_i \cos 2\pi \nu t_i = \tilde{y}_{2\nu+1}, \quad 1 \le \nu \le \lambda. \end{aligned} \tag{16.7}$$

Let $\widehat{f}_\lambda$ be the estimate of f where the dependence on λ is expressed explicitly. Then the fits

$$\widehat{\mathbf{f}}_\lambda \triangleq \left(\widehat{f}_\lambda(t_1), \cdots, \widehat{f}_\lambda(t_n) \right)^\top = X_\lambda \widehat{\beta} = P(\lambda) \mathbf{y}\,,$$

where

$$P(\lambda) = X_\lambda \left(X_\lambda^\top X_\lambda \right)^{-1} X_\lambda^\top = X_\lambda X_\lambda^\top / n \tag{16.8}$$

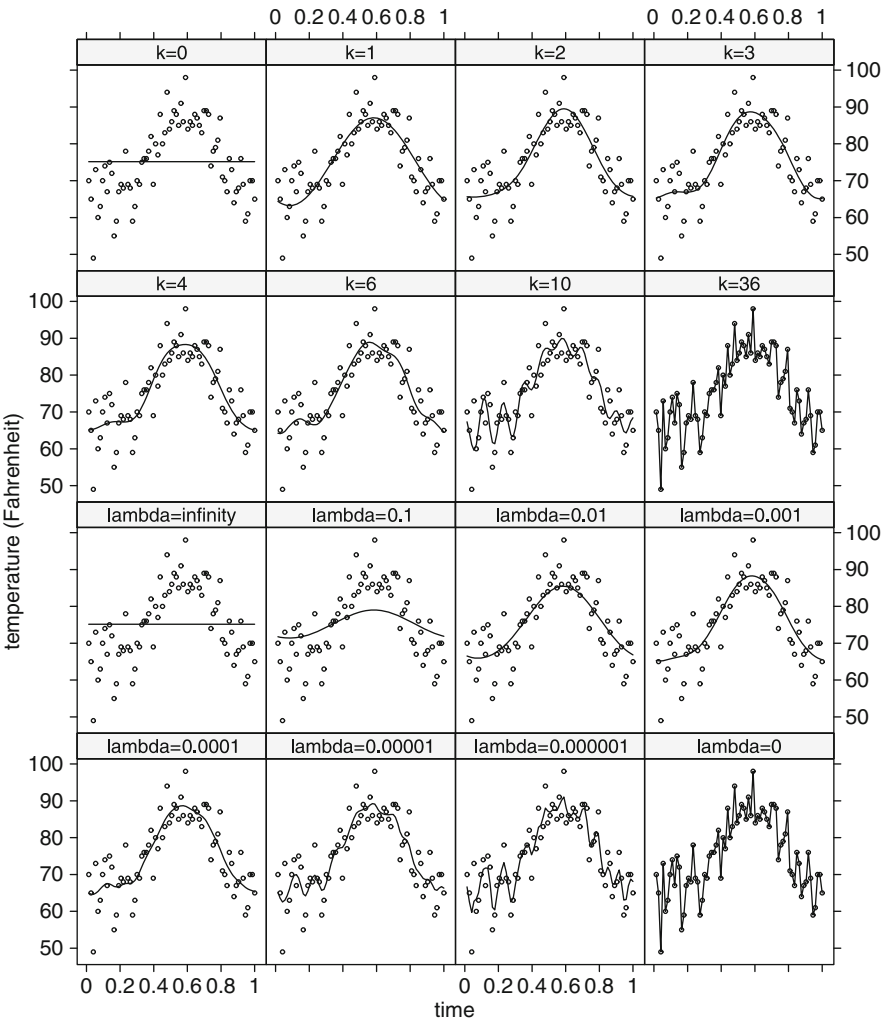


Fig. 16.2 *Circles* are observations. *Lines* are the estimated functions. *Top two rows* are fits from trigonometric models with frequencies indicated in the strips (we used k instead of λ for distinction). *Bottom two rows* are fits from the periodic spline with smoothing parameters indicated in the strips

is the projection matrix. Note that $P(K) = I_n$. Thus model M_K interpolates the data.

Fits for several λ (labeled as k in strips) are shown in the top two rows of Fig. 16.2. Obviously as λ increases from zero to K, we have a family of models ranging from a constant to interpolation. A natural question is that which model (λ) gives the "best" estimate of f.

Periodic Spline. In addition to the periodicity, it is often reasonable to assume that f is a smooth function of $t \in [0, 1]$. Specifically, we assume the following infinite dimensional space (Gu 2002; Wahba 1990)

$$W_2(per) = \Big\{ f : f \text{ and } f' \text{ are absolutely continuous,} \\ f(0) = f(1),\ f'(0) = f'(1), \int_0^1 (f''(t))^2 dt < \infty \Big\} \quad (16.9)$$

as the model space for f. A smoothing spline estimate of f is the minimizer of the following penalized LS (Gu 2002; Wahba 1990)

$$\min_{f \in W_2(per)} \left\{ \frac{1}{n} \sum_{i=1}^{n} (y_i - f(t_i))^2 + \lambda \int_0^1 (f''(t))^2 dt \right\}, \quad (16.10)$$

where the first part (LS) measures the goodness-of-fit, the second part is a penalty to the roughness of the estimate, and λ ($0 \le \lambda \le \infty$) is the so called *smoothing parameter* which controls the trade-off between the goodness-of-fit and the roughness. When $\lambda = 0$, there is no penalty to the roughness of the estimate and the spline estimate interpolates the data. When $\lambda = \infty$, the spline estimate is forced to be a constant. As λ varies from zero to infinity, we have a family of models ranging from interpolation to a constant parametric model. Thus λ can be considered as a model index in $\Lambda = [0, \infty]$. Fits with several choices of λ are shown in the bottom two rows of Fig. 16.2.

The exact solution to (16.10) can be found in Wahba (1990). To simplify the argument, as in Wahba (1990), we consider the following approximation of the original problem

$$\min_{f \in M_K} \left\{ \frac{1}{n} \sum_{i=1}^{n} (y_i - f(t_i))^2 + \lambda \int_0^1 (f''(t))^2 dt \right\}, \quad (16.11)$$

where $W_2(per)$ in (16.10) is replaced by M_K which is defined in (16.3) with $\lambda = K$. The following discussions hold true for the exact solution in the infinite dimensional spaces (Gu 2002; Wahba 1990). The approximation makes the following argument transparent and provides insights into smoothing.

Let

$$\widehat{f}_\lambda(t) = \widehat{\alpha}_1 + \sum_{\nu=1}^{K} \left(\widehat{\alpha}_{2\nu} \sqrt{2} \sin 2\pi\nu t + \widehat{\alpha}_{2\nu+1} \sqrt{2} \cos 2\pi\nu t \right)$$

be the solution to (16.11). Then $\widehat{\mathbf{f}}_\lambda \triangleq (\widehat{f}_\lambda(t_1), \cdots, \widehat{f}_\lambda(t_n))^\top = X_K \widehat{\boldsymbol{\alpha}}$, where $\widehat{\boldsymbol{\alpha}} = (\widehat{\alpha}_1, \cdots, \widehat{\alpha}_{2K+1})^\top$. The LS

$$\frac{1}{n} ||\mathbf{y} - \widehat{\mathbf{f}}_\lambda||^2 = \frac{1}{n} || \frac{1}{\sqrt{n}} X_K^\top (\mathbf{y} - \widehat{\mathbf{f}}_\lambda) ||^2 = || \frac{1}{n} X_K^\top \mathbf{y} - \frac{1}{n} X_K^\top X_K \widehat{\boldsymbol{\alpha}} ||^2 = ||\tilde{\mathbf{y}} - \widehat{\boldsymbol{\alpha}}||^2 .$$

Thus (16.11) reduces to the following ridge regression problem

$$(\widehat{\alpha}_1 - \tilde{y}_1)^2 + \sum_{\nu=1}^{K} \left((\widehat{\alpha}_{2\nu} - \tilde{y}_{2\nu})^2 + (\widehat{\alpha}_{2\nu+1} - \tilde{y}_{2\nu+1})^2 \right) + \lambda \sum_{\nu=1}^{K} (2\pi\nu)^4 \left(\widehat{\alpha}_{2\nu}^2 + \widehat{\alpha}_{2\nu+1}^2 \right) . \tag{16.12}$$

The solutions to (16.12) are

$$\begin{aligned} \widehat{\alpha}_1 &= \tilde{y}_1, \\ \widehat{\alpha}_{2\nu} &= \tilde{y}_{2\nu} / \left(1 + \lambda(2\pi\nu)^4\right), \qquad \nu = 1, \cdots, K, \\ \widehat{\alpha}_{2\nu+1} &= \tilde{y}_{2\nu+1} / \left(1 + \lambda(2\pi\nu)^4\right), \qquad \nu = 1, \cdots, K . \end{aligned} \tag{16.13}$$

Thus the periodic spline with equally spaced design points is essentially a low-pass filter: components at frequency ν are down-weighted by a factor of $1 + \lambda(2\pi\nu)^4$. The right panel of Fig. 16.3 shows how λ controls the nature of the filter: more high frequencies are filtered out as λ increases. It is clear from (16.7) and (16.13) that selecting an order for the trigonometric model may be viewed as hard thresholding and selecting the smoothing parameter for the periodic spline may be viewed as soft thresholding.

Let $D = \text{diag}(1, 1/(1+\lambda(2\pi)^4), 1/(1+\lambda(2\pi)^4), \cdots, 1/(1+\lambda(2\pi K)^4), 1/(1+\lambda(2\pi K)^4))$. Then $\widehat{\boldsymbol{\alpha}} = D\tilde{\mathbf{y}}$, and the fit

$$\widehat{\mathbf{f}}_\lambda = X_K \widehat{\boldsymbol{\alpha}} = \frac{1}{n} X_K D X_K^\top \mathbf{y} = A(\lambda)\mathbf{y} ,$$

where

$$A(\lambda) = X_K D X_K^\top / n \tag{16.14}$$

is the hat (smoother) matrix.

We choose the trigonometric regression and periodic spline models for illustration because of their simple model indexing: the first has a finite set of consecutive integers $\Lambda = \{0, 1, \cdots, K\}$ and the second has a continuous interval $\Lambda = [0, \infty]$.

This chapter is organized as follows. In Sect. 16.2, we discuss the trade-offs between the goodness-of-fit and model complexity, and the trade-offs between bias and variance. We also introduce mean squared error as a target criterion. In Sect. 16.3, we introduce some commonly used model selection methods: AIC, BIC, C_p, AIC_c and a data-adaptive choice of the penalty. In Sect. 16.4, we discuss the cross-validation and the generalized cross-validation methods. In Sect. 16.5, we discuss Bayes factor and its approximations. In Sect. 16.6, we illustrate potential effects of heteroscedasticity and correlation on model selection methods. The chapter ends with some general comments in Sect. 16.7.

16.2 Basic Concepts: Trade-Offs

We illustrate in this section that model selection boils down to compromises between different aspects of a model. Occam's razor has been the guiding principle for the compromises: the model that *fits observations sufficiently well* in *the least complex way* should be preferred. Formalization of this principle is, however, nontrivial.

To be precise on *fits observations sufficiently well*, one needs a quantity that measures how well a model fits the data. This quantity is often called the *goodness-of-fit* (GOF). It usually is the criterion used for estimation, after deciding on a model. For example, we have used the LS as a measure of the GOF for regression models in Sect. 16.1. Other GOF measures include likelihood for density estimation problems and classification error for pattern recognition problems.

To be precise on *the least complex way*, one needs a quantity that measures the complexity of a model. For a parametric model, a common measure of model complexity is the number of parameters in the model, often called the *degrees of freedom* (df). For a non-parametric regression model like the periodic spline, $trA(\lambda)$, a direct extension from its parametric version, is often used as a measure of model complexity (Hastie and Tibshirani 1990). $trA(\lambda)$ will also be refered to as the degrees of freedom. The middle panel of Fig. 16.3 depicts how $trA(\lambda)$ for the periodic spline depends on the smoothing parameter λ. Since there is an one-to-one correspondence between λ and $trA(\lambda)$, both of them are used as model index (Hastie and Tibshirani 1990). See Gu (1998) for discussions on some subtle issues concerning model index for smoothing spline models. For some complicated models such as tree-based regression, there may not be an obvious measure of model complexity (Ye 1998). In these cases the generalized degrees of freedom defined in Ye (1998) may be used. Section 16.3 contains more details on the generalized degrees of freedom.

To illustrate the interplay between the GOF and model complexity, we fit trigonometric regression models from the smallest model with $\lambda = 0$ to the largest

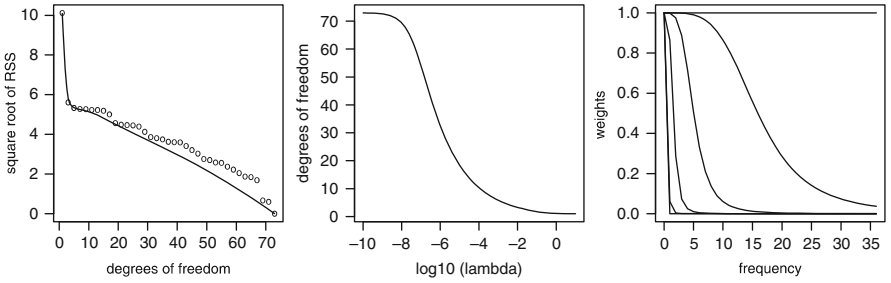


Fig. 16.3 *Left*: square root of RSS from the trigonometric model (*circles*) and periodic spline (*line*) plotted against the degrees of freedom. *Middle*: degrees of freedom of periodic spline plotted against the smoothing parameter on the logarithm base 10 scale. *Right*: weights of the periodic spline filter, $1/(1 + \lambda(2\pi\nu)^4)$, plotted as a function of frequency ν. Six curves from top down corresponds to six different λ: 0, 10^{-8}, 10^{-6}, 10^{-4}, 10^{-2} and ∞

model with $\lambda = K$. The square root of residual sum of squares (RSS) are plotted against the degrees of freedom ($= 2\lambda + 1$) as circles in the left panel of Fig. 16.3. Similarly, we fit the periodic spline with a wide range of values for the smoothing parameter λ. Again, we plot the square root of RSS against the degrees of freedom ($= trA(\lambda)$) as the solid line in the left panel of Fig. 16.3. Obviously, RSS decreases to zero (interpolation) as the degrees of freedom increases to n. RSS keeps declining almost linearly after the initial big drop. It is quite clear that the constant model does not fit data well. But it is unclear which model *fits observations sufficiently well*.

The previous example shows that the GOF and complexity are two opposite aspects of a model: the approximation error decreases as the model complexity increases. On the other hand, the Occam's razor suggests that simple models should be preferred to more complicated ones, other things being equal. Our goal is to find the "best" model that strikes a balance between these two conflicting aspects. To make the word "best" meaningful, one needs a target criterion which quantifies a model's performance. It is clear that the GOF cannot be used as the target because it will lead to the most complex model. Even though there is no universally accepted measure, some criteria are widely accepted and used in practice. We now discuss one of them which is commonly used for regression models.

Let $\widehat{f}_\lambda$ be an estimate of the function in model (16.2) based on the model space M_λ. Let $\mathbf{f} = (f(t_1), \cdots, f(t_n))^\top$ and $\widehat{\mathbf{f}}_\lambda = (\widehat{f}_\lambda(t_1), \cdots, \widehat{f}_\lambda(t_n))^\top$. Define the mean squared error (MSE) by

$$\mathrm{MSE}(\lambda) = \mathrm{E}\left(\frac{1}{n}||\widehat{\mathbf{f}}_\lambda - \mathbf{f}||^2\right).$$

We want the estimate $\widehat{f}_\lambda$ to be as close to the true function f as possible. Obviously MSE is the expectation of the Euclidean distance between the estimates and the true values. L_2 distance between $\widehat{f}_\lambda$ and f may also be used. MSE can be decomposed into two components:

$$\begin{aligned}
\mathrm{MSE}(\lambda) &= \frac{1}{n}\mathrm{E}||(\mathrm{E}\widehat{\mathbf{f}}_\lambda - \mathbf{f}) + (\widehat{\mathbf{f}}_\lambda - \mathrm{E}\widehat{\mathbf{f}}_\lambda)||^2 \\
&= \frac{1}{n}\mathrm{E}||\mathrm{E}\widehat{\mathbf{f}}_\lambda - \mathbf{f}||^2 + \frac{2}{n}\mathrm{E}(\mathrm{E}\widehat{\mathbf{f}}_\lambda - \mathbf{f})^\top(\widehat{\mathbf{f}}_\lambda - \mathrm{E}\widehat{\mathbf{f}}_\lambda) + \frac{1}{n}\mathrm{E}||\widehat{\mathbf{f}}_\lambda - \mathrm{E}\widehat{\mathbf{f}}_\lambda||^2 \\
&= \frac{1}{n}||\mathrm{E}\widehat{\mathbf{f}}_\lambda - \mathbf{f}||^2 + \frac{1}{n}\mathrm{E}||\widehat{\mathbf{f}}_\lambda - \mathrm{E}\widehat{\mathbf{f}}_\lambda||^2 \\
&\triangleq \mathrm{Bias}^2 + \mathrm{Variance}. \qquad (16.15)
\end{aligned}$$

The Bias^2 measures how well the model M_λ approximates the true function f, and the Variance measures how well the function can be estimated in M_λ. Usually larger model space leads to smaller Bias^2 but larger Variance. Thus, the MSE represents a trade-off between Bias^2 and Variance.

Another closely related target criterion is the average predictive squared error (PSE)

$$\mathrm{PSE}(\lambda) = \mathrm{E}\left(\frac{1}{n}||\mathbf{y}^+ - \widehat{\mathbf{f}}_\lambda||^2\right), \tag{16.16}$$

where $\mathbf{y}^+ = \mathbf{f} + \epsilon^+$ are new observations at the same design points, $\epsilon^+ = (\epsilon_1^+, \cdots, \epsilon_n^+)^\top$ are independent of ϵ, and ϵ_i^+'s are independent and identically distributed with mean zero and variance σ^2. PSE measures the performance of a model's prediction for new observations. We have

$$\mathrm{PSE}(\lambda) = \mathrm{E}\left(\frac{1}{n}||(\mathbf{y}^+ - \mathbf{f}) + (\mathbf{f} - \widehat{\mathbf{f}}_\lambda)||^2\right) = \sigma^2 + \mathrm{MSE}(\lambda)\,.$$

Thus PSE differs from MSE only by a constant σ^2. When justifying some criteria including the C_p in Sect. 16.3, we will ignore a constant σ^2. Thus the targets of these criteria are really PSE rather than MSE.

To illustrate the bias-variance trade-off, we now calculate MSE for the trigonometric regression and periodic spline models. For notational simplicity, we assume that $f \in M_K$:

$$f(t) = \alpha_1 + \sum_{\nu=1}^{K}\left(\alpha_{2\nu}\sqrt{2}\sin 2\pi\nu t + \alpha_{2\nu+1}\sqrt{2}\cos 2\pi\nu t\right). \tag{16.17}$$

Then $\mathbf{f} = X_K\boldsymbol{\alpha}$ where $\boldsymbol{\alpha} = (\alpha_1, \cdots, \alpha_n)^\top$. From the orthogonality relations (16.6), it is easy to check that $\boldsymbol{\alpha} = X_K^\top\mathbf{f}/n$, the discrete Fourier transformation of $\mathbf{f}$.

Bias-Variance Trade-Off for the Trigonometric Regression. X_λ consists of the first $2\lambda + 1$ columns of the orthogonal matrix X_K. Thus $X_\lambda^\top X_K = (nI_{2\lambda+1}, \mathbf{0})$. $\mathrm{E}(\widehat{\alpha}_\lambda) = X_\lambda^\top X_K\boldsymbol{\alpha}/n = \boldsymbol{\alpha}_\lambda$, where $\boldsymbol{\alpha}_\lambda$ consists of the first $2\lambda + 1$ elements in $\boldsymbol{\alpha}$. Thus $\widehat{\alpha}_\lambda$ is unbiased. Furthermore,

$$\begin{aligned}
\mathrm{Bias}^2 &= \frac{1}{n}||(I_n - P(\lambda))\mathbf{f}||^2 \\
&= \frac{1}{n}||\frac{1}{\sqrt{n}}X_K^\top\left(I_n - \frac{1}{n}X_\lambda X_\lambda^\top\right)X_K\boldsymbol{\alpha}||^2 \\
&= \frac{1}{n^2}||\left(nI_n - \begin{pmatrix} nI_{2\lambda+1} & \mathbf{0} \\ \mathbf{0} & \mathbf{0}\end{pmatrix}\right)\boldsymbol{\alpha}||^2 \\
&= \sum_{\nu=\lambda+1}^{K}\left(\alpha_{2\nu}^2 + \alpha_{2\nu+1}^2\right), \\
\mathrm{Variance} &= \frac{1}{n}\mathrm{E}||P(\lambda)(\mathbf{y} - \mathbf{f})||^2 = \frac{\sigma^2}{n}trP^2(\lambda) = \frac{\sigma^2}{n}(2\lambda + 1)\,.
\end{aligned}$$

Thus

$$\text{MSE}(\lambda) = \sum_{\nu=\lambda+1}^{K} \left(\alpha_{2\nu}^2 + \alpha_{2\nu+1}^2\right) + \frac{1}{n}\sigma^2(2\lambda+1)\,.$$

Remarks

1. Adding the λth frequency terms into the model space reduces the Bias^2 by the amount of $(\alpha_{2\lambda}^2 + \alpha_{2\lambda+1}^2)$ and increases the Variance by $2\sigma^2/n$.
2. The optimal model based on the MSE may not be the true model when $\sigma^2 > 0$.
3. Assuming $\alpha_{2\lambda}^2 + \alpha_{2\lambda+1}^2$ decreases with increasing λ, one should keep adding frequencies until
$$\alpha_{2\lambda}^2 + \alpha_{2\lambda+1}^2 \le 2\sigma^2/n. \tag{16.18}$$
4. Bias^2 does not depend on the sample size n and the Variance is inversely proportional to n. Thus as n increases, more frequencies should be included.

Bias-Variance Trade-Off for Periodic Spline. For the approximate periodic spline estimator, it is easy to check that $\text{E}(\tilde{\mathbf{y}}) = \boldsymbol{\alpha}$, $\text{Var}(\tilde{\mathbf{y}}) = \sigma^2 I/n$, $\text{E}(\widehat{\boldsymbol{\alpha}}) = D\boldsymbol{\alpha}$, $\text{Var}(\widehat{\boldsymbol{\alpha}}) = \sigma^2 D^2$, $\text{E}(\widehat{\mathbf{f}}_\lambda) = X_K D\boldsymbol{\alpha}$, and $\text{Var}(\widehat{\mathbf{f}}_\lambda) = \sigma^2 X_K D^2 X_K^\top/n$. Thus all coefficients are shrunk to zero except $\widehat{\alpha}_1$ which is unbiased. The amount of shinkage is controlled by the smoothing parameter λ. It is straightforward to calculate the Bias^2 and Variance in (16.15).

$$\begin{aligned}
\text{Bias}^2 &= \frac{1}{n}||X_K\boldsymbol{\alpha} - X_K D\boldsymbol{\alpha}||^2 = \frac{1}{n}||\frac{1}{\sqrt{n}}X_K^\top\left(X_K\boldsymbol{\alpha} - X_K D\boldsymbol{\alpha}\right)||^2 \\
&= ||(I-D)\boldsymbol{\alpha}||^2 = \sum_{\nu=1}^{K}\left(\frac{\lambda(2\pi\nu)^4}{1+\lambda(2\pi\nu)^4}\right)^2\left(\alpha_{2\nu}^2 + \alpha_{2\nu+1}^2\right)\,, \\
\text{Variance} &= \frac{1}{n^2}\sigma^2 tr\left(X_K D^2 X_K^\top\right) = \frac{\sigma^2}{n}tr(D^2) \\
&= \frac{\sigma^2}{n}\left(1 + 2\sum_{\nu=1}^{K}\left(\frac{1}{1+\lambda(2\pi\nu)^4}\right)^2\right).
\end{aligned}$$

Thus

$$\begin{aligned}
\text{MSE} = &\sum_{\nu=1}^{K}\left(\frac{\lambda(2\pi\nu)^4}{1+\lambda(2\pi\nu)^4}\right)^2\left(\alpha_{2\nu}^2 + \alpha_{2\nu+1}^2\right) \\
&+\frac{\sigma^2}{n}\left(1 + 2\sum_{\nu=1}^{K}\left(\frac{1}{1+\lambda(2\pi\nu)^4}\right)^2\right).
\end{aligned}$$

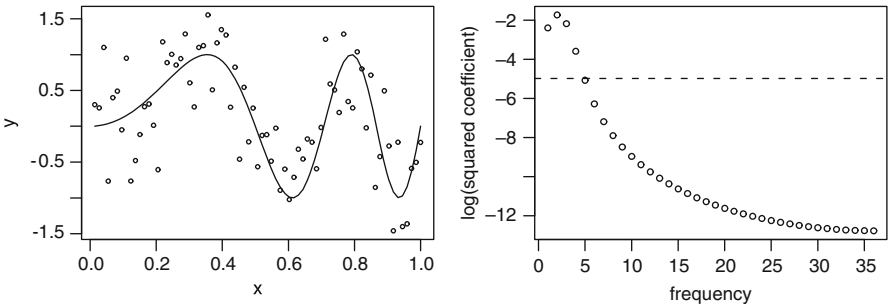


Fig. 16.4 *Left*: true function for the simulation (*line*) and observations (*circle*). *Right*: plots of b_ν, $\nu = 1, \cdots, K$, as *circles* and the threshold, $\log(2\sigma^2/n)$, as the *dashed line*

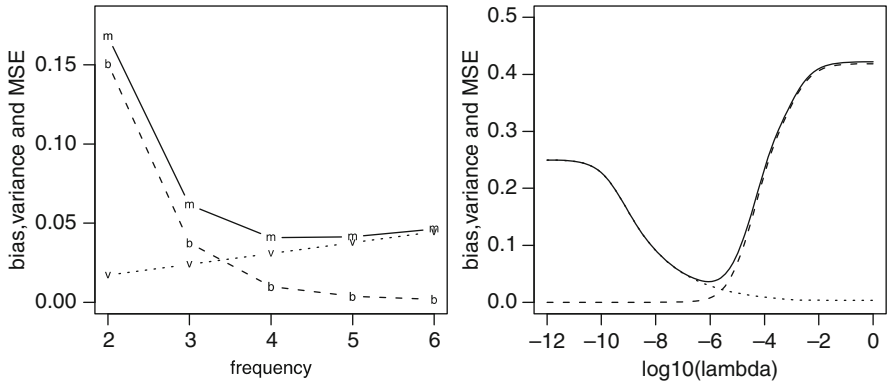


Fig. 16.5 Bias^2 (*dashed lines*), Variance (*dotted lines*) and MSE (*solid lines*) for trigonometric regression on the *left* and periodic spline on the *right*

It is easy to see that as λ increases from zero to infinity, the Bias^2 increases from zero to $\sum_{\nu=1}^{K}(\alpha_{2\nu}^2 + \alpha_{2\nu+1}^2)$ and the Variance decreases from σ^2 to σ^2/n.

To calculate MSE, one needs to know the true function. We use the following simulation for illustration. We generate responses from model (16.2) with $f(t) = \sin(4\pi t^2)$ and $\sigma = 0.5$. The same design points in the climate data is used: $t_i = i/n$, $i = 1, \cdots, n$ and $n = 73$. The true function and responses are shown in the left panel of Fig. 16.4. We compute $b_\nu = \log(a_{2\nu}^2 + a_{2\nu+1}^2)$, $\nu = 1, \cdots, K$. b_ν represents the contribution from frequency ν. In the right panel of Fig. 16.4, we plot b_ν against frequency ν with the threshold, $\log(2\sigma^2/n)$, marked as the dashed line. Except for $\nu = 1$, b_ν decreases as ν increases. Values of b_ν are above the threshold for the first four frequencies. Thus the optimal choice is $\nu = 4$.

Bias^2, Variance and MSE are plotted against frequency ($\log_{10}(\lambda)$) for trigonometric regression (periodic spline) in the left (right) panel of Fig. 16.5. Obviously, as the frequency (λ) increases (decreases), the Bias^2 decreases and the Variance increases. The MSE represents a balance between Bias^2 and Variance. For the trigonometric regression, the minimum of the MSE is reached at $\nu = 4$, as expected.

After deciding on a target criterion such as the MSE, ideally one would select the model to minimize this criterion. This is, however, not practical because the criterion usually involves some unknown quantities. For example, MSE depends on the unknown true function f which one wants to estimate in the first place. Thus one needs to estimate this criterion from the data and then minimize the estimated criterion. We discuss unbiased and cross-validation estimates of MSE in Sects. 16.3 and 16.4 respectively.

16.3 AIC, BIC, C_p and Their Variations

Let GOF(M_λ) and $|M_\lambda|$ be measures of the GOF and complexity for model M_λ. A direct compromise between these two conflicting quantities is

$$\text{GOF}(M_\lambda) + \xi |M_\lambda| \,, \tag{16.19}$$

where the parameter ξ controls how this compromise should be achieved. Note that the penalized LS (16.10) may be considered as a special case with the LS regarded as a measure of GOF and the squared integral regarded as a measure of complexity.

To be concrete, let us consider the regression model (16.2). For a fixed λ, the estimates are linear, $\widehat{\mathbf{f}}_\lambda = H(\lambda)\mathbf{y}$, where $H(\lambda) = P(\lambda)$ for the trigonometric model and $H(\lambda) = A(\lambda)$ for the periodic spline. Suppose that LS is used as the measure of GOF and $|M_\lambda| = tr(H(\lambda))$. Let us first consider the case when the error variance σ^2 is known. Then the criterion (16.19) can be re-expressed as

$$U(\lambda, \boldsymbol{\theta}) = \frac{1}{n}||\mathbf{y} - \widehat{\mathbf{f}}_\lambda||^2 + \frac{\boldsymbol{\theta}}{n}\sigma^2 trH(\lambda) \,. \tag{16.20}$$

(16.20) is known as the final prediction error (FPE) criterion (Akaike 1970). Many existing model selection methods corresponds to special choices of $\boldsymbol{\theta}$: $\boldsymbol{\theta} = 2$ for Akaike's AIC and Mallow's C_p, and $\boldsymbol{\theta} = \log n$ for Schwarz's BIC. The C_p method is also called the *unbiased risk method* (UBR) in smoothing spline literature (Wahba 1990). The following simple argument provides the motivation behind the C_p (UBR) method.

$$\begin{aligned} \text{E}\left(\frac{1}{n}||\mathbf{y} - \widehat{\mathbf{f}}_\lambda||^2\right) &= \text{E}\left(\frac{1}{n}||\mathbf{y} - \mathbf{f}||^2 + \frac{2}{n}(\mathbf{y} - \mathbf{f})^\top(\mathbf{f} - \widehat{\mathbf{f}}_\lambda) + \frac{1}{n}||\mathbf{f} - \widehat{\mathbf{f}}||^2\right) \\ &= \sigma^2 - \frac{2}{n}\sigma^2 trH(\lambda) + \text{MSE}(\lambda). \end{aligned} \tag{16.21}$$

Therefore, ignoring the constant σ^2, $U(\lambda, 2)$ is an unbiased estimate of MSE(λ).

Other choices of $\boldsymbol{\theta}$ were motivated from different principles: AIC is an estimate of the expected Kullback-Leibler discrepancy where the second term in (16.20)

is considered as a bias correction (Burnham and Anderson 2002) and BIC is an asymptotic Bayes factor (Sect. 16.5). Since each method was derived with different motivations, it is not surprising that they have quite different theoretical properties (Shao 1997). $\boldsymbol{\theta}$ in (16.20) can be considered as a penalty to the model complexity. A larger penalty ($\boldsymbol{\theta}$) leads to a simpler model. As a result, AIC and C_p perform well for "complex" true models and poorly for "simple" true models, while BIC does just the opposite. In practice the nature of the true model, "simple" or "complex", is never known. Thus a data driven choice of model complexity penalty $\boldsymbol{\theta}$ would be desirable. Several methods have been proposed to estimate $\boldsymbol{\theta}$ (Bai et al. 1999; Rao and Wu 1989; Rao 1999; Rao and Tibshirani 1997; Shen and Ye 2002). We now discuss Shen and Ye (2002)'s method based on the generalized degrees of freedom. We will discuss the cross-validation method (Rao and Tibshirani 1997) in the next section.

Now consider both λ and $\boldsymbol{\theta}$ in (16.20) as unknown parameters. Denote $\widehat{\lambda}(\boldsymbol{\theta})$ as the selected model index based on (16.20) for a fixed $\boldsymbol{\theta}$, and $\widehat{f}_{\widehat{\lambda}(\boldsymbol{\theta})}$ as the estimate based on the selected model. The dependence on $\boldsymbol{\theta}$ is made explicit. We now want to find $\boldsymbol{\theta}$ which minimizes the MSE

$$\mathrm{MSE}(\boldsymbol{\theta}) = \mathrm{E}\left(\frac{1}{n}||\widehat{\mathbf{f}}_{\widehat{\lambda}(\boldsymbol{\theta})} - \mathbf{f}||^2\right).$$

Again, we need to estimate $\mathrm{MSE}(\boldsymbol{\theta})$. As (16.21), we have

$$\begin{aligned}\mathrm{E}\left(\frac{1}{n}||\mathbf{y} - \widehat{\mathbf{f}}_{\widehat{\lambda}(\boldsymbol{\theta})}||^2\right) &= \mathrm{E}\left(\frac{1}{n}||\mathbf{y}-\mathbf{f}||^2 + \frac{2}{n}(\mathbf{y}-\mathbf{f})^\top\left(\mathbf{f} - \widehat{\mathbf{f}}_{\widehat{\lambda}(\boldsymbol{\theta})}\right)\right.\\ &\quad \left. + \frac{1}{n}||\mathbf{f} - \widehat{\mathbf{f}}_{\widehat{\lambda}(\boldsymbol{\theta})}||^2\right)\\ &= \sigma^2 - \frac{2}{n}\sigma^2 g_0(\boldsymbol{\theta}) + \mathrm{MSE}(\boldsymbol{\theta}),\end{aligned}$$

where

$$g_0(\boldsymbol{\theta}) = \mathrm{E}\boldsymbol{\epsilon}^\top\left(\widehat{\mathbf{f}}_{\widehat{\lambda}(\boldsymbol{\theta})} - \mathbf{f}\right)/\sigma^2$$

is the *generalized degrees of freedom* (gdf) defined in Ye (1998). Thus, ignoring a constant σ^2,

$$G(\boldsymbol{\theta}) = \frac{1}{n}||\mathbf{y} - \widehat{\mathbf{f}}_{\widehat{\lambda}(\boldsymbol{\theta})}||^2 + \frac{2}{n}\sigma^2 g_0(\boldsymbol{\theta}) \tag{16.22}$$

is an unbiased estimate of $\mathrm{MSE}(\boldsymbol{\theta})$. More rigorous justification can be found in Shen and Ye (2002). Note that $\widehat{\lambda}(\boldsymbol{\theta})$ depends on $\mathbf{y}$. Thus $\widehat{\mathbf{f}}_{\widehat{\lambda}(\boldsymbol{\theta})} = H(\widehat{\lambda}(\boldsymbol{\theta}))\mathbf{y}$ is a non-linear estimator. Usually $g_0(\boldsymbol{\theta}) \neq trH(\widehat{\lambda}(\boldsymbol{\theta}))$. We need to estimate the gdf $g_0(\boldsymbol{\theta})$. It can be shown that (Ye 1998)

$$\widehat{g}_0(\boldsymbol{\theta}) = \int \boldsymbol{\epsilon}^\top \widehat{\mathbf{f}}_{\widehat{\lambda}(\boldsymbol{\theta})}(\mathbf{y} + \boldsymbol{\epsilon})\phi_\tau(\boldsymbol{\epsilon})d\boldsymbol{\epsilon}$$

is an approximately unbiased estimate of $g_0(\boldsymbol{\theta})$, where $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \tau^2 I)$, $\phi_\tau(\boldsymbol{\epsilon})$ is the n-dimensional density of $N(\mathbf{0}, \tau^2 I)$, and $\widehat{\mathbf{f}}_{\widehat{\lambda(\boldsymbol{\theta})}}(\mathbf{y}+\boldsymbol{\epsilon})$ is the fit to the perturbed data $\mathbf{y}+\boldsymbol{\epsilon}$. Ye (1998) suggested the following Monte Carlo approach to compute $\widehat{g}_0(\boldsymbol{\theta})$: for a fixed $\boldsymbol{\theta}$:

1. draw a n-dimensional sample $\boldsymbol{\epsilon} \sim N(\mathbf{0}, \tau^2 I)$, use the perturbed sample $\mathbf{y}+\boldsymbol{\epsilon}$ to select a model based on (16.20) with fixed $\boldsymbol{\theta}$ and calculate the fits $\widehat{\mathbf{f}}_{\widehat{\lambda(\boldsymbol{\theta})}}(\mathbf{y}+\boldsymbol{\epsilon})$.
2. Repeating above step T times, one has $\boldsymbol{\epsilon}_t = (\delta_{t1} \cdots, \delta_{tn})^\top$ and $\widehat{\mathbf{f}}_{\widehat{\lambda(\boldsymbol{\theta})}}(\mathbf{y}+\boldsymbol{\epsilon}_t) \triangleq (\widehat{f}_{t1}, \cdots, \widehat{f}_{tn})^\top, t = 1, \cdots, T$.
3. For a fixed i, $i = 1, \cdots, n$, calculate the regression slope $\widehat{h}_i$ of the following linear model
$$\widehat{f}_{ti} = \mu + \widehat{h}_i \delta_{ti}, \qquad t = 1, \cdots, T.$$
4. Estimate $g_0(\boldsymbol{\theta})$ by $\sum_{i=1}^n \widehat{h}_i$.

$\tau \in [0.5\sigma, \sigma]$ is generally a good choice and the results are usually insensitive to the choice of τ when it is in this region (Ye 1998). A data-driven choice of $\boldsymbol{\theta}$ is the minimum of (16.22) with $g_0(\boldsymbol{\theta})$ replaced by its estimate $\sum_{i=1}^n \widehat{h}_i$ (Shen and Ye 2002).

When σ^2 is unknown, one may replace σ^2 in (16.20) and (16.22) by a consistent estimate. Many estimators were proposed in literature (Dette et al. 1998; Donoho and Johnston 1994; Gasser et al. 1986; Hall et al. 1990; Rice 1984). The Rice's estimator is one of the simplest. For model (16.2), Rice (1984) proposed to estimate σ^2 by
$$\tilde{\sigma}^2 = \frac{1}{2(n-1)} \sum_{i=2}^{n} (y_i - y_{i-1})^2 .$$

In the remaining of this chapter, σ^2 is replaced by $\tilde{\sigma}^2$ whenever necessary.

Another option, assuming the distribution of y_i's is known, is to replace GOF(M_λ) in (16.19) by $-2\log$(maximum likelihood). For the regression models with Gaussian random errors, this leads to
$$n \log(||\mathbf{y} - \widehat{\mathbf{f}}_\lambda||^2) + \boldsymbol{\theta} trH(\lambda) . \tag{16.23}$$

Again, $\boldsymbol{\theta} = 2$ and $\boldsymbol{\theta} = \log n$ correspond to AIC and BIC criteria respectively. The same data-driven procedure discussed above may also be used to select $\boldsymbol{\theta}$.

Derived from asymptotic argument, the AIC method may lead to over-fitting for small samples (Burnham and Anderson 2002; Hurvich and Tsai 1989). The following AIC_c criterion modifies (16.23) with a second order bias adjustment (Hurvich and Tsai 1989)
$$\text{AIC}_c = n \log(||\mathbf{y} - \widehat{\mathbf{f}}_\lambda||^2) + 2trH(\lambda) \frac{n}{n - trH(\lambda) - 1} .$$

AIC_c should be used when the ratio between n and the number of parameters in the largest candidate model is small, say less than 40 (Burnham and Anderson 2002). In our trigonometric model, the highest dimension may reach n. Thus we will use AIC_c in our computations.

Now consider the trigonometric model. It is easy to check that criterion (16.20) reduces to

$$\sum_{\nu=\lambda+1}^{K} \left(\tilde{y}_{2\nu}^2 + \tilde{y}_{2\nu+1}^2\right) + \frac{\boldsymbol{\theta}}{n}\sigma^2(2\lambda+1)\,.$$

Thus adding the λth frequency reduces RSS by $\tilde{y}_{2\lambda}^2 + \tilde{y}_{2\lambda+1}^2$ and increases the complexity part by $2\boldsymbol{\theta}\sigma^2/n$. When $\tilde{y}_{2\lambda}^2 + \tilde{y}_{2\lambda+1}^2$ decreases with increasing λ, one should keeping adding frequencies until $\tilde{y}_{2\lambda}^2 + \tilde{y}_{2\lambda+1}^2 \le 2\boldsymbol{\theta}\sigma^2/n$. It is not difficult to see that the C_p criterion corresponds to applying rule (16.18) with $\alpha_{2\lambda}^2 + \alpha_{2\lambda+1}^2$ replaced by its unbiased estimate $\tilde{y}_{2\lambda}^2 + \tilde{y}_{2\lambda+1}^2 - 2\sigma^2/n$. Other data-based thresholding can be found in Donoho and Johnston (1994), Beran (1996), Zhou and Huang (2005) and Hinkley (2003).

Fitting trigonometric models to the climate data, we plot scores of AIC_c, BIC and C_p criteria as functions of the frequency in the left panel of Fig. 16.6. The AIC_c and C_p criteria reach minimum at $\lambda = 2$ and the BIC criterion reaches the minimum at $\lambda = 1$. For a grid of $\boldsymbol{\theta}$ in the interval $[0, \log n]$, we calculate the optimal λ, $\widehat{\lambda}(\boldsymbol{\theta})$, based on (16.20). We also calculate the estimated gdf using $T = 1{,}000$ and $\tau = 0.75\tilde{\sigma}$. The middle panel of Fig. 16.6 shows the estimated gdf together with the degrees of freedom based on the selected model, $2\widehat{\lambda}(\boldsymbol{\theta}) + 1$. The gdf is intended to account for the extra cost for estimating λ. As expected, the gdf is almost always larger than the degrees of freedom. The gdf is close to the degrees of freedom when $\boldsymbol{\theta}$ is small or large. In the middle, it can have significant corrections to the degrees of freedom. Overall, the gdf smoothes out the corners in the discrete degrees of freedom. The RSS, complexity $2\widehat{g}_0\tilde{\sigma}^2$ and $G(\boldsymbol{\theta})$ are plotted in the right panel of Fig. 16.6. The minimum of $G(\boldsymbol{\theta})$ is reached at $\boldsymbol{\theta} = 3.68$ with $\widehat{\lambda}(3.68) = 2$. Trigonometric model fits with $\lambda = 1$ and $\lambda = 2$ are shown in Fig. 16.2.

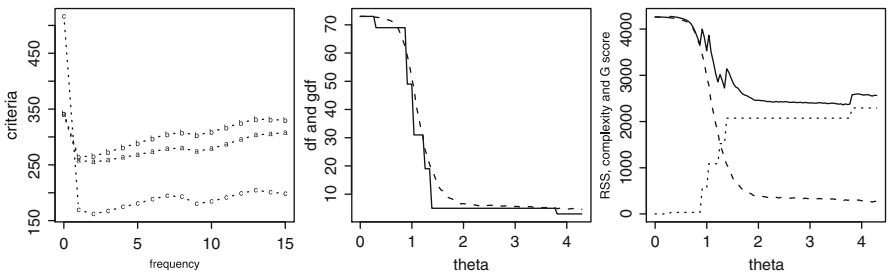


Fig. 16.6 *Left*: Scores of AIC_c, BIC and C_p criteria marked as letters "a", "b" and "c" respectively. *Middle*: degrees of freedom (*solid line*) and estimated gdf (*dashed line*). *Right*: RSS (*dotted line*), model complexity part (*dashed line*) and the G score (*solid line*) in (16.22)

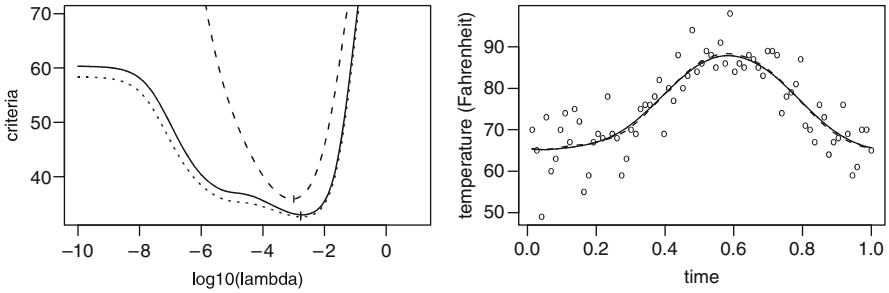


Fig. 16.7 *Left*: Scores of the C_p (UBR), GCV and GML criteria plotted as *dotted*, *solid* and *dashed lines* respectively. Minimum points are marked with vertical bars. *Right*: Observations (*circles*) and fits from the periodic spline with the UBR (*dotted line*), GCV (*solid line*) and GML (*dashed line*) choices of smoothing parameter

Fitting periodic spline models to the climate data, we plot the C_p (UBR) criterion in the left panel of Fig. 16.7. Fits with the UBR choice of the smoothing parameter is shown in the right panel of Fig. 16.7.

16.4 Cross-Validation and Generalized Cross-Validation

The reason one cannot use the GOF for model selection is that it generally under-estimates the generalization error of a model (Efron 1986; Hastie et al. 2002). For example, (16.21) shows that the RSS under-estimates the PSE by $2\sigma^2 trH(\lambda)/n$. Thus, similar to the correction term in the AIC, the second term in the C_p criterion corrects this bias. The bias in RSS is a result of using the same data for model fitting and model evaluation. Ideally, these two tasks should be separated using independent samples. This can be achieved by splitting the whole data into two subsamples, a training (calibration) sample for model fitting and a test (validation) sample for model evaluation. This approach, however, is not efficient unless the sample size is large. The idea behind the cross-validation is to recycle data by switching the roles of training and test samples.

Suppose that one has decided on a measure of discrepancy for model evaluation, for example the prediction error. A V-fold cross-validation selects a model as follows:

1. Split the whole data into V disjoint subsamples $S_1, \cdots, S_V$.
2. For $v = 1, \cdots, V$, fit model M_λ to the training sample $\cup_{i \neq v} S_i$, and compute discrepancy, $d_v(\lambda)$, using the test sample S_v.
3. Find optimal λ as the minimizer of the overall discrepancy $d(\lambda) = \sum_{v=1}^{V} d_v(\lambda)$.

The cross-validation is a general procedure that can be applied to estimate tuning parameters in a wide variety of problems. To be specific, we now consider the regression model (16.2). For notational simplicity, we consider the delete-1 (leave-

one-out) cross-validation with $V = n$. Suppose our objective is prediction. Let $\mathbf{y}^{-i}$ be the $n-1$ vector with the ith observation, y_i, removed from the original response vector $\mathbf{y}$. Let $\widehat{f}_\lambda^{-i}$ be the estimate based on $n-1$ observations $\mathbf{y}^{-i}$. The ordinary cross-validation (OCV) estimate of the prediction error is

$$\mathrm{OCV}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{f}_\lambda^{-i}(t_i))^2 . \tag{16.24}$$

A cross-validation estimate of λ is the minimizer of (16.24). The cross-validation method was introduced by Allen (1974) (also called PRESS) in the context of linear regression and by Wahba and Wold (1975) in the context of smoothing splines. To compute the OCV score, one needs to fit model M_λ n times, once for each delete-one data $\mathbf{y}^{-i}$. This is computationally intensive. Fortunately, a short-cut exists for many situations. Let

$$\tilde{y}_{ij} = \begin{cases} y_j, & j \neq i, \\ \widehat{f}_\lambda^{-i}(t_i), & j = i, \end{cases}$$

and $\tilde{\mathbf{y}}_i = (\tilde{y}_{i1}, \cdots, \tilde{y}_{in})^\top$. $\tilde{\mathbf{y}}_i$ simply replaces the ith element in $\mathbf{y}$, the one deleted to get $\mathbf{y}^{-i}$, by $\widehat{f}_\lambda^{-i}(t_i)$. Let $\tilde{f}_\lambda^{-i}$ be the estimate of f with data $\tilde{\mathbf{y}}_i$. For many methods such as the trigonometric regression (linear regression in general) and periodic spline (smoothing spline in general), we have the following

Lemma 1 (Leaving-Out-One Lemma). $\tilde{f}_\lambda^{-i}(t_i) = \widehat{f}_\lambda^{-i}(t_i),\ i = 1, \cdots, n.$

See Wahba (1990) and Hastie and Tibshirani (1990) for proofs. Note that even though it is called the leaving-out-one lemma, similar results hold for the leaving-out-of-cluster cross-validation (Wang et al. 2000). See also Xiang and Wahba (1996), Zhang et al. (2002) and Ke and Wang (2002) for the leaving-out-one lemma for more complicated problems.

For trigonometric regression and periodic spline models, $\widehat{\mathbf{f}}_\lambda = H(\lambda)\mathbf{y}$ for any $\mathbf{y}$. Thus when $\mathbf{y}$ is replaced by $\tilde{\mathbf{y}}_i$, we have $(\tilde{f}_\lambda^{-i}(t_1), \cdots, \tilde{f}_\lambda^{-i}(t_n))^\top = H(\lambda)\tilde{\mathbf{y}}_i$. Denote the elements of $H(\lambda)$ as $h_{ij}, i, j = 1, \cdots, n$. Then

$$\widehat{f}_\lambda(t_i) = \sum_{j=1}^{n} h_{ij} y_j ,$$

$$\widehat{f}_\lambda^{-i}(t_i) = \tilde{f}_\lambda^{-i}(t_i) = \sum_{j=1}^{n} h_{ij}\tilde{y}_j = \sum_{j\neq i} h_{ij} y_j + h_{ii}\widehat{f}_\lambda^{-i}(t_i) .$$

Combined, we have

$$\widehat{f}_\lambda(t_i) - \widehat{f}_\lambda^{-i}(t_i) = h_{ii}(y_i - \widehat{f}_\lambda^{-i}(t_i)) .$$

Then it is easy to check that

$$y_i - \widehat{f}_\lambda^{-i}(t_i) = (y_i - \widehat{f}_\lambda(t_i))/(1 - h_{ii}) ,$$

and the OCV reduces to

$$\mathrm{OCV}(\lambda) = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{y_i - \widehat{f}_\lambda(t_i)}{1 - h_{ii}}\right)^2 . \tag{16.25}$$

Thus one only needs to fit the model once with the full data and compute the diagonal elements of the $H(\lambda)$ matrix.

Replacing h_{ii} in (16.25) by the average of all diagonal elements, Craven and Wahba (1979) proposed the following *generalized cross-validation* (GCV) criterion

$$\mathrm{GCV}(\lambda) = \frac{\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{f}_\lambda(t_i))^2}{(1 - trH(\lambda)/n)^2} . \tag{16.26}$$

It is easy to see that the GCV criterion is a weighted version of OCV with weights $(1 - h_{ii})^2/(1 - trH(\lambda)/n)^2$. When $trH(\lambda)/n$ is small, using the approximation $(1 - x)^2 \approx 1 + 2x$,

$$\mathrm{GCV}(\lambda) \approx \frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{f}_\lambda(t_i))^2 + \frac{2}{n}trH(\lambda)\left[\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{f}_\lambda(t_i))^2\right] .$$

Regarding $\frac{1}{n}\sum_{i=1}^{n}(y_i - \widehat{f}_\lambda(t_i))^2$ in the second part as an estimate of σ^2, the GCV is approximately the same as the C_p (UBR) criterion. Originally proposed to reduce the computational burden, the GCV criterion has been found to possess several favorable properties (Golub et al. 1979; Li 1985, 1986; 1987, Gu 2002; Wahba 1990). Sometimes it is difficult to compute each diagonal element in $H(\lambda)$ directly. Nevertheless, it is relatively easy to approximate $trH(\lambda)$ using the randomized trace method (Zhang et al. 2002). Thus the GCV criterion may be adopted to many complicated problems (Xiang and Wahba 1996; Zhang et al. 2002). The GCV criterion has non-zero probability to select $\lambda = 0$ (interpolation) which may cause problems when the sample size is small. Fortunately, this probability tends to zero exponentially fast as sample size increases (Wahba and Wang 1993).

For the trigonometric regression,

$$h_{ii} = \frac{1}{n}\left(1 + \sum_{\nu=1}^{\lambda} 2(\sin^2 2\pi\nu t_i + \cos^2 2\pi\nu t_i)\right) = \frac{1}{n}(1 + 2\lambda) = \frac{trH(\lambda)}{n} .$$

For the periodic spline,

$$h_{ii} = \frac{1}{n} + \frac{1}{n}\sum_{\nu=1}^{K} 2(\sin^2 2\pi\nu t_i + \cos^2 2\pi\nu t_i)/(1+\lambda(2\pi\nu)^4) = \frac{trD}{n} = \frac{trH(\lambda)}{n}\,.$$

Thus the OCV and GCV are the same for both cases.

Instead of deleting one observation at a time, one may delete d observations at a time as described in the V-fold cross-validation. We will call such a method as delete-d CV. Shao (1997) classified various model selection criteria into the following three classes:

Class 1: AIC, C_p, delete-1 CV and GCV.
Class 2: Criterion (16.20) with $\boldsymbol{\theta} \to \infty$ as $n \to \infty$, and delete-d CV with $d/n \to 1$.
Class 3: Criterion (16.20) with a fixed $\boldsymbol{\theta} > 2$, and delete-d CV with $d/n \to \tau \in (0,1)$.

BIC is a special case of the Class 2. Shao (1997) showed that the criteria in Class 1 are asymptotically valid if there is no fixed-dimensional correct model and the criteria in Class 2 are asymptotically valid when the opposite is true. Methods in Class 3 are compromises of those in Classes 1 and 2. Roughly speaking, criteria in the first class would perform better if the true model is "complex" and the criteria in the second class would do better if the true model is "simple". See also Zhang (1993) and Shao (1993).

The climate data subset was selected by first dividing 365 days in the year 1990 into 73 five-day periods, and then selecting measurements on the third day in each period as observations. This is our training sample. Measurements excluding these selected 73 days may be used as the test sample. This test sample consists $365-73 = 292$ observations. For the trigonometric model with fixed frequency λ, we calculate the prediction error using the test sample

$$\mathrm{PE}(\lambda) = \frac{1}{292}\sum_{i=1}^{292}(y_i - \widehat{f}_\lambda(s_i))^2\,, \tag{16.27}$$

where s_i are time points for observations in the test sample. The prediction errors are plotted in the left panel of Fig. 16.8 where the minimum is reached at $\lambda = 1$. The GCV scores for the trigonometric model is also plotted in the left panel of Fig. 16.8 where the minimum is reached at $\lambda = 2$. The GCV score for the periodic spline and the corresponding fits are plotted in the left and right panels of Fig. 16.7 respectively. As expected, the GCV scores are similar to the UBR scores.

As a general methodology, the cross-validation may also be used to select $\boldsymbol{\theta}$ in (16.20) (Rao and Tibshirani 1997). Let $\widehat{f}^{-i}_{\widehat{\lambda}^{-i}(\boldsymbol{\theta})}$ be the estimate based on the delete-one data $\mathbf{y}^{-i}$ where $\widehat{\lambda}^{-i}(\boldsymbol{\theta})$ is selected using (16.20), also based on $\mathbf{y}^{-i}$. Then the OCV estimate of prediction error is

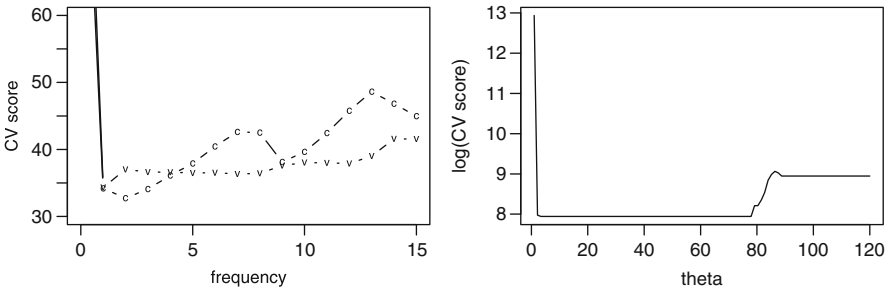


Fig. 16.8 *Left*: prediction errors (16.27) (marked as "v") and GCV scores (marked as "c") for the trigonometric model. *Right*: the OCV scores in (16.28) on the logarithm scale

$$\mathrm{OCV}(\boldsymbol{\theta}) = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \widehat{f}^{-i}_{\widehat{\lambda}^{-i}(\boldsymbol{\theta})}(t_i)\right)^2 . \tag{16.28}$$

The minimum of (16.28) provides an estimate of $\boldsymbol{\theta}$. The OCV score for the trigonometric model is plotted as a function of $\boldsymbol{\theta}$ in the right panel of Fig. 16.8. The minimum is reached at a wide range of $\boldsymbol{\theta}$ values with $\lambda = 1$ or $\lambda = 2$.

16.5 Bayes Factor

Let $P(M_\lambda)$ be the prior probability for model M_λ. For any two models M_{λ_1} and M_{λ_2}, the Bayes factor

$$B(\lambda_1, \lambda_2) = \frac{P(M_{\lambda_1}|\mathbf{y})}{P(M_{\lambda_2}|\mathbf{y})} \div \frac{P(M_{\lambda_1})}{P(M_{\lambda_2})} \tag{16.29}$$

is the posterior odds in favor of model M_{λ_1} divided by the prior odds in favor of model M_{λ_1} (Kass and Raftery 1995). The Bayes factor provides a scale of evidence in favor of one model versus another. For example, $B(\lambda_1, \lambda_2) = 2$ indicates that the data favor model M_{λ_1} over model M_{λ_2} at odds of two to one. Table 16.1 lists a possible interpretation for Bayes factor suggested by Jeffreys (1961).

The Bayes factor is easy to understand and applicable to a wide range of problems. Methods based on the Bayes factor behave like an Occam's razor (Jeffreys and Berger 1992). Non-Bayesian analysis typically selects a model and then proceeds as if the data is generated by the chosen model. Ignoring the fact that the model has been selected from the same data, this approach often leads to underestimation of the uncertainty in quantities of interest, a problem know as the *model selection bias* (Chatfield 1995). Specifically, the estimates of parameters based on the selected model are biased and their variances are usually too optimistic. The Bayesian approach accounts for model uncertainty with the posterior probability

Table 16.1 Jeffreys' scale of evidence for Bayes factors

Bayes factor	Interpretation
$B(\lambda_1, \lambda_2) < 1/10$	Strong evidence for M_{λ_2}
$1/10 < B(\lambda_1, \lambda_2) < 1/3$	Moderate evidence for M_{λ_2}
$1/3 < B(\lambda_1, \lambda_2) < 1$	Weak evidence for M_{λ_2}
$1 < B(\lambda_1, \lambda_2) < 3$	Weak evidence for M_{λ_1}
$3 < B(\lambda_1, \lambda_2) < 10$	Moderate evidence for M_{λ_1}
$B(\lambda_1, \lambda_2) > 10$	Strong evidence for M_{λ_1}

$P(M_\lambda|\mathbf{y})$. For example, to predict a new observation y^+, the best prediction under squared loss is

$$\mathrm{E}(y^+|\mathbf{y}) = \sum_{\lambda \in \Lambda} \mathrm{E}(y^+|M_\lambda, \mathbf{y}) P(M_\lambda|\mathbf{y}) ,$$

a weighted average of predictions from all models with weights equal to the posterior probabilities. Instead of using a single model, such model averaging incorporates model uncertainty. It also indicates that selecting a single model may not be desirable or necessary for some applications such as prediction (Hoeting et al. 1999).

The practical implementation of Bayesian model selection is, however, far from straightforward. In order to compute the Bayes factor (16.29), ones needs to specify priors $P(M_\lambda)$ as well as priors for parameters in each model. While providing a way to incorporating other information into the model and model selection, these priors may be hard to set in practice, and standard non-informative priors for parameters cannot be used (Berger and Pericchi 1996; Gelman et al. 1995). See Kass and Raftery (1995), Chipman et al. (2001) and Berger and Pericchi (2001) for more discussions on the choice of priors.

After deciding on priors, one needs to compute (16.29) which can be re-expressed as

$$B(\lambda_1, \lambda_2) = \frac{P(\mathbf{y}|M_{\lambda_1})}{P(\mathbf{y}|M_{\lambda_2})} , \tag{16.30}$$

where $P(\mathbf{y}|M_\lambda)$ is the marginal likelihood. The marginal likelihood usually involves an integral which can be evaluated analytically only for some special cases. When the marginal likelihood does not have a closed form, several methods for approximation are available including Laplace approximation, importance sampling, Gaussian quadrature and Markov chain Monte Carlo (MCMC) simulations. Details about these methods are out of the scope of this chapter. References can be found in Kass and Raftery (1995).

Under certain conditions, Kass and Wasserman (1995) showed that

$$-2 \log P(\mathbf{y}|M_\lambda) \approx -2 \log(\text{maximum likelihood}) + |M_\lambda| \log n .$$

Thus the BIC is an approximation to the Bayes factor.

In the following we discuss selection of the smoothing parameter λ for the periodic spline. Based on (16.30), our goal is to find λ which maximizes the

marginal likelihood $P(\mathbf{y}|M_\lambda)$, or equivalently, $P(\tilde{\mathbf{y}}|M_\lambda)$ where $\tilde{\mathbf{y}}$ is the discrete Fourier transformation of $\mathbf{y}$. Note that

$$\tilde{\mathbf{y}} = \boldsymbol{\alpha} + \tilde{\boldsymbol{\varepsilon}}, \tag{16.31}$$

where $\tilde{\boldsymbol{\varepsilon}} = X_K^\top \boldsymbol{\varepsilon}/n \sim N(\mathbf{0}, \sigma^2 I/n)$. Let $b = \sigma^2/n\lambda$. Assume the following prior for $\boldsymbol{\alpha}$:

$$\begin{aligned} \alpha_1 &\propto 1, \\ \alpha_{2\nu} &\sim N(0, b(2\pi\nu)^{-4}), \qquad \nu = 1, \cdots, K, \\ \alpha_{2\nu+1} &\sim N(0, b(2\pi\nu)^{-4}), \qquad \nu = 1, \cdots, K, \end{aligned} \tag{16.32}$$

where α_i are mutually independent and are independent of $\tilde{\boldsymbol{\varepsilon}}$. An improper prior is assumed for α_1. It is not difficult to check that $\mathrm{E}(\boldsymbol{\alpha}|\tilde{\mathbf{y}}) = \widehat{\boldsymbol{\alpha}}$. Thus the posterior means of the Bayes model (16.31) and (16.32) are the same as the periodic spline estimates.

Let $\mathbf{z} = (\tilde{y}_2, \cdots, \tilde{y}_n)^\top$ and write $P(\tilde{\mathbf{y}}|M_\lambda) = P(\tilde{y}_1|M_\lambda)P(\mathbf{z}|M_\lambda)$. Since $P(\tilde{y}_1|M_\lambda)$ is independent of λ, we will estimate λ using the marginal likelihood $P(\mathbf{z}|M_\lambda)$. Since $\tilde{y}_{2\nu}$ or $\tilde{y}_{2\nu+1} \sim N(0, b((2\pi\nu)^{-4} + \lambda))$, the log marginal likelihood of $\mathbf{z}$ is

$$\begin{aligned} l(b, \lambda) = &-\frac{n-1}{2}\log 2\pi - \frac{n-1}{2}\log b \\ &- \sum_{\nu=1}^{K} \log[(2\pi\nu)^{-4} + \lambda] - \frac{1}{2b}\sum_{\nu=1}^{K} \frac{\tilde{y}_{2\nu}^2 + \tilde{y}_{2\nu+1}^2}{(2\pi\nu)^{-4} + \lambda}. \end{aligned}$$

Fixing λ and maximizing with respect to b, we have

$$\widehat{b} = \frac{1}{n-1}\sum_{\nu=1}^{K} \frac{\tilde{y}_{2\nu}^2 + \tilde{y}_{2\nu+1}^2}{(2\pi\nu)^{-4} + \lambda}.$$

Plugging back, we have

$$l(\widehat{b}, \lambda) = \text{constant} - \frac{n-1}{2}\log \sum_{\nu=1}^{K} \frac{\tilde{y}_{2\nu}^2 + \tilde{y}_{2\nu+1}^2}{(2\pi\nu)^{-4} + \lambda} - \sum_{\nu=1}^{K} \log\left[(2\pi\nu)^{-4} + \lambda\right].$$

Thus maximizing the log likelihood is equivalent to minimizing

$$M(\lambda) = \frac{\sum_{\nu=1}^{K}(\tilde{y}_{2\nu}^2 + \tilde{y}_{2\nu+1}^2)/((2\pi\nu)^{-4} + \lambda)}{\left(\prod_{\nu=1}^{K}((2\pi\nu)^{-4} + \lambda)\right)^{2/(n-1)}},$$

It is not difficult to check that

$$M(\lambda) = \frac{\mathbf{y}^\top (I - A(\lambda))\mathbf{y}}{\left(\det{}^+(I - A(\lambda))\right)^{1/(n-1)}}, \tag{16.33}$$

where $\det^+$ is the product of non-zero eigenvalues. The criterion (16.33) is called the *generalized maximum likelihood* method in smoothing spline literature (Wahba 1990). It is the same as the *restricted maximum likelihood* (REML) method in the mixed effects literature (Wang 1998). Note that the marginal likelihood is approximated by plugging-in $\widehat{b}$ rather than averaging over a prior distribution for b.

For the climate data, the GML scores for the periodic spline and the corresponding fits are plotted in the left and right panels of Fig. 16.7 respectively. The fits with three different choices of the smoothing parameter are very similar.

16.6 Impact of Heteroscedasticity and Correlation

In our climate example we used one fifth of all measurements in the year 1990. Figure 16.9 shows all measurements in 1990 and periodic spline fits using all measurements with GCV, GML and UBR choices of the smoothing parameter. Obviously the GCV and UBR criteria under-estimate the smoothing parameter which leads to wiggly fits. What is causing the GCV and UBR methods to breakdown?

In model (16.2) we have assumed that random errors are iid with mean zero and variance σ^2. The middle panel of Fig. 16.1 indicates that variation of the maximum temperature is larger during the winter. Also, temperatures close in time may be correlated. Thus the assumption of homoscedasticity and independence may not hold. What kind of impact, if any, do these potential violations have on the model selection procedures?

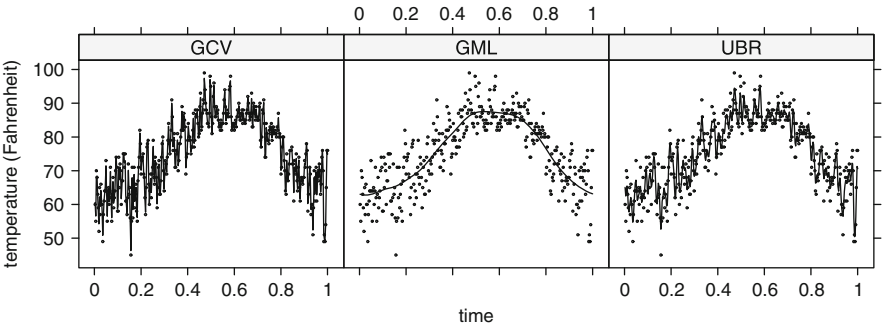


Fig. 16.9 *Points* are all measurements in 1990. *Lines* are the periodic spline estimates. The methods for selecting the smoothing parameter are indicated in strips

For illustration, we again consider two simulations with heteroscedastic and auto-correlated random errors respectively. We use the same function and design points as the simulation in Sect. 16.2 with the true function shown in the left panel of Fig. 16.4. For heteroscedasticity, we generate random errors $\epsilon_i \sim N(0, ((i + 36.5)/147)^2)$, $i = 1, \cdots, 73$, where the variance increases with i. For correlation, we generate the ϵ_i's as a first-order autoregressive process with mean zero, standard deviation 0.5 and first-order correlation 0.5. The first and the second rows in Fig. 16.10 show the fits by the trigonometric model with cross-validation, BIC and C_p choices of orders under heteroscedastic and auto-correlated random errors

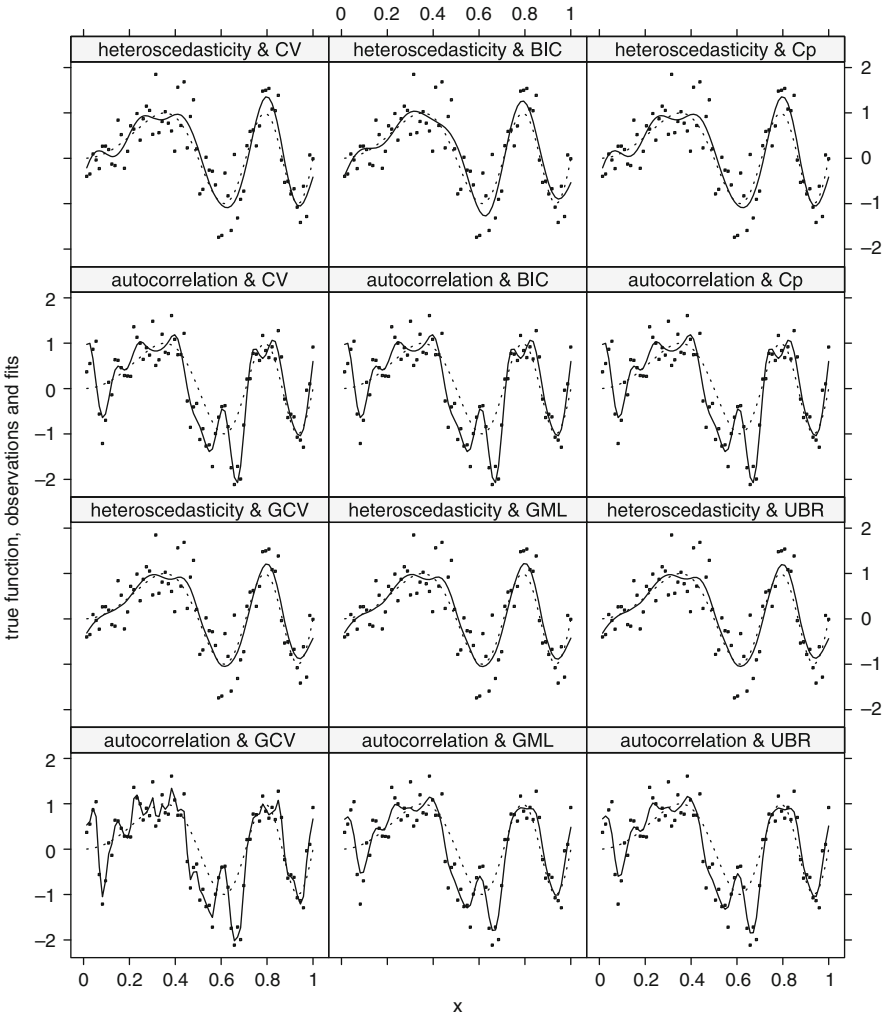


Fig. 16.10 *Circles* are observations. *Dotted lines* are true functions. *Solid lines* are the estimated functions. Simulation settings and model selection criteria are marked in strips

respectively but without adjustment for the heteroscedasticity or correlation. The third and the fourth rows in Fig. 16.10 show the fits by the periodic spline with GCV, GML and UBR choices of smoothing parameters under heteroscedastic and auto-correlated random errors respectively but without adjustment for the heteroscedasticity or correlation. These kind of fits are typical under two simulation settings. The heteroscedasticity has some effects on the model selection, but far less severe than the impact of auto-correlation. It is well-known that positive auto-correlation leads to under-smoothing for non-parametric models with data-driven choices of the smoothing parameter (Opsomer et al. 2001; Wang 1998). Figure 16.10 shows that the same problem exists for parametric regression models as well.

The breakdown of the GCV and UBR criteria for the climate data is likely caused by the auto-correlation which is higher when daily measurements are used as observations. Extensions of the GCV, GML and UBR criteria for correlated data can be found in Wang (1998).

16.7 Discussion

There are many fundamental and philosophical issues in model selection. For example, a model is usually a simplification or approximation of the complicated reality. "All models are wrong, but some are useful" (Box 1976). A selected model tell us what the finite data are likely to support, not the full reality.

Data analysts are constantly making model selections (assumptions), consciously or unconsciously. For example, certain choices have to be made for selecting the candidate models $\mathcal{M}$ (Burnham and Anderson 2002). The selection methods have been formalized in the current literature represent only a fraction of the whole selection process in practice. As a consequence, model selection is considered as both science and art. Scientific knowledge, empirical evidence, common sense, and good judgment all play an important role in this process. It is rarely the case that sufficient information is available to fully specify the model. Thus creative, critical and careful thinking is required. The problem is often so complicated that one should not expect to achieve the final model in one attempt, regardless of which model selection method has been used. Instead, an iterative scheme including diagnostics suggested by Box and Jenkins (1976) should be used.

Some methods such as cross-validation can be applied to a wide variety of applications, while others are designed for specific applications. Different methods have been motivated and justified with different target criteria under different assumptions. Thus it is unrealistic to expect one method to serve all purposes and perform uniformly better under all circumstances.

We used prediction criteria including MSE and PSE as examples. This, of course, does not mean model selection is involved in (or for) prediction only. For example, another important objective of a model is data description. Identifying risk factors for diabetes is as important as predicting a person's chance of having this disease.

Wang and Ke (2002) have developed a user-friendly S package, ASSIST, which includes several functions for fitting various spline based models. The `ssr` function in this package is used to fit periodic spline models in this chapter. The ASSIST package can be downloaded from http://www.pstat.ucsb.edu/ faculty/yuedong/software. More details and examples can be found in the manual of the ASSIST package which also is available at this web-site.

Acknowledgements This work was supported by NIH Grants R01 GM58533.

References

Akaike, H.: Statistical predictor identification. Ann. Inst. Statist. Math. **21**, 203–217 (1970)
Akaike, H.: Information theory and the maximum likelihood principle. International Symposium on Information Theory, In: Petrov, V., Csáki, F. (eds.) pp. 267–281 Budapest: Akademiai Kiádo (1973)
Allen, D.M.: The relationship between variable selection and data augmentation and a method for prediction. Technometrics **16**, 125–127 (1974)
Bai, Z.D., Rao, C.R., Wu, Y.: Model selection with data-oriented penalty. J. Stat. Plann. Infer. **77**, 103–117 (1999)
Beran, R.: Bootstrap variable selection and confidence sets. In: Rieder, H. (eds.) Robust Statistics, Data Analysis and Computer Intensive Methods, Springer Lecture Notes in Statistics 109 (1996)
Berger, J.O., Pericchi, L.R.: The intrinsic bayes factor for model selection and prediction. J. Am. Stat. Assoc. **91**, 109–122 (1996)
Berger, J.O., Pericchi, L.R.: Objective bayesian methods for model selection: Introduction and comparison. In: Lahiri, P. (eds.) Model Selection, Institute of Mathematical Statistics Lecture Notes – Monograph Series, vol. 38, pp. 135–207. Beachwood Ohio (2001)
Box, G. E.P.: Science and statistics. J. Am. Stat. Assoc. **71**, 791–799 (1976)
Box, G.E.P., Jenkins, G.M.: Time Series Analysis, Holden-Day (1976)
Burnham, K.P., Anderson, D.R.: Model Selection and Multimodel Inference, (2nd ed.), Springer, New York (2002)
Chatfield, C.: Model uncertainty, data mining and statistical inference (with discussion). J. Roy. Stat. Soc. B **158**, 419–466 (1995)
Chipman, H., George, E.I., McCulloch, R.E.: The practical implementation of bayesian model selection. In: Lahiri, P. (eds.) Model Selection, Institute of Mathematical Statistics Lecture Notes – Monograph Series, vol. 38, pp. 65–134. Beachwood Ohio (2001)
Craven, P., Wahba, G.: Smoothing noisy data with spline functions. Numer. Math. **31**, 377–403 (1979)
Dette, H., Munk, A., Wagner, T.: Estimating the variance in nonparametric regression – what is a reasonable choice? J. Roy. Stat. Soc. B **60**, 751–764 (1998)
Donoho, D.L., Johnston, I.M.: Ideal spatial adaption by wavelet shrinkage. Biometrika **81**, 425–456 (1994)
Efron, B.: How biased is the apparent error rate of a prediction rule. J. Am. Stat. Assoc. **81**, 461–470 (1986)
Gasser, T., Sroka, L., Jennen-Steinmetz, C.: Residual variance and residual pattern in nonlinear regression. Biometrika **73**, 625–633 (1986)
Gelman, A., Carlin, J.B., Stern, H.S., Rubin, D.B.: Bayesian Data Analysis, Chapman & Hall, Boca Raton (1995)
George, E.I.: The variable selection problem. J. Am. Stat. Assoc. **95**, 1304–1308 (2000)

Golub, G., Heath, M., Wahba, G.: Generalized cross validation as a method for choosing a good ridge parameter. Technometrics **21**, 215–224 (1979)
Gu, C.: Model indexing and smoothing parameter selection in nonparametric function estimation (with discussion). Statistica Sinica **8**, 632–638 (1998)
Gu, C.: Smoothing Spline ANOVA Models, Springer, New York (2002)
Hall, P., Kay, J.W., Titterington, D.M.: Asymptotically optimal difference-based estimation of variance in nonparametric regression. Biometrika **77**, 521–528 (1990)
Hastie, T., Tibshirani, R.: Generalized Additive Models. Chapman and Hall, London (1990)
Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning, Springer, New York (2002)
Hinkley, D.: Bootstrap methods for variable selection and shrinkage estimatior confidence sets, Personal Communication (2003)
Hoeting, J.A., Madigan, D., Raftery, A.E., Volinsky, C.T.: Bayesian model averaging: a tutorial (with discussion). Stat. Sci. **14**, 382–417 (1999); Corrected version available at http://www.stat.washington.edu/www/research/online/hoeting1999.pdf.
Hurvich, C.M., Tsai, C.L.: Regression and time series model selection in small samples. Biometrika **76**, 297–207 (1989)
Jeffreys, H.: Theorey of Probability, Clarendon Press, Oxford (1961)
Jeffreys, W., Berger, J.O.: Ockham's razor and bayesian analysis. Am. Sci. **80**, 64–72 (1992)
Kass, R.E., Raftery, A.: Bayesian factors. J. Am. Stat. Assoc. **90**, 773–795 (1995)
Kass, R.E., Wasserman, L.: A reference bayesian test for nested hypotheses and its relationship to the schwarz criterion. J. Am. Stat. Assoc. **90**, 982–934 (1995)
Ke, C., Wang, Y.: Nonparametric nonlinear regression models, Technical Report # 385, Department of Statistics and Applied Probability, University of California, Santa Barbara (2002)
Li, K.C.: From Stein's unbaised risk estimates to the method of generalized cross-validation. Ann. Stat. **13**, 1352–1377 (1985)
Li, K.C.: Asymptotic optimality of C_L and generalized cross-validation in ridge regression with application to spline smoothing. Ann. Stat. **14**, 1101–1112 (1986)
Li, K.C.: Asymptotic optimality of C_p, C_L, cross-validation and generalized cross-validation: Discrete index set. Ann. Stat. **15**, 958–975 (1987)
Linhart, H., Zucchini, W.: Model Selection, Wiley, New York (1986)
Mallows, C.L.: Some comments on C_p. Technometrics **12**, 661–675 (1973)
Miller, A.: Subset Selection in Regression, (2nd edn.), Chapman & Hall, New York (2002)
Opsomer, J., Wang, Y., Yang, Y.: Nonparametric regression with correlated errors. Stat. Sci. **16**, 134–153 (2001)
Rao, C.R., Wu, Y.: A strongly consistent procedure for model selection in a regreesion problem. Biometrika **76**, 369–374 (1989)
Rao, J.S.: Bootstrap choice of cost complexity for better subset selection. Statistica Sinica **9**, 273–287 (1999)
Rao, J.S., Tibshirani, R.: Discussion to "an asympototic theory for model selection" by Jun Shao. Statistica Sinica **7**, 249–252 (1997)
Rice, J.A.: Bandwidth choice for nonparametric regression. Ann. Stat. **12**, 1215–1230 (1984)
Schwarz, G.: Estimating the dimension of a model. Ann. Stat. **12**, 1215–1231 (1978)
Shao, J.: Linear model selection by cross-validation. J. Am. Stat. Assoc. **88**, 486–494 (1993)
Shao, J.: An asymptotic theory for linear model selection (with discussion). Statistica Sinica **7**, 221–264 (1997)
Shen, X., Ye, J.: Adaptive model selection. J. Am. Stat. Assoc. **97**, 210–221 (2002)
Stone, M.: Cross-validatory choice and assessment of statistical prediction. J. Roy. Stat. Soc. B **36**, 111–147 (1974)
Wahba, G.: Spline Models for Observational Data, CBMS-NSF Regional Conference Series in Applied Mathematics, vol. 59, SIAM, Philadelphia (1990)
Wahba, G., Wang, Y.: Behavior near zero of the distribution of GCV smoothing parameter estimates for splines. Stat. Probab. Lett. **25**, 105–111 (1993)
Wahba, G., Wold, S.: A completely automatic french curve. Comm. Stat. **4**, 1–17 (1975)

Wang, Y.: Smoothing spline models with correlated random errors. J. Am. Stat. Assoc. **93**, 341–348 (1998)
Wang, Y: Smoothing Splines: Methods and Applications, Chapman and Hall, London (2011)
Wang, Y., Ke, C.: ASSIST: A suite of s-plus functions implementing spline smoothing techniques, Proceedings of the Hawaii International Conference on Statistics (2002); Available at http://www.pstat.ucsb.edu/faculty/yuedong/software
Wang, Y., Guo, W., Brown, M.B.: Spline smoothing for bivariate data with applications to association between hormones. Statistica Sinica **10**, 377–397 (2000)
Xiang, D., Wahba, G.: A genralized approximate cross validation for smoothing splines with non-gaussian data. Statistica Sinica **6**, 675–692 (1996)
Yang, Y.: Model selection for nonparametric regression. Statistica Sinica **9**, 475–499 (1999)
Ye, J.: On measuring and correcting the effects of data mining and model selection. J. Am. Stat. Assoc. **93**, 120–131 (1998)
Zhang, H., Wahba, G., Lin, Y., Voelker, M., Ferris, M., Klein, R., Klein, B.: Variable selection and model building via likelihood basis pursuit, Technical Report No. 1059, Department of Statistics, University of Wisconsin (2002)
Zhang, P.: Model selection via multifold cross validation. Ann. Stat. **21**, 299–313 (1993)
Zhou, H., Huang, J.T.: Minimax estimation with thresholding and its application to wavelet analysis. Ann. Stat. **33**, 101–125 (2005)

Chapter 17
Bootstrap and Resampling

Enno Mammen and Swagata Nandi

17.1 Introduction

The bootstrap is by now a standard method in modern statistics. Its roots go back to a lot of resampling ideas that were around in the seventies. The seminal work of Efron (1979) synthesized some of the earlier resampling ideas and established a new framework for simulation based statistical analysis. The idea of the bootstrap is to develop a setup to generate more (pseudo) data using the information of the original data. True underlying sample properties are reproduced as closely as possible and unknown model characteristics are replaced by sample estimates.

In its basic nature the bootstrap is a data analytic tool. It allows to study the performance of statistical methods by applying them repeatedly to bootstrap pseudo data ("resamples"). The inspection of the outcomes for the different bootstrap resamples allows the statistician to get a good feeling on the performance of the statistical procedure. In particular, this concerns graphical methods. The random nature of a statistical plot is very difficult to be summarized by quantitative approaches. In this respect data analytic methods differ from classical estimation and testing problems. We will illustrate data analytic uses of the bootstrap in the next section.

Most of the bootstrap literature is concerned with bootstrap implementations of tests and confidence intervals and bootstrap applications for estimation problems. It has been argued that for these problems bootstrap can be better understood if it is described as a plug-in method. Plug-in method is an approach used for the

E. Mammen (✉)
University of Mannheim, Mannheim, Germany
e-mail: emammen@rumms.uni-mannheim.de

S. Nandi
Indian Statistical Institute, Delhi Centre, New Delhi, India
e-mail: nandi@isid.ac.in

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_17,

estimation of functionals that depend on unknown finite or infinite dimensional model parameters of the observed data set. The basic idea of plug-in estimates is to estimate these unknown parameters and to plug them into the functional. A wide known example is the plug-in bandwidth selector for kernel estimates. Asymptotical optimal bandwidths typically depend e.g. on averages of derivatives of unknown curves (e.g. densities, regression functions), residual variances, etc. Plug-in bandwidth selectors are constructed by replacing these unknown quantities by finite sample estimates. We now illustrate why the bootstrap can be understood as a plug-in approach. We will do this for i.i.d. resampling. This is perhaps the most simple version of the bootstrap. It is applied to an i.i.d. sample $X_1, \ldots, X_n$ with underlying distribution P. I.i.d. resamples are generated by drawing n times with replacement from the original sample $X_1, \ldots, X_n$. This gives a resample $X_1^*, \ldots, X_n^*$. More formally, the resample is constructed by generating $X_1^*, \ldots, X_n^*$ that are conditionally independent (given the original data set) and have conditional distribution $\hat{P}_n$. Here $\hat{P}_n$ denotes the empirical distribution. This is the distribution that puts mass $1/n$ on each value of $X_1, \ldots, X_n$ in case that all observations have different values (or more generally, mass j/n on points that appear j times in the sample), i.e. for a set A we have $\hat{P}_n(A) = n^{-1} \sum_{i=1}^n \mathrm{I}(X_i \in A)$ where I denotes the indicator function. The bootstrap estimate of a functional $T(P)$ is defined as the plug-in estimate $T(\hat{P}_n)$. Let us consider the mean $\mu(P) = \int xP(\mathrm{d}x)$ as a simple example. The bootstrap estimate of $\mu(P)$ is given by $\mu(\hat{P}_n)$. Clearly, the bootstrap estimate is equal to the sample mean $\overline{X}_n = n^{-1} \sum_{i=1}^n X_i$. In this simple case, simulations are not needed to calculate the bootstrap estimate. Also in more complicated cases it is very helpful to distinguish between the statistical performance and the algorithmic calculation of the bootstrap estimate. In some cases it may be more appropriate to calculate the bootstrap estimate by Monte-Carlo simulations, in other cases powerful analytic approaches may be available. The discussion which algorithmic approach is preferable should not be mixed up with the discussion of the statistical properties of the bootstrap estimate. Perhaps, clarification of this point is one of the major advantages of viewing the bootstrap as a plug-in method. Let us consider now a slightly more complicated example. Suppose that the distribution of $\sqrt{n}[\overline{X}_n - \mu(P)]$ is our functional $T_n(P) = T(P)$ that we want to estimate. The functional now depends on the sample size n. The factor $\sqrt{n}$ has been introduced to simplify asymptotic considerations following below. The bootstrap estimate of $T_n(P)$ is equal to $T_n(\hat{P}_n)$. This is the conditional distribution of $\sqrt{n}[\overline{X}_n^* - \mu(\hat{P}_n)] = \sqrt{n}(\overline{X}_n^* - \overline{X}_n)$, given the original sample $X_1, \ldots, X_n$. In this case the bootstrap estimate could be calculated by Monte-Carlo simulations. Resamples are generated repeatedly, say m-times, and for the j-th resample the bootstrap statistic $\Delta_j = \sqrt{n}(\overline{X}_n^* - \overline{X}_n)$ is calculated. This gives m values $\Delta_1, \ldots, \Delta_m$. Now the bootstrap estimate $T_n(\hat{P}_n)$ is approximated by the empirical distribution of these m values. E.g. the quantiles of the distribution $T_n(P)$ of $\sqrt{n}[\overline{X}_n - \mu(P)]$ are estimated by the sample quantiles of $\Delta_1, \ldots, \Delta_m$. The bootstrap quantiles can be used to construct "bootstrap confidence intervals" for $\mu(P)$. We will come back to bootstrap confidence intervals in Sect. 17.3.

There are two other advantages of the plug-in view of the bootstrap. First, the estimate of P that is plugged into the functional T_n could be replaced by other estimates. For example if one is willing to assume that the observations have a symmetric distribution around their mean one could replace $\hat{P}_n$ by a symmetrized version. Or if one is using a parametric model $\{P_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta\}$ for the observations one could use $P_{\hat{\boldsymbol{\theta}}}$ where $\hat{\boldsymbol{\theta}}$ is an estimate of the parameter $\boldsymbol{\theta}$. In the latter case one also calls the procedure parametric bootstrap. In case that the parametric model holds one may expect a better accuracy of the parametric bootstrap whereas, naturally, the "nonparametric" bootstrap is more robust against deviations from the model. We now come to another advantage of the plug-in view. It gives a good intuitive explanation when the "bootstrap works". One says that the bootstrap works or bootstrap is consistent if the difference between $T_n(\widetilde{P}_n)$ and $T_n(P)$, measured by some distance, converges to zero. Here $\widetilde{P}_n$ is some estimate of P. The Bootstrap will work when two conditions hold:

(1) The estimate $\widetilde{P}_n$ is a consistent estimate of P, i.e. $\widetilde{P}_n$ converges to P, in some appropriate sense.
(2) The functionals T_n are continuous, uniformly in n.

Consistency of the bootstrap has been proved for a broad variety of models and for a large class of different bootstrap resampling schemes. Typically for the proofs another approach has been used than (1) and (2). Using asymptotic theory often one can verify that $T_n(\widetilde{P}_n)$ and $T_n(P)$ have the same limiting distribution, see Bickel and Freedman (1981) for one of the first consistency proofs for the bootstrap. In our example if the observations have a finite variance $\sigma^2(P)$ then both $T_n(\widetilde{P}_n)$ and $T_n(P)$ have a limiting normal limit $N(0, \sigma^2(P))$. For a more general discussion of the approach based on (1) and (2), see also Beran and Ducharme (1991). The importance of (1) and (2) also lies in the fact that it gives an intuitive reasoning when the bootstrap works. For a recent discussion if assumption (2) is necessary see also Inoue and Kilian (2003).

There exist bootstrap methods that cannot be written or interpreted as plug-in estimates. This concerns different bootstrap methods where random weights are generated instead of random (pseudo) observations (see Bose and Chatterjee 2002). Or this may happen in many applications where the data model is not fully specified. Important examples are models for dependent data. Whereas classical parametric time series models specify the full dimensional distribution of the complete data vector, some non- and semi-parametric models only describe the distribution of neighbored observations. Then the full data generating process is not specified and a basic problem arises how bootstrap resamples should be generated. There are some interesting proposals around and the research on bootstrap for dependent data is still going on. We give a short introduction to this topic in Sect. 17.4. It is a nice example of an active research field on the bootstrap.

Several reasons have been given why the bootstrap should be applied. The Bootstrap can be compared with other approaches. In our example the classical approach would be to use the normal approximation $N(0, \sigma^2(\hat{P}_n))$. It has been shown that the bootstrap works if and only if the normal approximation works, see

Mammen (1992a). This even holds if the observations are not identically distributed. Furthermore, one can show that the rate of convergence of both the bootstrap and the normal approximation is $n^{-1/2}$. This result can be shown by using Edgeworth expansions. We will give a short outline of the argument. The distribution function $F(x) = P(\sqrt{n}[\overline{X}_n - \mu(P)] \le x)$ can be approximated by

$$\Phi\left(\frac{x}{\sigma(P)}\right) - \frac{1}{6\sqrt{n}}\frac{\mu_3(P)}{\sigma(P)^3}\left[\left(\frac{x}{\sigma(P)}\right)^2 - 1\right]\phi\left(\frac{x}{\sigma(P)}\right) .$$

Here, Φ is the distribution function of a standard normal distribution and ϕ is its density. $\mu_3(P) = E[X_i - \mu(P)]^3$ is the third centered moment of the observations X_i. Under regularity conditions this approximation holds with errors of order $O(n^{-1})$. For the bootstrap estimate of F a similar expansion can be shown where $\sigma(P)$ and $\mu_3(P)$ are replaced by their sample versions $\sigma(\hat{P}_n)$ and $\mu_3(\hat{P}_n) = n^{-1}\sum_{i=1}^n (X_i - \overline{X}_n)^3$

$$\Phi\left(\frac{x}{\sigma(\hat{P}_n)}\right) - \frac{1}{6\sqrt{n}}\frac{\mu_3(\hat{P}_n)}{\sigma(\hat{P}_n)^3}\left[\left(\frac{x}{\sigma(\hat{P}_n)}\right)^2 - 1\right]\phi\left(\frac{x}{\sigma(\hat{P}_n)}\right) .$$

The difference between the bootstrap estimate and F is of order $n^{-1/2}$ because the first order terms $\Phi(x/\sigma(P))$ and $\Phi(x/\sigma(\hat{P}_n))$ differ by a term of order $O_P(n^{-1/2})$ as the same holds for $\sigma(P)$ and $\sigma(\hat{P}_n)$. Thus there seems to be no asymptotic advantage in using the bootstrap compared to the classical normal approximation although the skewness of the distribution is accurately mimicked by the bootstrap. However, if the functional T_n is replaced by the distribution of the studentized statistic $\sqrt{n}\sigma(\hat{P}_n)^{-1}(\overline{X}_n - \mu(P))$ then the bootstrap achieves a rate of convergence of order n^{-1} whereas the normal approximation $N(0,1)$ still only has a rate of accuracy of order $n^{-1/2}$. Again, this can be easily seen by Edgeworth expansions. For the distribution function of the studentized statistic the following expansion holds with accuracy $O(1/n)$.

$$\Phi(x) + \frac{1}{6\sqrt{n}}\frac{\mu_3(P)}{\sigma(P)^3}\left[2x^2 + 1\right]\phi(x) .$$

The normal approximation $\Phi(x)$ differs from this expansion by terms of order $O(n^{-1/2})$. For the bootstrap estimate one gets the following expansion with error terms of order $O(1/n)$.

$$\Phi(x) + \frac{1}{6\sqrt{n}}\frac{\mu_3(\hat{P}_n)}{\sigma(\hat{P}_n)^3}\left[2x^2 + 1\right]\phi(x) .$$

This approximates the distribution function of the studentized statistic with accuracy $O_P(n^{-1})$ because $\mu_3(\hat{P}_n) - \mu_3(P) = O_P(n^{-1/2})$ and $\sigma(\hat{P}_n) - \sigma(P) = O_P(n^{-1/2})$.

That means in this case the classical normal approximation is outperformed by the bootstrap. This result for studentized statistics has been used as the main asymptotic argument for the bootstrap. It has been verified for a large class of models and resampling methods. For a rigorous and detailed discussion see Hall (1992).

There also exist some other arguments in favor of the bootstrap. For linear models with increasing dimension it has been shown in Bickel and Freedman (1983) and Mammen (1989, 1992b, 1993) that the bootstrap works under weaker conditions than the normal approximation. These results have been extended to more general sequences of models and resampling schemes, see Bose and Chatterjee (2002) and references cited therein. These results indicate that the bootstrap still may give reasonable results even when the normal approximation does not work. For many applications this type of result may be more important than a comparison of higher order performances. Higher order Edgeworth expansions only work if the simple normal approximation is quite reasonable. But then the normal approximation is already sufficient for most statistical applications because typically not very accurate approximations are required. For example an actual level .06 instead of an assumed level .05 may not lead to a misleading statistical analysis. Thus one may argue that higher order Edgeworth expansions can only be applied when they are not really needed and for these reasons they are not the appropriate methods for judging the performance of the bootstrap. On the other hand no other mathematical technique is available that works for such a large class of problems as the Edgeworth expansions do. Thus there is no general alternative way for checking the accuracy of the bootstrap and for comparing it with normal approximations.

The Bootstrap is a very important tool for statistical models where classical approximations are not available or where they are not given in a simple form. Examples arise e.g. in the construction of tests and confidence bands in nonparametric curve estimation. Here approximations using the central limit theorem lead to distributions of functionals of Gaussian processes. Often these distributions are not explicitly given and must be calculated by simulations of Gaussian processes. We will give an example in the next section (number of modes of a kernel smoother as a function of the bandwidth). Compared with classical asymptotic methods the bootstrap offers approaches for a much broader class of statistical problems.

By now, the bootstrap is a standard method of statistics. It has been discussed in a series of papers, overview articles and books. The books Efron (1982), Efron and Tibshirani (1993) and Davison and Hinkley (1997) give a very insightful introduction into possible applications of the bootstrap in different fields of statistics. The books Beran and Ducharme (1991) and Mammen (1992b) contain a more technical treatment of consistency of the bootstrap, see also Gine (1997). Higher order performance of the bootstrap is discussed in the book Hall (1992). The book Shao and Tu (1995) gives a rather complete overview on the theoretical results on the bootstrap in the mid-nineties. The book Politis et al. (1999) gives a complete discussion of the subsampling, a resampling method where the resample size is smaller than the size of the original sample. The book Lahiri (2003b) discusses the bootstrap for dependent data. Some overview articles are contained in Statistical Science (2003), Vol. 18, Number 2. Here, Efron (2003) gives a short

(re)discussion of bootstrap confidence intervals, Davison et al. (2003) report on recent developments of the bootstrap, in particular in classification, Hall (2003) discusses the roots of the bootstrap, and Boos (2003), Beran (2003) give a short introduction to the bootstrap, and other articles give an overview over recent developments of bootstrap applications in different fields of statistics. Overview articles over special bootstrap applications have been given for sample surveys (Lahiri 2003a; Shao 1996, 2003), for econometrics (Horowitz 1997, 2001, 2003a), nonparametric curve estimation (Härdle and Mammen 1991; Mammen 2000), estimating functions (Lele 2003): dependent data (Kreiss and Paparoditis 2011) and financial time series analysis (Paparoditis and Politis 2009).

17.2 Bootstrap as a Data Analytical Tool

In a data analysis the statistician wants to get a basic understanding of the stochastic nature of the data. For this purpose he/she applies several data analytic tools and interprets the results. A basic problem of a data analysis is over-interpretation of the results after a battery of statistical methods has been applied. A similar situation occurs in multiple testing but there exist approaches to capture the joint stochastics of several test procedures. The situation becomes more involved in modern graphical data analysis. The outcomes of a data analytic tool are plots and the interpretation of the data analysis relies on the interpretation of these (random) plots. There is no easy way to have an understanding of the joint distribution of the inspected graphs. The situation is already complicated if only one graph is checked. Typically it is not clearly specified for which characteristics the plot is checked. We will illustrate this by a simple example. We will argue that the bootstrap and other resampling methods offer a simple way to get a basic understanding for the stochastic nature of plots that depend on random data. In the next section we will discuss how this more intuitive approach can be translated into the formal setting of mathematical decision theoretical statistics. Our example is based on the study of a financial time series. Figure 17.1 shows the daily values of the German DAX index from end of 1993 until November 2003. In Fig. 17.2 mean-corrected log returns are shown. Logreturns for a series x_t are defined as $\log x_t - \log x_{t-1}$. Mean-corrected logreturns r_t are

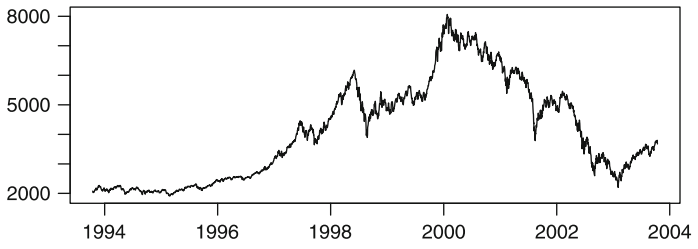


Fig. 17.1 Plot of DAX data

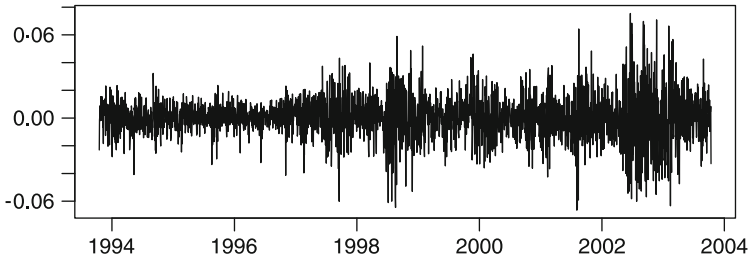


Fig. 17.2 1-lag mean-corrected logreturn of the DAX data

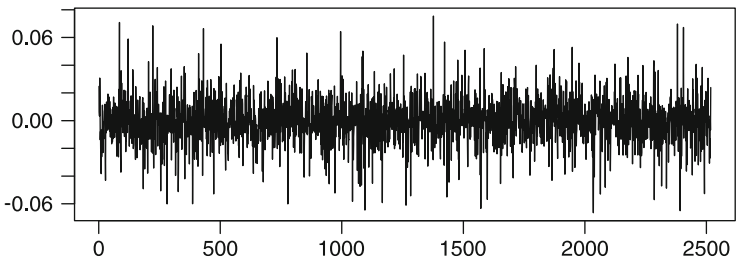


Fig. 17.3 Random permutation of the data in Fig. 17.2

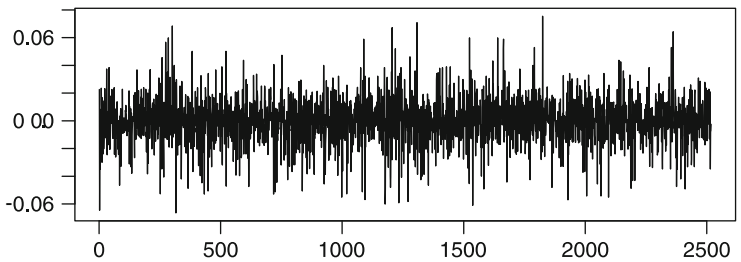


Fig. 17.4 Nonparametric bootstrap sample of the data in Fig. 17.2

defined as this difference minus its sample average. Under the Black-Sholes model the logreturns r_t are i.i.d. It belongs to folklore in finance that this does not hold. We now illustrate how this could be seen by application of resampling methods.

Figure 17.3 shows a plot of the same logreturns as in Fig. 17.2 but with changed order. The logreturns are plotted against a random permutation of the days. The clusters appearing in Fig. 17.2 dissappear. Figure 17.3 shows that these clusters could not be explained by stochastic fluctuations. The same story is told in Fig. 17.4. Here a bootstrap sample of the logreturns is shown. Logreturns are drawn with replacement from the set of all logreturns (i.i.d. resampling) and they are plotted in the order as they were drawn. Again the clusters disappear and the same happens for typical repetitions of random permutation or bootstrap plots. The clusters in Fig. 17.2 can be interpreted as volatility clusters. The volatility of a logreturn for

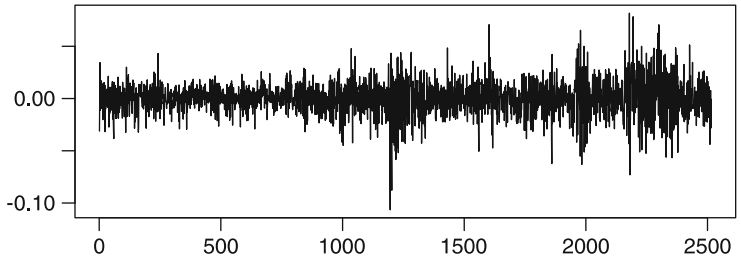


Fig. 17.5 GARCH(1,1) bootstrap sample of the data in Fig. 17.2

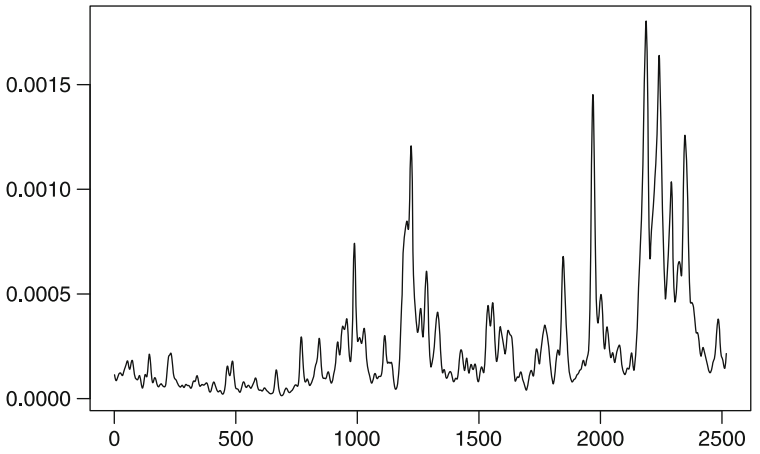


Fig. 17.6 Plot of kernel smooth of squared logreturns from the data in Fig. 17.2

a day is defined as the conditional variance of the logreturn given the logreturns of all past days. The volatilities of neighbored days are positively correlated. This results in volatility clusters. A popular approach to model the clusters are GARCH (Generalized Autoregressive Conditionally Heteroscedastic) models. In the GARCH(1,1) specification one assumes that $r_t = \sigma_t \varepsilon_t$ where ε_t are i.i.d. errors with mean zero and variance 1 and where σ_t^2 is a random conditional variance process fulfilling $\sigma_t^2 = a_0 + a_1 \sigma_{t-1}^2 + b_1 r_{t-1}^2$. Here a_0, a_1 and b_1 are unknown parameters. Figure 17.5 shows a bootstrap realization of a fitted GARCH(1,1) model. Fitted parameters $\hat{a}_0$, $\hat{a}_1$ and $\hat{b}_1$ are calculated by a quasi-likelihood method (i.e. likelihood method for normal ε_t). In the bootstrap resampling the errors ε_t are generated by i.i.d. resampling from the residuals $r_t/\hat{\sigma}_t$ where $\hat{\sigma}_t^2$ is the fitted volatility process $\hat{\sigma}_t^2 = \hat{a}_0 + \hat{a}_1 \hat{\sigma}_{t-1}^2 + \hat{b}_1 r_{t-1}^2$. An alternative resampling would to generate normal i.i.d. errors in the resampling. This type of resampling is also called parametric bootstrap. At first sight the volatility clusters in the parametric bootstrap have similar shape as in the plot of the observed logreturns. Figure 17.6 shows local averages $\hat{m}(t)$ over squared logreturns. We have chosen $\hat{m}(t)$ as Nadaraya–Watson estimate $\hat{m}_h(t) = [\sum_{s=1}^N K\{(s-t)/h\} r_t^2] / [\sum_{s=1}^N K\{(s-t)/h\}]$. We used

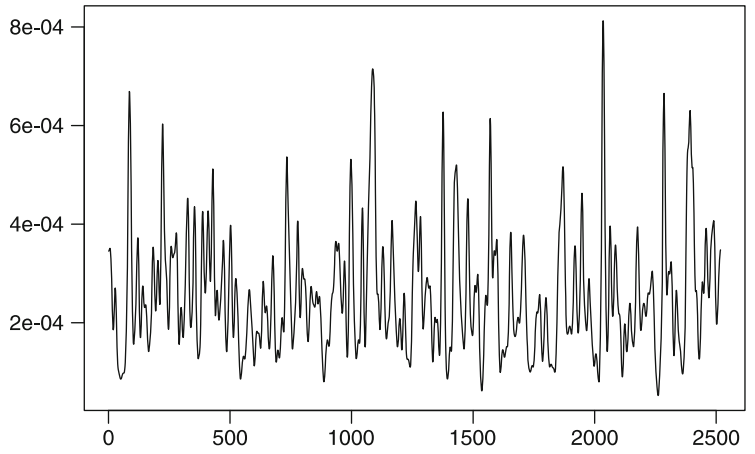


Fig. 17.7 Plot of kernel smooth of squared logreturns from the data in Fig. 17.3

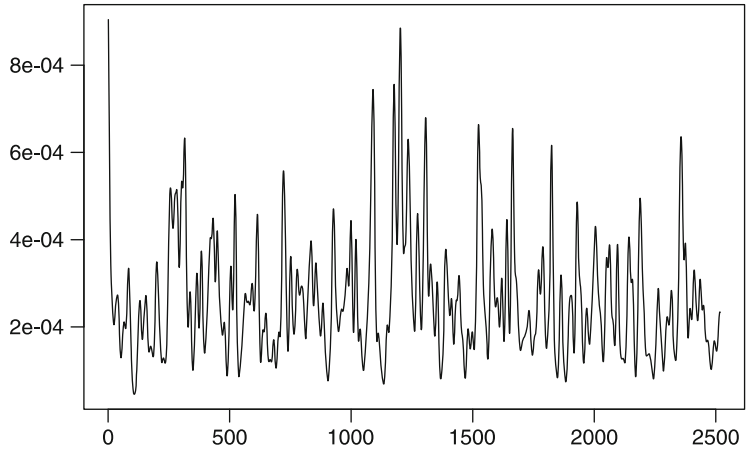


Fig. 17.8 Plot of kernel smooth of squared logreturns from the data in Fig. 17.4

a Gaussian kernel K and bandwidth $h = 5$ days. Figures 17.7–17.9 show the corresponding plots for the three resampling methods. Again the plots for random permutation resampling and nonparametric i.i.d. bootstrap qualitatively differ from the plot for the observed time series (Figs. 17.7 and 17.8). In Fig. 17.9 the GARCH bootstrap shows a qualitatively similar picture as the original logreturns ruling again not out the GARCH(1,1) model.

As a last example we consider plots that measure local and global shape characteristics of the time series. We consider the number of local maxima of the kernel smoother $\hat{m}_h$ as a function of the bandwidth h. We compare this function with the number of local maxima for resamples. Figures 17.10–17.12 show the corresponding plots for the permutation resampling, the nonparametric bootstrap

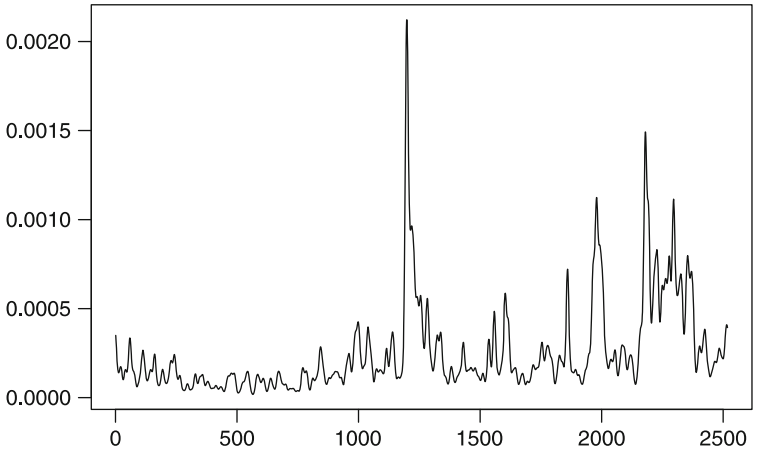


Fig. 17.9 Plot of kernel smooth of squared logreturns from the data in Fig. 17.5

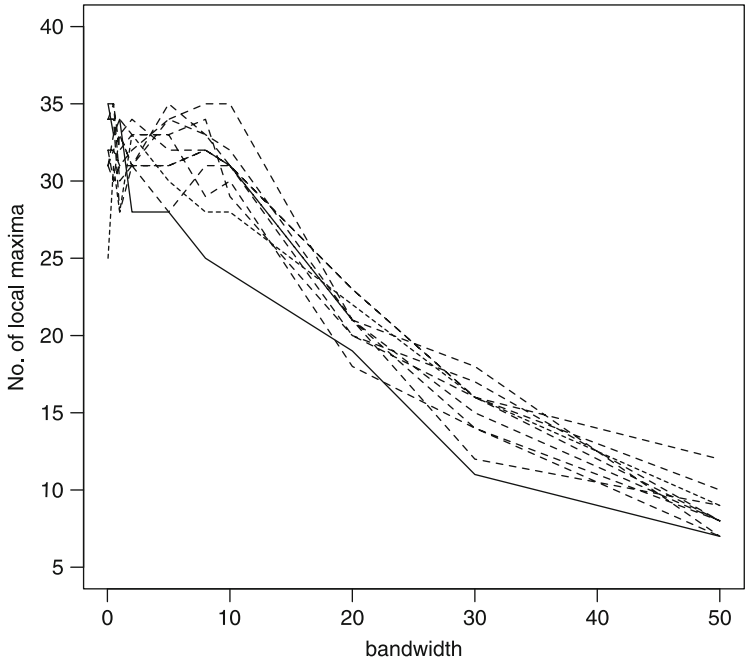


Fig. 17.10 Number of local maxima of kernel smoother $\hat{m}_h$ of squared mean-corrected logreturns of DAX data from Fig. 17.1 (*black line*) compared with number of local maxima for 10 random permutation resamples (*dashed lines*)

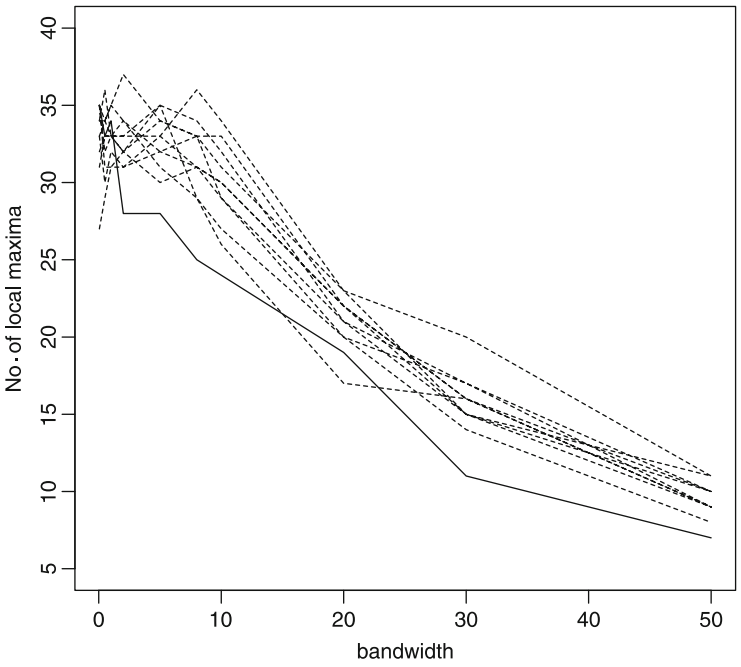


Fig. 17.11 Number of local maxima of kernel smoother $\hat{m}_h$ of squared mean-corrected logreturns of DAX data from Fig. 17.1 (*black line*) compared with number of local maxima for 10 nonparametric bootstrap resamples (*dashed lines*)

and the GARCH(1,1) bootstrap. The plot of the original data set is always compared with the plots for 10 resamples. Again i.i.d. structures are not supported by the resampling methods. GARCH(1,1) bootstrap produces plots that are comparable to the original plot.

The last approach could be formalized to a test procedure. This could e.g. be done by constructing uniform resampling confidence bands for the expected number of local maxima. We will discuss resampling tests in the next section. For our last example we would like to mention that there seems to be no simple alternative to resampling. An asymptotic theory for the number of maxima that could be used for asymptotic confidence bands is not available (to our knowledge) and it would be rather complicated. Thus, resampling offers an attractive way out. It could be used for a more data analytic implementation as we have used it here. But it could also be used for getting a formal test procedure.

The first two problems, discussed in Figs. 17.1–17.9, are too complex to be formalized as a testing approach. It is impossible to describe for what differences the human eye is looking in the plots and to summarize the differences in one simple quantity that can be used as a test statistic. The eye is using a battery of "tests" and it is applying the same or similar checks for the resamples. Thus, resampling is a good way to judge statistical findings based on the original plots.

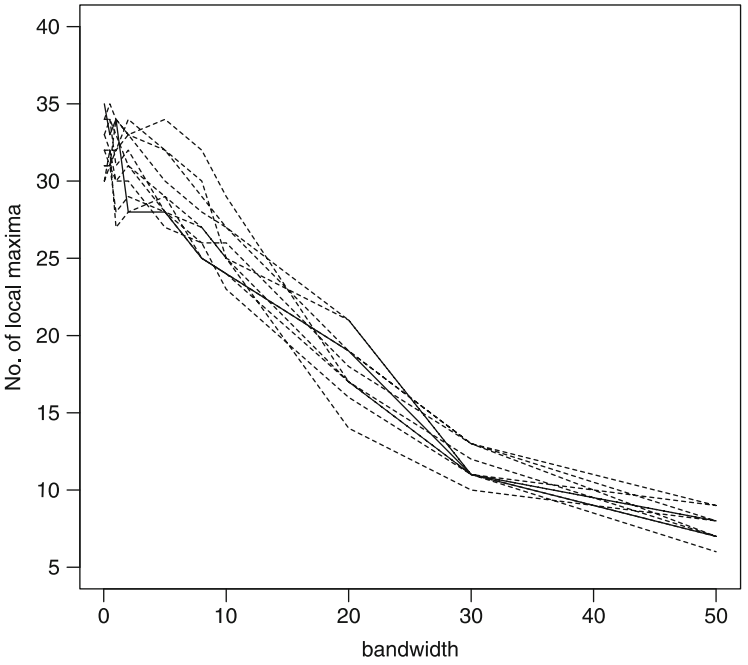


Fig. 17.12 Number of local maxima of kernel smoother $\hat{m}_h$ of squared mean-corrected logreturns of DAX data from Fig. 17.1 (*black line*) compared with number of local maxima for GARCH(1,1) bootstrap resamples (*dashed lines*)

17.3 Resampling Tests and Confidence Intervals

In the last section we have pointed out how resampling can offer additional insights in a data analysis. We now want to discuss applications of bootstrap that are more in the tradition of classical statistics. We will introduce resampling approaches for the construction of confidence intervals and of testing procedures. The majority of the huge amount of the bootstrap literature is devoted to these topics. There exist two basic approaches for the construction of confidence regions:

- Bootstrapping asymptotic pivots, bootstrap-t intervals
- Confidence intervals based on bootstrap percentiles

We will outline both methods below. There also exist two basic approaches for the construction of resampling tests:

- Resampling from the hypothesis
- Conditional tests

We will discuss testing after confidence intervals.

Approaches based on pivot statistics are classical methods for the construction of confidence sets. In a statistical model $\{P_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta\}$ a pivot statistic is a random quantity $Q = Q(\boldsymbol{\theta}, X)$ that depends on the unknown parameter $\boldsymbol{\theta}$ and on the observation (vector) X and that has the following property. The distribution of Q under $P_{\boldsymbol{\theta}}$ does not depend on $\boldsymbol{\theta}$. Thus the distribution of Q is known and one can calculate quantiles $q_{1,\alpha}, q_{2,\alpha}$ such that $P_{\boldsymbol{\theta}}\{q_{1,\alpha} \le Q(\boldsymbol{\theta}, X) \le q_{2,\alpha}\} = 1-\alpha$. Then $C_\alpha = \{\boldsymbol{\theta} \in \Theta : q_{1,\alpha} \le Q(\boldsymbol{\theta}, X) \le q_{2,\alpha}\}$ is a confidence set of the unknown parameter $\boldsymbol{\theta}$ with coverage probability $P(\boldsymbol{\theta} \in C_\alpha) = 1-\alpha$. Classical examples are i.i.d. normal observations X_i with mean μ and variance σ^2. Then $Q = (\overline{X} - \mu)/\hat{\sigma}$ is a pivot statistic. Here $\overline{X}$ is the sample mean and $\hat{\sigma}^2 = (n-1)^{-1}\sum_{i=1}^n (X_i - \overline{X})^2$ is the sample variance. Then we get, e.g. $C_\alpha = [\overline{X} - n^{-1/2}k_{1-\alpha/2}\hat{\sigma}, \overline{X} + n^{-1/2}k_{1-\alpha/2}\hat{\sigma}]$ is a confidence interval for μ with exact coverage probability $1-\alpha$. Here $k_{1-\alpha/2}$ is the $1-\alpha/2$ quantile of the t-distribution with $n-1$ degrees of freedom.

Pivot statistics only exist in very rare cases. However for a very rich class of settings one can find statistics $Q = Q(\boldsymbol{\theta}, X)$ that have a limiting distribution $\mathcal{L}(\boldsymbol{\theta})$ that smoothly depends on $\boldsymbol{\theta}$. Such statistics are called asymptotic pivot statistics. If now $q_{1,\alpha}, q_{2,\alpha}$ are chosen such that under $\mathcal{L}(\hat{\boldsymbol{\theta}})$ the interval $[q_{1,\alpha}, q_{2,\alpha}]$ has probability $1-\alpha$ then we get that $P(\boldsymbol{\theta} \in C_\alpha)$ converges to $1-\alpha$. Here $\hat{\boldsymbol{\theta}}$ is a consistent estimate of $\boldsymbol{\theta}$ and the confidence set C_α is defined as above. A standard example can be easily given if an estimate $\hat{\tau}$ of a (one-dimensional, say) parameter $\tau = \tau(\boldsymbol{\theta})$ is given that is asymptotically normal. Then $\sqrt{n}(\hat{\tau} - \tau)$ converges in distribution towards a normal limit with mean zero and variance $\sigma^2(\boldsymbol{\theta})$ depending on the unknown parameter $\boldsymbol{\theta}$. Here $Q = \sqrt{n}(\hat{\tau} - \tau)$ or the studentized version $Q = \sqrt{n}(\hat{\tau} - \tau)/\sigma(\hat{\boldsymbol{\theta}})$ with a consistent estimate $\hat{\boldsymbol{\theta}}$ of $\boldsymbol{\theta}$ could be used as asymptotic pivot. Asymptotic pivot confidence intervals are based on the quantiles of the asymptotic distribution $\mathcal{L}$ of Q. The bootstrap idea is to simulate the finite sample distribution $\mathcal{L}_n(\boldsymbol{\theta})$ of the pivot statistic Q instead of using the asymptotic distribution of Q. This distribution depends on n and on the unknown parameter $\boldsymbol{\theta}$. The bootstrap idea is to estimate the unknown parameter and to plug it in. Then bootstrap quantiles for Q are defined as the (random) quantiles of $\mathcal{L}_n(\hat{\boldsymbol{\theta}})$. For the unstudentized statistic $Q = \sqrt{n}(\hat{\tau} - \tau)$ we get the bootstrap confidence interval $[\hat{\tau} - n^{-1/2}\hat{q}_{2,\alpha}, \hat{\tau} - n^{-1/2}\hat{q}_{1,\alpha}]$ where $\hat{q}_{1,\alpha}$ is the $\alpha/2$ bootstrap quantile and $\hat{q}_{2,\alpha}$ is the $1-\alpha/2$ bootstrap quantile. This confidence interval has an asymptotic coverage probability equal to $1-\alpha$. We want to illustrate this approach by the data example of the last section. Suppose we fit a GARCH(1,1) model to the logreturns and we want to have a confidence interval for $\tau = a_1 + b_1$. It is known that a GARCH(1,1) process is covariance stationary if and only if $|\tau| < 1$. For values of τ that approximate 1, one gets a very high persistency of shocks on the process. We now construct a bootstrap confidence interval for τ. We used $Q = \sqrt{n}(\hat{\tau} - \tau)$ as asymptotic pivot statistic. The results are summarized in Table 17.1.

Table 17.1 Estimate of $a_1 + b_1$ and 90% bootstrap confidence interval using GARCH(1,1) bootstrap (asymptotic pivot method)

$\hat{a}_1 + \hat{b}_1$	Confidence lower bound	Upper bound
0.9919	0.9874	0.9960

Table 17.2 Estimate of $a_1 + b_1$ and 90% bootstrap-t confidence interval using GARCH(1,1) bootstrap for the first half and for the second half of the DAX returns (asymptotic pivot method)

	$\hat{a}_1 + \hat{b}_1$	Confidence lower bound	Upper bound
Using part I	0.9814	0.9590	0.9976
Using part II	0.9842	0.9732	0.9888

Table 17.3 Estimate of $a_1 + b_1$ and 90% bootstrap percentile confidence interval using GARCH(1,1) bootstrap

$\hat{a}_1 + \hat{b}_1$	Confidence lower bound	Upper bound
0.9919	0.9877	0.9963

We also applied the GARCH(1,1) bootstrap to the first half and to the second half of our data set. The results are summarized in Table 17.2. The value of $\hat{\tau}$ is quite similar for both halves. The fitted parameter is always contained in the confidence interval based on the other half of the sample. Both confidence intervals have a broad overlap. So there seems no reason to expect different values of τ for the two halves of the data. The situation becomes a little bit confused if we compare Table 17.2 with Table 17.1. Both fitted values of τ, the value for the first half and for the second half, are not contained in the confidence interval that is based on the whole sample. This suggests that a GARCH(1,1) model with fixed parameters for the whole sample is not an appropriate model. A model with different values seems to be more realistic. When for the whole time series a GARCH(1,1) model is fitted the change of the parameters in time forces the persistency parameter τ closer to 1 and this effect increases for GARCH fits over longer periods. We do not want to discuss this point further here and refer to Mikosch and Starica (2004) for more details.

In Efron (1979) another approach for confidence intervals was suggested. It was supposed to use the bootstrap quantiles of a test statistic directly as bounds of the bootstrap confidence intervals. In our example then the estimate $\hat{\tau}$ has to be calculated repeatedly for bootstrap resamples and the 5% and 95% empirical quantiles are used as lower and upper bound for the bootstrap confidence intervals. It can be easily checked that we then get $[\hat{\tau} + n^{-1/2}\hat{q}_{1,\alpha}, \hat{\tau} + n^{-1/2}\hat{q}_{2,\alpha}]$ as bootstrap confidence interval where the quantiles $\hat{q}_{1,\alpha}$ and $\hat{q}_{2,\alpha}$ are defined as above, see also Efron and Tibshirani (1993). Note that the interval is just reflected around $\hat{\tau}$. The resulting confidence interval for τ is shown in Table 17.3. For asymptotic normal test statistics both bootstrap confidence intervals are asymptotically equivalent. Using higher order Edgeworth expansions it was shown that bootstrap pivot intervals achieve a higher order level accuracy. Modifications of percentile intervals have been proposed that achieve level accuracy of the same order, see Efron and Tibshirani (1993). For a recent discussion on bootstrap confidence intervals see also Efron (2003), Davison et al. (2003). In our data example there is only a minor difference between the two intervals, cf. Tables 17.1 and 17.3. This may be caused by the very large sample size.

The basic idea of bootstrap tests is rather simple. Suppose that for a statistical model $\{P_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta\}$ a testing hypothesis $\boldsymbol{\theta} \in \Theta_0 \subset \Theta$ and a test statistic $T(X)$ is

given. Then bootstrap is used to calculate critical values for $T(X)$. This can be done by fitting a model on the hypothesis and by generating bootstrap resamples under the fitted hypothesis model. The $1-\alpha$ quantile of the test statistic in the bootstrap samples can be used as critical value. The resulting test is called a bootstrap test. Alternatively, a testing approach can be based on the duality of testing procedures and confidence regions. Each confidence region defines a testing procedure by using the following rule. A hypothesis is rejected if no hypothesis parameter lies in the confidence region. We shortly describe this method for bootstrap confidence intervals based on an asymptotic pivot statistic, say $\sqrt{n}(\hat{\boldsymbol{\theta}}_n - \boldsymbol{\theta})$, and the hypothesis $\Theta_0 = (-\infty, \boldsymbol{\theta}_0] \subset \mathbb{R}$. Bootstrap resamples are generated (in the unrestricted model) and are used for estimating the $1-\alpha$ quantile of $\sqrt{n}(\hat{\boldsymbol{\theta}}_n - \boldsymbol{\theta})$ by $\hat{k}_{1-\alpha}$, say. The bootstrap test rejects the hypothesis, if $\sqrt{n}(\hat{\boldsymbol{\theta}}_n - \boldsymbol{\theta}_0)$ is larger than $\hat{k}_{1-\alpha}$. Higher order performance of bootstrap tests has been discussed in Hall (1992). For a discussion of bootstrap tests we also refer to Beran (1988), Beran and Ducharme (1991).

We now compare bootstrap testing with a more classical resampling approach for testing ("conditional tests"). There exist some (important) examples where, for all test statistics, resampling can be used to achieve a correct level on the whole hypothesis for finite samples. Such tests are called similar. For some testing problems resampling tests turn out to be the only way to get similar tests. This situation arises when a statistic is available that is sufficient on the hypothesis $\{P_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta_0\}$. Then, by definition of sufficiency, the conditional distribution of the data set given this statistic is fixed on the hypothesis and does not depend on the parameter of the underlying distribution as long as the parameter lies on the hypothesis. Furthermore, because this distribution is unique and thus known, resamples can be drawn from this conditional distribution. The resampling test then has correct level on the whole hypothesis. We will now give a more formal description.

A test $\phi(X)$ for a vector X of observations is called similar if $E_{\boldsymbol{\theta}}\, \phi(X) = \alpha$ for all $\boldsymbol{\theta} \in \Theta_0$, where Θ_0 is the set of parameters on the null hypotheses. We suppose that a statistic S is available that is sufficient on the hypothesis. Let $\mathcal{P}_0 = \{P_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta_0\}$ be the family of distributions of X on the hypothesis. Then the conditional distribution of X given S does not depend on the underlying parameter $\boldsymbol{\theta} \in \Theta_0$ because S is sufficient. In particular, $E(\phi(X)|S = s)$ does not depend on $\boldsymbol{\theta}$. Then any test satisfying

$$E\left[\phi(X)|S = s\right] = \alpha \tag{17.1}$$

is similar on $\mathcal{P}_0$. This immediately follows from

$$E[\phi(X)] = EE[\phi(X)|S] = \alpha\ .$$

A test satisfying (17.1) is said to have Neyman structure with respect to S.

For a given test statistic T similar tests can be constructed by choosing $k_\alpha(S)$ such that

$$P[T > k_\alpha(S)|S = s] = \alpha\ . \tag{17.2}$$

Here the conditional probability on the left hand side does not depend on $\boldsymbol{\theta}$ because S is assumed to be sufficient. We now argue that this is the only way to construct similar tests if the family of distributions of S (for $\boldsymbol{\theta} \in \Theta_0$) is "rich enough". For such families $E_{\boldsymbol{\theta}} u(S) = 0$ for all $\boldsymbol{\theta} \in \Theta_0$ for a function u implies $u(s) \equiv 0$. In particular, with $u(s) = P[T > k_\alpha(S)|S = s] - \alpha$ we get that the test $T > k_\alpha(S)$ is similar if $E_{\boldsymbol{\theta}} u(S) = 0$ for all $\boldsymbol{\theta} \in \Theta_0$. This implies $u(s) \equiv 0$, i.e. (17.2). Thus, given a test statistic T, the only way to construct similar tests is by choosing $k_\alpha(S)$ according to (17.2). The relation between similar tests and tests with Neyman structure belongs to the classical material of mathematical statistics and can be found in text books, e.g. Lehmann (1986).

We will consider two examples of conditional tests. The first one are permutation tests. For a sample of observations $X = (X_1, \ldots, X_n)$ the order statistic $S = (X_{(1)}, \ldots, X_{(n)})$ containing the ordered sample values $X_{(1)} \leq \ldots \leq X_{(n)}$ is sufficient on the hypothesis of i.i.d. observations. Given S, the conditional distribution of X is a random permutation of $X_1, \ldots, X_n$. The resampling scheme is very similar to the nonparametric bootstrap. In the resampling, n pseudo observations are drawn from the original data sample. Now this is done without replacement whereas in the bootstrap scheme this is done with replacement. For a comparison of bootstrap and permutation tests see also Janssen and Pauls (2003). Also for the subsampling (i.e. resampling with a resample size that is smaller than the sample size) both schemes (with and without replacement) have been considered. For a detailed discussion of the subsampling without replacement see Politis et al. (1999).

The second example is a popular approach in the physical literature on nonlinear time series analysis. For odd sample size n a series $X_1, \ldots, X_n$ can be written as

$$X_t = \overline{X} + \sqrt{\frac{2\pi}{n}} \sum_{j=1}^{(n-1)/2} 2\sqrt{I_X\left(\frac{2\pi j}{n}\right)} \cos\left(\frac{2\pi j}{n} t + \boldsymbol{\theta}_j\right)$$

with sample mean $\overline{X}$, periodogram $I_X(\omega) = \frac{1}{2\pi n} \left|\sum_{t=1}^n X_t \exp(-i\omega t)\right|^2$ and phases $\boldsymbol{\theta}_j$. On the hypothesis that X is a circular stationary Gaussian process the statistic $S = (\overline{X}, I_X(2\pi j/n) : j = 1, \ldots, (n-1)/2)$ is sufficient. Conditional on S, the phases $\boldsymbol{\theta}_j$ are conditional i.i.d. and have a uniform distribution on $[0, 2\pi]$. Resamples with observations

$$X_t^* = \overline{X} + \sqrt{\frac{2\pi}{n}} \sum_{j=1}^{(n-1)/2} 2\sqrt{I_X\left(\frac{2\pi j}{n}\right)} \cos\left(\frac{2\pi j}{n} t + \boldsymbol{\theta}_j^*\right),$$

where $\boldsymbol{\theta}_j^*$ are i.i.d. with uniform distribution are called "surrogate data". They can be used to construct similar tests for the hypothesis of circular stationary Gaussian processes. In the physics literature these tests are applied for testing the hypothesis of stationary Gaussian processes. It is argued that for tests that do not heavily depend on boundary observations the difference between stationarity and circular stationarity becomes negligible for large data sets. Surrogate data tests are used as

first checks if deterministic nonlinear time series models are appropriate for a data set. The relation of surrogate data tests to conditional tests was first observed in Chan (1997). The method of surrogate data was first proposed in Theiler et al. (1992).

We would like to highlight a major difference between bootstrap and conditional tests. Bootstrap tests work if they are based on resampling of an asymptotic pivot statistic. Then the bootstrap critical values stabilize asymptotically and converge against the quantile of the limiting distribution of the test statistic. For conditional tests the situation is quite different. They work for all test statistics. However, not for all test statistics it is guaranteed that the critical value $k_\alpha(S)$ converges to a deterministic limit. In Mammen and Nandi (2004) this is discussed for surrogate data tests. It is shown that also for very large data sets the surrogate data quantile $k_\alpha(S)$ may have a variance of the same order as the test statistic T. Thus the randomness of $k_\alpha(S)$ may change the nature of a test. This is illustrated by a test statistic for kurtosis of the observations that is transformed to a test for circular stationarity.

17.4 Bootstrap for Dependent Data

The Bootstrap for dependent data is a lively research area. A lot of ideas are around and have let to quite different proposals. In this section we do not want to give a detailed overview and description of the different proposals. We only want to sketch the main ideas. Models for dependent data may principally differ from i.i.d. models. For dependent data the data generating process is often not fully specified. Then there exists no unique natural way for resampling. The resampling should be carried out in such a way that the dependence structure should be captured. This can be easily done in case of classical finite-dimensional ARMA models with i.i.d. residuals. In these models the resamples can be generated by fitting the parameters and by using i.i.d. residuals in the resampling. We will discuss the situation when no finite-dimensional model is assumed. For other overviews on the bootstrap for time series analysis, see Buhlmann (2002), Härdle et al. (2003), Politis (2003) and the time series chapter in Davison et al. (2003) and the book Lahiri (2003b). In particular, Härdle et al. (2003) give an overview over the higher order performance of the different resampling schemes.

The most popular bootstrap methods for dependent data are block, sieve, local, wild and Markov bootstrap and subsampling. They all are nonparametric procedures.

17.4.1 The Subsampling

The method that works under a minimal amount of assumptions is the subsampling. It is used to approximate the distribution of an estimate $\hat{\boldsymbol{\theta}}_n$ estimating an unknown

parameter $\boldsymbol{\theta}$. In the subsampling subsamples of consecutive observations of length $l < n$ are taken. These subsamples are drawn randomly from the whole time series. For the subsamples estimates $\hat{\boldsymbol{\theta}}^*$ are calculated. If it is known that for a sequence a_n the statistic $a_n(\hat{\boldsymbol{\theta}}_n - \boldsymbol{\theta})$ has a limiting distribution then under very weak conditions the conditional distribution of $a_l(\hat{\boldsymbol{\theta}}^* - \hat{\boldsymbol{\theta}}_n)$ has the same limiting distribution. Higher order considerations show that the subsampling has a very poor rate of convergence, see Hall and Jing (1996). It does not even achieve the rate of convergence of a normal approximation. It may be argued that this poor performance is the price for its quite universal applicability. Subsampling has also been used in i.i.d. settings where classical bootstrap does not work. For a detailed discussion of the subsampling see Politis et al. (1999).

17.4.2 The Block Bootstrap

The basic idea of the block bootstrap is closely related to the i.i.d. nonparametric bootstrap. Both procedures are based on drawing observations with replacement. In the block bootstrap however instead of single observations blocks of consecutive observations are drawn. This is done to capture the dependence structure of neighbored observations. Different versions of this idea have been proposed in Hall (1985), Carlstein (1986), Künsch (1989), Liu and Singh (1992b) and Politis and Romano (1994). It has been shown that this approach works for a large class of stationary processes. The blocks of consecutive observations are drawn with replacement from a set of blocks. In the first proposal this was done for a set of nonoverlapping blocks of fixed length l: $\{X_j : j = 1, \ldots, l\}$, $\{X_{l+j} : j = 1, \ldots, l\}, \ldots$ Later papers proposed to use all (also overlapping) blocks of length l, i.e. the k-th block consists of the observations $\{X_{k-1+j} : j = 1, \ldots, l\}$ (Moving block bootstrap). The bootstrap resample is obtained by sampling n/l blocks randomly with replacement and putting them together to a time series of length n. By construction, the bootstrap time series has a nonstationary (conditional) distribution. The resample becomes stationary if the block length l is random and generated from a geometric distribution. This version of the block bootstrap is called the stationary bootstrap and was introduced in Politis and Romano (1994). Recently, Paparoditis and Politis (2001a, 2002a) proposed another modification that uses tapering methods to smooth the effects of boundaries between neighbored blocks. With respect to higher order properties the moving block bootstrap outperforms the version with non overlapping blocks and both achieve a higher order accuracy as the stationary bootstrap (see Hall et al. 1995; Lahiri 1999a,b, 2003b).

The block bootstrap has turned out as a very powerful method for dependent data. It does not achieve the accuracy of the bootstrap for i.i.d. data but it outperforms the subsampling. It works reasonably well under very weak conditions on the dependency structure. It has been applied to a very broad range of applications. For the block bootstrap no specific assumption is made on the structure of the data generating process.

We now describe some methods that use more specific assumptions on the dependency structure.

17.4.3 The Sieve Bootstrap

The i.i.d. resampling can also be applied to models of dependent data where the stochastics is driven by i.i.d. innovations. The distribution of the innovations can be estimated by using fitted residuals. In the resampling i.i.d. innovations can be generated by i.i.d. resampling from this fitted distribution. An example is an autoregressive linear model:

$$X_t - \mu_X = \sum_{j=1}^{p} \rho_j \left(X_{t-j} - \mu_X\right) + \varepsilon_t \ , \quad t \in \mathbb{Z} \tag{17.3}$$

where $\mu_X = E(X_t)$ is the observation mean and where $\{\varepsilon_t\}$ is a sequence of i.i.d. innovations with $E(\varepsilon_t) = 0$ and ε_t is independent of $\{X_s, s < t\}$. The parameters $\rho_1, \ldots, \rho_p$ can be estimated by least squares or by using Yule-Walker equations. Residuals can be fitted by putting

$$\tilde{\varepsilon}_t = X_t - \hat{\mu}_X - \sum_{j=1}^{p} \hat{\rho}_j \left(X_{t-j} - \hat{\mu}_X\right) \ ,$$

where $\hat{\mu}_X = n^{-1} \sum_{t=1}^{n} X_t$ and $\hat{\rho}_1, \ldots, \hat{\rho}_p$ are the fitted parameters. Bootstrap resamples can be generated by

$$X_t^* - \hat{\mu}_X = \sum_{j=1}^{p} \hat{\rho}_j \left(X_{t-j}^* - \hat{\mu}_X\right) + \varepsilon_t^* \tag{17.4}$$

where ε_t^* are drawn with replacement from the estimated centered residuals $\hat{\varepsilon}_t = \tilde{\varepsilon}_t - n^{-1} \sum_{i=1}^{n} \tilde{\varepsilon}_i$. For a study of this bootstrap procedure in model (17.3), see e.g. Franke and Kreiss (1992) and references cited therein.

In a series of papers this approach has been studied for the case that model (17.3) only approximately holds. This is the case if the underlying time series is a stationary linear process, i.e. $\{X_t\}$ has an infinite order autoregressive representation:

$$X_t - \mu_X = \sum_{j=1}^{\infty} \rho_j \left(X_{t-j} - \mu_X\right) + \varepsilon_t \ , \quad t \in \mathbb{Z} \ . \tag{17.5}$$

The bootstrap scheme (17.4) has been proposed for this AR(∞) model. In a first step a model (17.3) of finite order p is fitted to the time series. Bootstrap resamples are generated as in (17.4) according to model (17.3). This resampling scheme has been called the sieve bootstrap because the AR(∞) model (17.5) is approximated by an AR(p) model, where, in the asymptotics, p converges to infinity for increasing sample size n. It is argued that this asymptotic approach reflects practical uses of

AR models where the order p is selected data adaptively and one is only thinking of the finite order AR model as an approximation to the truth. The Sieve bootstrap and its asymptotic consistency was first considered by Kreiss (1988, 1992) and further analyzed by Buhlmann (1997), Buhlmann (1998), Bickel and Buhlmann (1999), Paparoditis (1996), Park (2002), Choi and Hall (2000) showed that under appropriate conditions the sieve bootstrap achieves nearly the rates of convergence of the i.i.d resampling. In particular, it usually outperforms the block bootstrap. Buhlmann (1997) studied higher order performance of sieve bootstrap variance estimates for the sample mean under assumptions on the decay of the coefficients $\rho_j \leq cj^{-v}$ for constants $c > 0$ and $v > 2$.

17.4.4 The Nonparametric Autoregressive Bootstrap

Another residual based bootstrap scheme has been proposed for a nonparametric autoregression model:

$$X_t = m(X_{t-1}, \ldots, X_{t-p}) + \sigma(X_{t-1}, \ldots, X_{t-q})\varepsilon_t \quad t = 1, 2 \ldots \tag{17.6}$$

where $\{\varepsilon_t\}$ is a sequence of i.i.d. error variables with zero mean and unit variance and where m and σ are unknown smooth functions. The functions m and σ can be estimated by nonparametric smoothing estimates $\hat{m}$ and $\hat{\sigma}$. These estimates can be used to fit residuals. In the nonparametric autoregressive bootstrap resamples are generated

$$X_t^* = \tilde{m}\left(X_{t-1}^*, \ldots, X_{t-p}^*\right) + \tilde{\sigma}\left(X_{t-1}^*, \ldots, X_{t-q}^*\right)\varepsilon_t^* \quad t = 1, 2 \ldots$$

where $\tilde{m}$ and $\tilde{\sigma}$ are nonparametric smoothing estimates and where ε_t^* are drawn with replacement from the centered fitted residuals. The choice of the bootstrap autoregression function $\tilde{m}$ and of the bootstrap volatility function $\tilde{\sigma}^2$ is rather delicate because inappropriate choices can lead to explosive dynamics for the bootstrap time series. The nonparametric autoregressive bootstrap was discussed in Franke et al. (2002a). They give conditions under which this bootstrap approach is consistent. Franke et al. (2002b) used this bootstrap approach for the construction of uniform confidence bands for the regression function m.

17.4.5 The Regression-type Bootstrap, the Wild Bootstrap and the Local Bootstrap

Franke et al. (2002a) also consider two other bootstrap procedures for the model (17.6): the regression bootstrap and the wild bootstrap. In the regression bootstrap, a nonparametric regression model is generated with (conditionally) fixed

design. We describe this approach for the case of a homoscedasstic autoregression model:

$$X_t = m(X_{t-1}, \ldots, X_{t-p}) + \varepsilon_t \quad t = 1, 2, \ldots \tag{17.7}$$

where again $\{\varepsilon_t\}$ is a sequence of i.i.d. error variables with zero mean and m is an unknown smooth autoregression function. Bootstrap error variables ε_t^* can be generated by drawing with replacement from centered fitted residuals in model (17.7). In contrast to the autoregression bootstrap the resamples are now generated in a regression model

$$X_t^* = \tilde{m}(X_{t-1}, \ldots, X_{t-p}) + \varepsilon_t^* \quad t = 1, 2, \ldots , \tag{17.8}$$

where $\tilde{m}$ is again a nonparametric smoothing estimate of m. The stochastic behavior of the autoregression estimates in model (17.7) is fitted by the bootstrap regression estimates in (17.8). Thus regression of X_t onto $(X_{t-1}, \ldots, X_{t-p})$ is mimicked in the bootstrap by regression of X_t^* onto the same covariable $(X_{t-1}, \ldots, X_{t-p})$. The regression bootstrap principally differs from the autoregressive bootstrap because no autoregressive scheme is generated in the resampling. Because the original time series is used as covariables in a regression problem the regression bootstrap has the advantage that there is no danger for the bootstrap process to be unstable or to explode. Thus the choice of the bootstrap error distribution and of the estimate $\tilde{m}$ is not so crucial as for the autoregression bootstrap. On the other hand the randomness of the covariables is not mimicked in the resampling. This leads to a poorer finite sample performance, see Franke et al. (2002a).

Modifications of the regression bootstrap are the local bootstrap (Paparoditis and Politis 2000) and the wild bootstrap. The wild bootstrap also uses a regression model with (conditionally) fixed covariables. But it is designed to work also for heteroscedastic errors. It has been first proposed for regression models with independent but not identically distributed error variables, see Wu (1986), Beran (1986). For nonparametric models it was first proposed in Härdle and Mammen (1993). In the nonparametric autoregression model (17.7) wild bootstrap resamples are generated as in (17.8). But now the error variables ε_t^* are generated as $\varepsilon_t^* = \hat{\varepsilon}_t \eta_t$ where $\hat{\varepsilon}_t$ are centered fitted residuals and where $\eta_1, \ldots, \eta_n$ are (conditionally) i.i.d. variables with conditional zero mean and conditional unit variance (given the original sample). For achieving higher order accuracy it has also been proposed to use η_t with conditional third moment equal to 1. One could argue that in this resampling scheme the distribution of ε_t is fitted by the conditional distribution of η_t. Then n different distributions are fitted in a model where only n observations are available. This is the reason why in Härdle and Mammen (1993) this approach was called wild bootstrap. For a more detailed discussion of the wild bootstrap, see Liu (1988), Liu and Singh (1992a), Mammen (1992a,b, 1993). The asymptotic analysis of the wild bootstrap and other regression type bootstrap methods in model (17.7) is much simpler than the autoregression bootstrap. In the bootstrap world it only requires mathematical analysis of a nonparametric regression model. Only the discussion of uniform nonparametric confidence bands remains rather complicated because it

involves strong approximations of the bootstrap nonparametric regression estimates by Gaussian processes, see Neumann and Kreiss (1988). The wild bootstrap works under quite weak model assumptions. Essentially it is only assumed that the conditional expectation of an observation given the past is a smooth function of the last p observations (for some finite p). Generality has its price. Resampling schemes that use more detailed modeling may achieve a better accuracy. We now consider resampling under the stronger assumption that not only the mean but also the whole conditional distribution of an observation smoothly depends on the last p observations (for some finite p). Resampling schemes that work under this smooth Markov assumption are the Markov Bootstrap schemes.

17.4.6 The Markov Bootstrap

We discuss the Markov bootstrap for a Markov model of order 1. We will describe two implementations of the Markov bootstrap. For both implementations one has to assume that the conditional distribution of X_{t+1} given $X_1, \ldots, X_t$ smoothly depends on X_t. The first version was introduced by Rajarshi (1990). It is based on a nonparametric estimate of the transition density $f(y|x)$ of $X_{t+1} = y$ given $X_t = x$. Using kernel density estimates of the density of X_t and of the joint density of (X_t, X_{t+1}) one can estimate $f(y|x)$ by

$$\hat{f}(y|x) = \frac{\hat{f}(x, y)}{\hat{f}(x)} ,$$

where

$$\hat{f}(x, y) = \frac{1}{n-1} \sum_{t=1}^{n-1} K_h(X_t - x) K_g(X_{t+1} - y) ,$$

$$\hat{f}(x) = \frac{1}{n} \sum_{t=1}^{n} K_h(X_t - x)$$

are kernel density estimates with kernel functions $K_r(u) = r^{-1} K\left(r^{-1} u\right)$ for bandwidths $r = h, g$. In the bootstrap resampling one starts with an observation X_1^* from the density $\hat{f}(\cdot)$ and then one iteratively generates X_{t+1}^* by sampling from $\hat{f}(\cdot|X_t^*)$. Higher order performance of this resampling scheme has been discussed in Horowitz (2003b). It turns out that it achieves faster rates of convergence compared with the block bootstrap. This is in accordance with intuition because the Markov bootstrap requires an additional model assumption, namely the Markov property.

The second version of the Markov bootstrap can be described as a limiting version of the latter for $g \to 0$. Then in the limiting case the bootstrap process takes values only in the set of observations $\{X_1, \ldots, X_n\}$. Given $X_t^* = x$, the next

observation X^*_{t+1} is equal to X_s $(2 \le s \le n)$ with probability $K_h(X_{s-1} - x)/\sum_{r=1}^{n-1} K_h(X_r - x)$. This resampling scheme was introduced in Paparoditis and Politis (2001b, 2002b). Higher order properties are not yet known. It may be expected that it has similar asymptotic properties as the smoothed version of the Markov bootstrap. The unsmoothed version has the advantage that the bootstrap time series is forced to live on the observed values of the original time series. This leads to a more stable dynamic of the bootstrap time series, in particular for smaller sample sizes. Furthermore, for higher dimensional Markov processes the unsmoothed version is based on only d dimensional kernel density smoothing whereas smoothed bootstrap requires $2d$ dimensional kernel smoothing. Here, d denotes the dimension of the Markov process. Again, one can argue that this leads to a more stable finite sample performance of unsmoothed bootstrap. On the other hand, the smoothed Markov bootstrap takes advantage of smoothness of $f(y|x)$ with respect to y. For larger data sets this may lead to improvements, in case of smooth transition densities.

17.4.7 The Frequency Domain Bootstrap

For the periodogram $I_X(\omega) = \frac{1}{2\pi n}\left|\sum_{t=1}^{n} X_t \exp(-i\omega t)\right|^2$ it is known that its values for $\omega_j = 2\pi j/n$, $0 < j < n/2$ are asymptotically independent. For the first two moments one gets that for $0 < j, k < n/2$, $j \neq k$

$$\mathrm{E}[I_X(\omega_j)] = f(\omega_j) + o(n^{-1/2}) , \tag{17.9}$$

$$\mathrm{Var}[I_X(\omega_j)] = f(\omega_j)^2 + o(1) , \tag{17.10}$$

$$\mathrm{Cov}[I_X(\omega_j), I_X(\omega_k)] = n^{-1} f(\omega_j) f(\omega_k) \left[\frac{\mathrm{E}\left[\varepsilon_j^4\right]}{\mathrm{E}\left[\varepsilon_j^2\right]^2} - 3 \right] + o(n^{-1}) , \tag{17.11}$$

where $f(\omega) = (2\pi)^{-1} \sum_{k=-\infty}^{\infty} \mathrm{Cov}(X_t, X_{t+k}) \exp(-\mathrm{i}k\omega)$ is the spectral density of the time series X_t and where $\varepsilon_j = X_j - \mathrm{E}\,[X_j | X_t : t \le j-1]$ are the innovations of the time series. These expansions hold under some regularity conditions on X_t. In particular, it is needed that X_t is a linear process. Thus approximately, we get that $\eta_j = I_X(\omega_j)/f(\omega_j)$, $0 < j < n/2$ is an i.i.d. sequence. This suggests the following bootstrap scheme, called the frequency domain bootstrap or the periodogram bootstrap.

In this resampling bootstrap values $I^*_X(\omega_j)$ of the periodogram $I_X(\omega_j)$ are generated. The resampling uses two estimates $\hat{f}$ and $\tilde{f}$ of the spectral density. In some implementations these estimates can be chosen identically. The first estimate is used for fitting residuals $\hat{\eta}_j = I_X(\omega_j)/\hat{f}(\omega_j)$. The bootstrap residuals $\eta^*_1, \ldots$ are drawn with replacement from the centered fitted residuals $\hat{\eta}_j/\hat{\eta}_\cdot$ where $\hat{\eta}_\cdot$ is the

average of $\hat{\eta}_j$ over $0 < j < n/2$. The bootstrap periodogram is then calculated by putting $I_X^*(\omega_j) = \tilde{f}(\omega_j)\eta_j^*$.

The frequency domain bootstrap can be used to estimate the distribution of statistics $n^{-1/2}\sum_{0<j<n/2} w_j I_X(\omega_j)$. Then the distribution of $n^{-1/2}\sum_{0<j<n/2}[w_j I_X(\omega_j) - w_j f(\omega_j)]$ is estimated by the conditional distribution of $n^{-1/2}\sum_{0<j<n/2}[w_j I_X^*(\omega_j) - w_j \tilde{f}(\omega_j)]$. Unfortunately, in general this approach does not work. This can be easily seen by a comparison of the asymptotic variances of the statistics. The original statistic $n^{-1/2}\sum_{0<j<n/2} w_j I_X(\omega_j)$ has variance that is asymptotically equivalent to

$$n^{-1}\sum w_j^2 f(\omega_j)^2 + \left[\frac{\mathrm{E}\left[\varepsilon_j^4\right]}{\mathrm{E}\left[\varepsilon_j^2\right]^2} - 3\right]\left[n^{-1}\sum w_j f(\omega_j)\right]^2 ,$$

see (17.9)–(17.11). In the bootstrap world the variance is approximately

$$n^{-1}\sum w_j^2 \tilde{f}(\omega_j)^2 .$$

Thus in general there are differences between the variances that do not vanish asymptotically. The reason is that the term on the right hand side of (17.11) contributes an additional term to the variance for the original time series. This term does not appear in the bootstrap because an i.i.d. resampling is used that produces conditionally uncorrelated $I_X^*(\omega_j)$.

Although the frequency domain bootstrap does not work in general, there exist three important examples where it works. In all three examples the second term in the asymptotic expansion of the variance vanishes. This happens e.g. if the kurtosis of the innovations is equal to zero:

$$\frac{\mathrm{E}\left[\varepsilon_j^4\right]}{\mathrm{E}\left[\varepsilon_j^2\right]^2} - 3 = 0 .$$

In particular, this is the case if the innovations have a normal distribution. Another more general example where the bootstrap works is given by statistics where it holds that $n^{-1}\sum w_j f(\omega_j) = o(1)$. A large class of examples for this case are ratio statistics

$$n^{1/2}\frac{\sum_{0<j<n/2} r_j I_X(\omega_j)}{\sum_{0<j<n/2} I_X(\omega_j)} .$$

By some Taylor expansion calculus one can see that

$$n^{1/2}\left[\frac{\sum_{0<j<n/2} r_j I_X(\omega_j)}{\sum_{0<j<n/2} I_X(\omega_j)} - \frac{\sum_{0<j<n/2} r_j f(\omega_j)}{\sum_{0<j<n/2} f(\omega_j)}\right]$$
$$\approx n^{-1/2} \sum_{0<j<n/2} \left[w_j I_X(\omega_j) - w_j f(\omega_j)\right]$$

with w_j proportional to $r_j - \sum_k r_k f(\omega_k) \big/ \sum_k f(\omega_k)$. Then $\sum_j w_j f(\omega_j) = 0$ and the bootstrap consistently estimates the variance of the ratio statistic. Consistency of the frequency domain bootstrap for ratio statistics has been shown in Dahlhaus and Janas (1996). They also showed that the frequency domain bootstrap achieves higher order accuracy. But for this it is necessary that the third moment of the innovations vanishes. This is a rather restrictive assumption. Examples of ratio statistics are autocorrelation estimates, see Dahlhaus and Janas (1996) where other examples are also given. Modifications of the frequency domain bootstrap have been proposed that work for a larger class of statistics. An example is the proposal of Kreiss and Paparoditis (2003) where ideas of the frequency domain bootstrap are combined with ideas of the the sieve bootstrap, see also the discussion in Kreiss and Paparoditis (2011) for more recent modifications of this proposal.

There exists also another example where the frequency domain bootstrap works. Nonparametric smoothing estimates of the spectral density are linear statistics where the weights w_j are now local. For example for kernel smoothing weights $w_j = h^{-1}K[(\omega_j - x)/h]$ with bandwidth h and kernel function K one has $n^{-1}\sum_j w_j^2 f(\omega_j)^2 = O(h^{-1})$. On the other hand, $n^{-1}\sum_j w_j f(\omega_j) = O(1)$ is of lower order. Now, both the variance of the original spectral density estimate and the variance of the bootstrap spectral density estimate have variance that is up to terms of order $o(h)$ is equal to the same quantity $(2\pi)^2 n^{-1}\sum w_j^2 f(\omega_j)^2$. The correlation between $I_X(\omega_j)$ and $I_X(\omega_k)$ for $j \neq k$ (see (17.11)) only contributes to higher order terms. Franke and Härdle (1992) firstly observed this relation and used this fact to show that the frequency domain bootstrap works for nonparametric spectral density estimation. In their approach, both $\hat{f}$ and $\tilde{f}$ are nonparametric kernel smoothing estimates. For $\tilde{f}$ a bandwidth has been chosen that is of larger order than the bandwidth h. Then bootstrap consistently estimates the bias of the spectral density estimate. Similar approaches have been used in bootstrap schemes for other settings of nonparametric curve estimation, see Mammen (2000). For the frequency domain bootstrap for parametric problems one can choose $\hat{f} = \tilde{f}$, see Dahlhaus and Janas (1996).

We now have discussed a large class of resampling schemes for dependent data. They are designed for different assumptions on the dependency structure ranging from quite general stationarity assumptions (subsampling), mixture conditions (block bootstrap), linearity assumptions (sieve bootstrap, frequency domain bootstrap), conditional mean Markov property (wild bootstrap), Markov properties (Markov bootstrap) and autoregressive structure (autoregressive bootstrap). It may be generally conjectured that resampling schemes for more restrictive models are more accurate as long as these more restrictive assumptions really apply. These

conjectures are supported by asymptotic results based on higher order Edgeworth expansions. (Although these results should be interpreted with care because of the poor performance of higher order Edgeworth expansions for finite samples, see also the discussion in the introduction.) The situation is also complicated by the fact that in time series analysis typically models are used as approximations to the truth and they are not interpreted as true models. Thus one has to study the much more difficult problem how resampling schemes perform if the underlying assumptions are only approximately fulfilled.

Resampling for dependent data has stimulated very creative ideas and discussions and it had lead to a large range of different approaches. Partially, the resampling structure is quite different from the stochastic structure of the original time series. In the regression bootstrap regression data are used instead of autoregression series. In the sieve bootstrap and in the frequency domain bootstrap models are used that only approximate the original model.

For dependent data the bootstrap has broadened the field of possible statistical applications. The bootstrap offered new ways of implementing statistical procedures and made it possible to treat new types of applied problems by statistical inference.

The discussion of thebootstrap for dependent data is not yet finished. For the comparison of the proposed resampling schemes a complete understanding is still missing and theoretical research is still going on. Applications of time series analysis will also require new approaches. Examples are unit root tests, cointegration analysis and the modeling of financial time series. See also Kreiss and Paparoditis (2011) for an overview on the recent developments in this area.

References

Beran, R.: Discussion of Wu, C.F.J., Jackknife, bootstrap, and other resampling methods in regression analysis (with discussion). Ann. Stat. **14**, 1295–1298 (1986)

Beran, R.: Prepivoting test statistics: A bootstrap view of asymptotic refinements. J. Amer. Stat. Assoc. **83**, 687–697 (1988)

Beran, R.: The impact of the bootstrap on statistical algorithms and theory. Stat. Sci. **18**, 175–184 (2003)

Beran, R., Ducharme, G.: Asymptotic Theory for Bootstrap Methods in Statistics, Les Publications CRM, University Montreal (1991)

Bickel, P.J., Bühlmann, P.: A new mixing notion and functional central limit theorems for a sieve bootstrap in time series. Bernoulli **5**, 413–446 (1999)

Bickel, P., Freedman, D.: Some asymptotic theory for the bootstrap. Ann. Stat. 9:1196–1217 (1981)

Bickel, P., Freedman, D.: Bootstrapping regression models with many parameters. In: Bickel, P.J., Doksum, K.A., Hodges, J.C.: (eds.) A Festschrift for Erich L. Lehmann, pp. 28–48. Wadsworth, Belmont (1983)

Boos, D.D.: Introduction to the bootstrap world. Stat. Sci. **18**, 168–174 (2003)

Bose, A., Chatterjee, S.: Dimension asymptotics for generalized bootstrap in linear regression. Ann. Inst. Stat. Math. **54**, 367–381 (2002)

Bühlmann, P.: Sieve bootstrap for time series. Bernoulli **3**, 123–148 (1997)

Bühlmann, P.: Sieve bootstrap for smoothing in nonstationary time series. Ann. Stat. **26**, 48–83 (1998)

Bühlmann, P.: Bootstraps for time series. Stat. Sci. **17**, 52–72 (2002)
Carlstein, E.: The use of subseries methods for estimating the variance of a general statistic from a stationary time series. Ann. Stat. **14**, 1171–1179 (1986)
Chan, K.S.: On the validity of the method of surrogate data. Fields Inst. Comm. **11**, 77–97 (1997)
Choi, E., Hall, P.: Bootstrap confidence regions computed from auto-regressions of arbitrary order. J. Roy. Stat. Soc. B **62**, 461–477 (2000)
Dahlhaus, R., Janas, D.: A frequency domain bootstrap for ratio statistics in time series. Ann. Stat. **24**, 1934–1963 (1996)
Davison, A.C., Hinkley, D.V.: Bootstrap Methods and their Applications, Cambridge University Press, Cambridge (1997)
Davison, A.C., Hinkley, D.V., Young, G.V.: Recent developments in bootstrap methodology. Stat. Sci. **18**, 141–157 (2003)
Efron, B.: Bootstrap methods: Another look at jackknife. Ann. Stat. **7**, 1–26 (1979)
Efron, B.: The Jackknife, the Bootstrap, and Other Resampling Plans. SIAM, Philadelphia (1982)
Efron, B.: Second thoughts on the bootstrap. Stat. Sci. **18**, 135–140 (2003)
Efron, B., Tibshirani, R.J.: An Introduction to the Bootstrap. Chapman and Hall, London (1993)
Franke, J., Härdle, W.: On bootstrapping kernel spectral estimates. Ann. Stat. **20**, 121–145 (1992)
Franke, J., Kreiss, J.-P.: Bootstrapping ARMA-models. J. Time Series Anal. **13**, 297–317 (1992)
Franke, J., Kreiss, J.-P., Mammen E.: Bootstrap of kernel smoothing in nonlinear time series. Bernoulli **8**, 1–37 (2002a)
Franke, J., Kreiss, J.-P., Mammen, E., Neumann, M.H.: Properties of the nonparametric autoregressive bootstrap. J. Time Series Anal. **23**, 555–585 (2002b)
Gine, E.: Lectures on some aspects of the bootstrap. In: Bernard, P.: (eds.) Lectures on Probability Theory and Statistics., pp. 37–151. Springer Lecture Notes Math., Berlin 1665 (1997)
Härdle, W., Mammen, E.: Bootstrap methods for nonparametric regression. In: Roussas, G. (eds.) Nonparametric Functional estimation and Related Topics, pp. 111–124. Kluwer, Dordrecht (1991)
Härdle, W., Mammen, E.: Testing parametric versus nonparametric regression. Ann. Stat. **21**, 1926–1947 (1993)
Härdle, W., Horowitz, J.L., Kreiss, J.-P.: Bootstrap methods for time series. Int. Stat. Rev. **71**, 435–459 (2003)
Hall, P.: Resampling a coverage process. Stoch. Proc. Appl. **19**, 259–269 (1985)
Hall, P.: The Bootstrap and Edgeworth Expansions. Springer, New York (1992)
Hall, P.: A short prehistory of the bootstrap. Stat. Sci. **18**, 158–167 (2003)
Hall, P., Horowitz, J.L., Jing, B.-Y.: On blocking rules for the bootstrap with dependent data. Biometrika **82**, 561–574 (1995)
Hall, P., Jing, B.-Y.: On sample reuse methods for dependent data. J. Roy. Stat. Soc. B **58**, 727–737 (1996)
Horowitz, J.L.: Bootstrap Methods in Econometrics: Theory and Numerical Performance. In: Kreps, D.M., Wallis, K.F. (eds.) Advances in Economics and Econometrics: Theory and Applications. pp. 188–222. Cambridge University Press, Cambridge (1997)
Horowitz, J.L.: The Bootstrap. In Heckman, J.J., Leamer, E.E.: (eds), Handbook of Econometrics. vol. 5, Chap. 52, pp. 3159–3228. Elsevier Science B.V (2001)
Horowitz, J.L.: The bootstrap in econometrics. Stat. Sci. **18**, 211–218 (2003a)
Horowitz, J.L.: Bootstrap methods for markov processes. Econometrica **71**, 1049–1082 (2003b)
Inoue, A., Kilian, L.: The continuity of the limit distribution in the parameter of interest is not essential for the validity of the bootstrap. Econometric Theor. **6**, 944–961 (2003)
Janssen, A., Pauls, T.: How do bootstrap and permutation tests work? Ann. Stat. **31**, 768–806 (2003)
Kreiss, J.-P.: Asymptotic Inference for a Class of Stochastic Processes, Habilitationsschrift. Faculty of Mathematics, University of Hamburg, Germany (1988)
Kreiss, J.-P.: Bootstrap procedures for AR(∞) processes. In: Jöckel, K.-H., Rothe, G., Sendler, W. (eds.) Bootstrapping and Related Techniques, Lecture Notes in Economics and Mathematical Systems 376, pp. 107–113. Springer, Berlin, New York (1992)

Kreiss, J.-P., Paparoditis, E.: Autoregressive aided periodogram bootstrap for time series. Ann. Stat. **31**, 1923–1955 (2003)
Kreiss, J.-P., Paparoditis, E.: Bootstrap methods for dependent data: A review. Journal of the Korean Statistical Society. In print (2011)
Künsch, H.R.: The jackknife and the bootstrap for general stationary observations. Ann. Stat. **17**, 1217–1241 (1989)
Lahiri, S.N.: Second order optimality of stationary bootstrap. Stat. Probab. Lett. **11**, 335–341 (1999a)
Lahiri, S.N.: Theoretical comparison of block bootstrap methods. Ann. Stat. **27**, 386–404 (1999b)
Lahiri, P.: On the impact of bootstrap in survey sampling and small-area estimation. Stat. Sci. **18**, 199–210 (2003a)
Lahiri, S.N.: Resampling Methods for Dependent data. Springer, New York (2003b)
Lehmann, E.L.: Testing Statistical Hypotheses. Springer, New York (1986)
Lele, S.R.: Impact of bootstrap on the estimating functions. Stat. Sci. **18**, 185–190 (2003)
Liu, R.Y.: Bootstrap procedures under some non i.i.d. models. Ann. Stat. **16**, 1696–1708 (1988)
Liu, R.Y., Singh, K.: Efficiency and robustness in resampling. Ann. Stat. **20**, 370–384 (1992a)
Liu, R.Y., Singh, K.: Moving blocks jackknife and bootstrap capture weak dependence. In: Lepage, R., Billard, L.: (eds.) Exploring the Limits of the Bootstrap, pp. 225–248. Wiley, New York (1992b)
Mammen, E.: Asymptotics with increasing dimension for robust regression with applications to the bootstrap. Ann. Stat. **17**, 382–400 (1989)
Mammen, E.: Bootstrap, wild bootstrap, and asymptotic normality. Probab. Theor. Relat. Field. **93**, 439–455 (1992a)
Mammen, E.: When Does Bootstrap Work? Asymptotic Results and Simulations. Springer Lecture Notes in Statistics 77, Springer, Heidelberg, Berlin (1992b)
Mammen, E.: Bootstrap and wild bootstrap for high-dimensional linear models. Ann. Stat. **21**, 2555–285 (1993)
Mammen, E.: Resampling methods for nonparametric regression. In: Schimek, M.G. (eds.) Smoothing and Regression: Approaches, Computation and Application. Wiley, New York (2000)
Mammen, E., Nandi, S.: Change of the nature of a test when surrogate data are applied. Phys. Rev. **E 70**, 016121 (2004)
Mikosch, T., Starica, C.: Nonstationarities in Financial Time Series, the Long-Range Dependence, and the IGARCH Effects. The Review of Economics and Statistics **86**, 378–390 (2004)
Neumann, M., Kreiss, J.-P.: Regression-type inference in nonparametric autoregression. Ann. Stat. **26**, 1570–1613 (1988)
Paparoditis, E.: Bootstrapping autoregressive and moving average parameter estimates of infinite order vector autoregressive processes. J. Multivariate Anal. **57**, 277–296 (1996)
Paparodtis, E., Politis, D.N.: The local bootstrap for kernel estimators under general dependence conditions. Ann. Inst. Stat. Math. **52**, 139–159 (2000)
Paparoditis, E., Politis, D.N.: Tapered block bootstrap. Biometrika **88**, 1105–1119 (2001a)
Paparoditis, E., Politis, D.N.: A Markovian local resampling scheme for nonparametric estimators in time series analysis. Econometric Theor. **17**, 540–566 (2001b)
Paparoditis, E., Politis D.N.: The tapered block bootstrap for general statistics from stationary sequences. Econometric. J. **5**, 131–148 (2002a)
Paparoditis, E., Politis, D.N.: The local bootstrap for Markov processes. J. Stat. Plann. Infern. **108**, 301–328 (2002b)
Paparoditis, E., Politis, D.N.: Resampling and subsampling for financial time series. In: Handbook of Financial Time Series (Andersen, T., Davis, R., Kreiss, J.-P., and Mikosch, T. (eds.)). Springer, New York, 983–999 (2009)
Park, J.Y.: An invariance principle for sieve bootstrap in time series. Econometric Theor. **18**, 469–490 (2002)
Politis, D.N.: The impact of bootstrap methods on time series analysis. Stat. Sci. **18**, 219–230 (2003)

Politis, D.N., Romano, J.P.: The stationary bootstrap. J. Amer. Stat. Assoc. **89**, 1303–1313 (1994)
Politis, D.N., Romano, J.P., Wolf, M.: Subsampling Springer, New York (1999)
Rajarshi, M.B.: Bootstrap in Markovsequences based on estimates of transition density. Ann. Inst. Staitst. Math. **42**, 253–268 (1990)
Shao, J.: Resampling methods in sample surveys (with discussions). Statistics **27**, 203–254 (1996)
Shao, J.: Impact of the bootstrap on sample surveys. Stat. Sci. **18**, 191–198 (2003)
Shao, J., Tu, T.: The Jackknife and Bootstrap. Springer, New York (1995)
Theiler, J., Eubank, S., Longtin, A., Galdrikan, B., Farmer, J.D.: Testing for nonlinearity in time series: The method of surrogate data. Physica D **58**, 77–94 (1992)
Wu, C.F.J.: Jackknife, bootstrap, and other resampling methods in regression analysis (with discussion). Ann. Stat. **14**, 1261–1295 (1986)

Chapter 18
Design and Analysis of Monte Carlo Experiments

Jack P.C. Kleijnen

18.1 Introduction

By definition, computer simulation or *Monte Carlo* models are not solved by mathematical analysis (such as differential calculus), but are used for numerical experimentation. The goal of these experiments is to answer questions about the real world; i.e., the experimenters may use their models to answer *what if* questions – this is also called *sensitivity analysis*. Sensitivity analysis – guided by the statistical theory on *design of experiments* (DOE) – is the focus of this chapter. This sensitivity analysis may serve validation of the model, optimization of the real system, and risk or uncertainty analysis to find robust solutions for real problems; see Kleijnen (2008). Note that optimization is also discussed at length in Chap. II.6 by Spall.

Though I assume that the reader is familiar with basic Monte Carlo methods, I shall summarize a simple Monte Carlo example (based on the well-known Student t statistic) in Sect. 18.2. This example will also illustrate bootstrap and variance-reduction techniques.

Furthermore, in this chapter I shall summarize classic DOE, and extend it to newer methods (for example, DOE for interpolation using Kriging; Kriging is named after the South-African mining engineer named Krige).

Traditionally, 'the shoemaker's children go barefoot'; i.e., users of computational statistics ignore statistical issues of their Monte Carlo results. Nevertheless, they should address the following two types of issues:

1. *Tactical* issues: number of (macro)replicates, variance-reduction techniques.
2. *Strategic* issues: situations to be simulated, and the sensitivity analysis of the resulting data.

J.P.C. Kleijnen (✉)
Department of Information Management/Center for Economic Research (CentER), Tilburg University, Tilburg, The Netherlands
e-mail: Kleijnen@UvT.NL

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_18,

Both types of issues will be addressed in this chapter.

As Monte Carlo methods are applied in many disciplines, the *terminologies* vary. DOE speaks of 'factors' with 'levels', whereas simulation analysts may speak of 'inputs'or 'parameters' with 'values'. DOE talks about 'design points' or 'runs', whereas simulationists may talk about 'situations', 'cases', or 'scenarios'.

Originally, classic DOE was developed for experiments with real, non-simulated systems in agriculture, in the 1930s. Since the 1950s, DOE has also been developed for experiments in engineering, psychology, etc. This classic DOE includes fractional factorial designs, as we shall see. In those real systems it is impractical to experiment with 'many' factors: ten factors seems a maximum. Moreover, it is then hard to experiment with factors that have more than 'a few' values: five values per factor seems a maximum. Finally, these experiments are run in 'one shot'(for example, in one growing season) and not sequentially. In Monte Carlo experiments, however, these limitations do not hold! A recent textbook on classic DOE for simulation is Kleijnen (2008), which covers a wider area and gives more details than this review does.

Continuing the discussion on terminology, I speak of the *Monte Carlo* method whenever (pseudo)random numbers are used; for example, I shall apply the Monte Carlo method to estimate the behavior of the t statistic in case of non-normality, in Sect. 18.2 (the Monte Carlo method may also be used to estimate multiple integrals, which is a deterministic problem that is outside the scope of this handbook). *Random numbers* (say) r are identically and independently distributed (IID) on the interval from zero to one: $r \in U(0, 1)$. *Pseudorandom numbers* (PRN) are generated through a computer, and are assumed to behave like truly random numbers when applying the Monte Carlo method.

I use the term *simulation* whenever the analysts compute the output of a dynamic model; i.e., the analysts do not use calculus to find the solution of a set of differential or difference equations. The dynamic model may be either stochastic or deterministic. Stochastic simulation uses the Monte Carlo method; it is often applied to solve waiting (queuing) problems in telecommunications and logistics. Deterministic simulation is often applied in computer-aided engineering (CAE), computer-aided design (CAD), and computer-aided manufacturing (CAM).

I use the term *metamodel* for models that approximate – or model – the input/output (I/O) behavior of the underlying simulation model; for example, a low-order polynomial regression model is a popular metamodel (as we shall see). Metamodels are used – consciously or not – to design and analyze experiments with simulation models. In the simulation literature, metamodels are also called emulators, response surfaces, surrogates, etc.

The remainder of this chapter is organized as follows. Section 18.2 presents a simple Monte Carlo experiment with Student's t statistic, including bootstrapping and variance-reduction techniques. Section 18.3 discusses the black-box approach to simulation experiments, and corresponding metamodels – especially, polynomial and Kriging models. Section 18.4 starts with simple regression (meta)models with a single factor, proceeds with designs for multiple factors including designs for first-order and second-order polynomial models, and concludes with screening

designs for 'many' (for example, a few hundred) factors. Section 18.5 introduces Kriging, which often uses space-filling designs, such as Latin hypercube sampling (LHS); Kriging has only recently been applied in random simulation, but it has already established a track record in deterministic simulation and spatial statistics. Section 18.6 gives conclusions and discusses topics for further research.

18.2 Simulation Techniques in Computational Statistics

Consider the well-known definition of the t statistic with $n-1$ degrees of freedom:

$$t_{n-1} = \frac{\bar{x} - \mu}{s_x / \sqrt{n}} \text{with} \bar{x} = \frac{\sum_{i=1}^{n} x_i}{n} \quad \text{and} \quad s_x^2 = \frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1} \tag{18.1}$$

where the x_i $(1 = 1, \ldots, n)$ are normally (Gaussian), independently, and identically distributed (NIID) with mean μ and variance σ^2:

$$x_i \in NIID(\mu, \sigma)(i = 1, \ldots, n) \tag{18.2}$$

A century ago, Gossett used a (manual) Monte Carlo experiment, before he analytically derived the density function of the statistic defined in (18.1) and (18.2); he published his results under the pseudonym 'Student'. So in this experiment, he sampled n values x_i from an urn for which (18.2) was an adequate model, and he computed the corresponding value for the statistic defined by (18.1). This experiment he repeated (say) m times, so that he could compute the estimated density function (EDF) – also called the empirical cumulative distribution function (ECDF) – of the statistic.

Let us imitate Student's experiment through the simulation experiment described in the following procedure with six steps.

1. Read the simulation inputs: μ (mean), σ^2 (variance), n(sample size), m(number of macro-replicates, used in step 4).
2. Take $n$samples $x_i \in \text{NIID}(\mu, \sigma)$ (see (18.2) and Chap. II.2 by L'Ecuyer).
3. Compute the statistic t_{n-1} defined in (18.1).
4. Repeat steps 2 and 3 m times.
5. Sort – in increasing order – the m values of t_{n-1} resulting from step 4.
6. Compute the EDF from the results in step 5.

To *verify* this simulation program, we may compare the result (namely the EDF in step 6) with the results that are tabulated for Student's density function; for example, does our EDF give an estimated 90% quantile that does not differ significantly from the exact value, $t_{n-1;0.90}$. Next we may proceed to the following more interesting experiment.

We may drop the classic assumption formulated in (18.2), and experiment with *non-normal* distributions. It is easy to sample from such distributions (see again Chap. II.2). However, we are now confronted with several so-called *strategic* choices (also see step 1 above): Which type of distribution should we select (lognormal, exponential, etc.); which parameter values for that distribution type (mean and variance for the lognormal, etc.), which sample size n (for a 'large' n, the t distribution is known to be a good approximation for our EDF)?

Besides these strategic issues, we must face some *tactical* issues: Which number of macro-replicates m gives a good EDF; can we use special *variance reducing techniques* (VRTs) – such as common random numbers and importance sampling – to reduce the variability of the EDF? I explain these techniques briefly, as follows.

Common random numbers (CRN) mean that the analysts use the same (pseudo)random numbers when estimating the effects of different strategic choices. For example, CRN may be applied when comparing the estimated 90% quantiles $\hat{t}_{n-1;0.90}$ for various sample sizes n and distribution types. Obviously, CRN reduces the variance of estimated differences in performance, provided CRN creates *positive* correlations between the estimators $\hat{t}_{n-1;0.90}$ being compared for various sample sizes, etc.

Antithetic variates (AV) mean that the analysts use the complements $(1 - r_i$ with $i = 1, \ldots, n)$ of the PRN $(r_i$ with $i = 1, \ldots, n)$ in two 'companion' macro-replicates (where I assume that each sample of x_i with $i = 1, \ldots, n$ requires one random number; for example, sampling from the exponential distribution with rate λ may use $x_i = -\ln r_i/\lambda$ with $i = 1, \ldots, n)$. AV reduces the variance of the estimator averaged over these two replicates, provided AV creates *negative* correlation between the two estimators resulting from the two replicates; obviously, $1 - r_i$ and r_i have correlation -1, but nonlinear transformations such as $-\ln(1 - r_i)/\lambda$ and $-\ln r_i/\lambda$ do not have such a strong negative correlation.

Importance sampling (IS) is used when the analysts wish to estimate a *rare* event, such as the probability of the Student statistic exceeding the 99.999% quantile, the probability of a 'disaster' occurring in ecology, finance, telecommunications, etc. IS increases that probability (for example, by sampling from a distribution with a fatter tail), and then corrects for this distortion of the input distribution (through the likelihood ratio). IS is not so simple as CRN and AV – but without IS too much computer time may be needed. Publications on rare event simulation may be found in the proceedings of the yearly Winter Simulation Conference (WSC); see http://www.wintersim.org/.

There are many more VRTs. Both CRN and AV are intuitively attractive and easy to implement, but most popular is CRN. The most useful VRT may be IS. In practice, the other VRTs often do not reduce the variance drastically so many users prefer to spend more computer time instead of applying VRTs.

Finally, suppose a very limited set of historical data is given and we must analyze these data while we know that these data do not satisfy the classic assumption formulated in (18.2). Then *bootstrapping* may help, as follows (also remember the six steps above).

1. Read the bootstrap sample size B(B is the usual symbol in bootstrapping, comparable with m, the number of macro-replicates in step 1 above).
2. Take n samples with replacement from the original sample x_i ($i = 1, \ldots, n$); this sampling gives x_i^* (the superscript * is the usual symbol to denote bootstrapped values, which are to be distinguished from the 'original' values).
3. From these bootstrapped values x_i^* compute the bootstrap statistic t_{n-1}^* (see (18.1)).
4. Repeat steps 2 and 3 B times.
5. Sort the B values of t_{n-1}^* that result from step 4.
6. Compute the EDF of t_{n-1}^* from the results in step 5.

So, bootstrapping is just a Monte Carlo experiment – using resampling with replacement of a given data set. (There is also a parametric bootstrap, which comes even closer to our simulation of Gossett's original experiment.) Bootstrapping is further discussed in Efron and Tibshirani (1993), Kleijnen (2008, pp. 80–87) and in Chap. III.2 (by Mammen).

18.3 Black-Box Metamodels of Simulation Models

DOE treats the simulation model as a *black box*; i.e., only the inputs and outputs are observed and analyzed. For example, in the simulation of the t statistic (in Sect. 18.2) the simulation inputs (listed in step 1) are μ (mean), σ^2 (variance), n(sample size), and m(number of macro-replicates); this m is probably a tactical factor that is not of interest to the user. Suppose the user is interested in the 90% quantile of the distribution function of the statistic in case of nonnormality. A *black-box* representation of this example is:

$$t_{n-1;0.90} = t(\mu, \sigma, n, r_0) \tag{18.3}$$

where t(*dot*) denotes the mathematical function implicitly defined by the simulation procedure (outlined in steps 1 through 6 below (18.2), which form the white-box simulation model); μ and σ now denote the parameters of the nonnormal distribution of the input x_i (for example, μ denotes how many exponential distributions with parameter $\sigma = \lambda$ are summed to form an Erlang distribution); r_0 denotes the seed of the (pseudo)random numbers.

One possible *metamodel* of the black-box model in (18.3) is a Taylor-series approximation that is cut off after the first-order effects of the three factors, μ, σ, and n:

$$y = \beta_0 + \beta_1 \mu + \beta_2 \sigma + \beta_3 n + e \tag{18.4}$$

where y is the metamodel predictor of the simulation output $t_{n-1;0.90}$ in (18.3); $\boldsymbol{\beta}^T = (\beta_0, \beta_1, \beta_2, \beta_3)$ denotes the row-vector with the parameters of the metamodel in (18.4), and e is the noise which includes both *lack of fit* of the metamodel and *intrinsic noise* caused by the (pseudo)random numbers with seed r_0.

Instead of the metamodel specified in (18.4), many alternative metamodels may be used. For example, taking the logarithm of the inputs and outputs in (18.4) makes the first-order polynomial approximate relative changes; i.e., the parameters β_1, β_2, and β_3 become so-called elasticity coefficients. There are also more complicated types of metamodels. Examples are Kriging models, neural nets, radial basis functions, splines, support vector regression, and wavelets; see the various chapters in Part III – especially Chaps. III.5 (by Loader), III.7 (Müller), III.8 (Cizek), and III.15 (Laskov and Müller) – and also the many references in Kleijnen (2008, p. 8). I, however, will focus on two types that have established a track record in simulation:

- linear regression models; see Sect. 18.4;
- Kriging; see Sect. 18.5.

To estimate the parameters of whatever black-box metamodel, the analysts must *experiment* with the simulation model; i.e., they must change the inputs (or factors) of the simulation, run the simulation, and analyze the resulting I/O data. This experimentation is the topic of the next sections.

18.4 Designs for Linear Regression Metamodels

18.4.1 Linear Regression Metamodels with a Single Factor

I start the discussion of DOE with the simplest metamodel; namely, a first-order polynomial with a single factor. An example is the 'Student' simulation in Sect. 18.2, for which I now assume that the response (output) of interest is the type-II error probability (or its complement, the power) so y in (18.4) now denotes this probability predicted through the regression metamodel. I further assume a single factor (say) $x = \sigma/n$ ('relative' variability; i.e., absolute variability corrected for sample size); see (18.4). Elementary mathematics proves that to fit a straight line it suffices to have two I/O observations; see 'local area 1' in Fig. 18.1. It is simple to prove that the 'best' estimators of the regression parameters result if those two values are as far apart as 'possible'. In practice, the analysts do not know over which *experimental area* a first-order polynomial is a 'valid' model. This validity depends on the goals of the simulation study; see Kleijnen and Sargent (2000).

So the analysts may start with a *local* area, and simulate the two (locally) extreme input values. Let us denote these two extreme values of the 'coded' variable x by -1 and $+1$, which implies the following *standardization* of the original variable z:

$$x = \frac{z - \bar{z}}{(z_{\max} - z_{\min})/2} \tag{18.5}$$

where $\bar{z}$ denotes the average value of the original variable (in the example, the relative variability $z = \sigma/n$) in the (local) experiment.

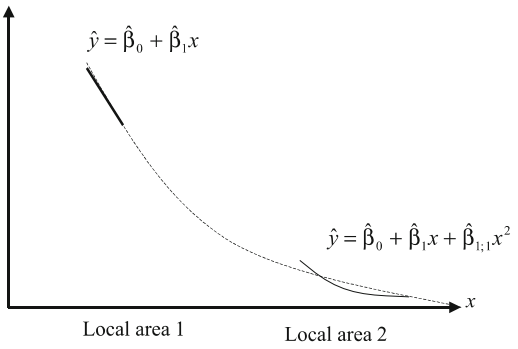


Fig. 18.1 Two simple polynomial regression models with predictor $\hat{y}$ for the output of a simulation with a single factor x

The Taylor series implies that – as the experimental area gets bigger (also see 'local area 2' in Fig. 18.1) – a better metamodel may be the second-order polynomial

$$y = \beta_0 + \beta_1 x + \beta_2 x^2 + e. \tag{18.6}$$

Obviously, estimation of the three parameters in (18.6) requires at least the simulation of three input values. Indeed, DOE provides designs with three values per factor; for example, 3^k designs for k factors. However, most publications on the application of DOE in simulation focus on *central composite designs* (CCD), which have five values per factor; see Sect. 18.4.2 below.

I emphasize that the second-order polynomial in (18.6) is nonlinear in x (the regression variable), but *linear* in ¸ (the regression parameters or factor effects to be estimated). Consequently, such a polynomial is a type of *linear regression* model (also see Chap. III.8).

When the experimental area covers the *whole* area in which the simulation model is valid (see again Fig. 18.1), then other *global* metamodels become relevant. For example, Kleijnen and Van Beers (2005) find that *Kriging* models (to be discussed in Sect. 18.5) outperform second-order polynomials.

Note that Zeigler, Praehofer, and Kim (2000) call the experimental area the 'experimental frame'. I call it the domain of admissible scenarios, given the goals of the simulation study.

I conclude that *lessons* learned from the simple example in Fig. 18.1, are:

1. The analysts should decide whether they want to experiment*locally* or*globally*.
2. Given that decision, they should select a specific *metamodel type* (low-order polynomial, Kriging, spline, etc.); also see Chaps. III.5, III.7, and III.8.

18.4.2 Linear Regression Metamodels with Multiple Factors

Let us now consider a regression model with k factors; for example, $k = 2$. A popular but inferior design *changes one factor at a time*. For $k = 2$ such a design

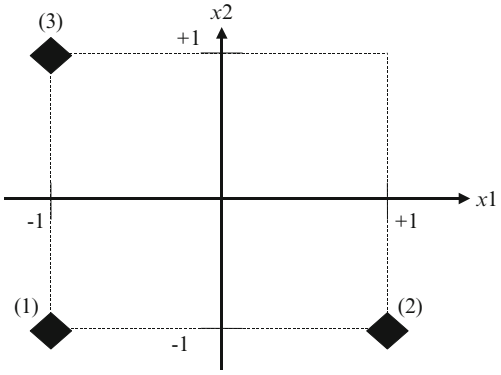


Fig. 18.2 One-factor-at-a-time design for two factors x_1 and $x_2 x_2$

Table 18.1 A one-factor-at-a-time design for two factors, and possible regression variables

Scenario	x_0	x_1	x_2	$x_1 x_2$
1	1	−1	−1	1
2	1	1	−1	−1
3	1	−1	1	−1

is shown in Fig. 18.2 and Table 18.1; this table displays the factor values – in the various factor combinations – in the columns denoted by x_1 and x_2; the 'dummy' column x_0 corresponds with the polynomial intercept $\hat{\beta}_0$ in (18.4). In this design the analysts usually start with the 'base' scenario, denoted by the factor combination $(-1, -1)$; see scenario 1 in the table. Next they simulate the two scenarios $(1, -1)$ and $(-1, 1)$; see the scenarios 2 and 3 in the table.

The reader may find it intuitively clear that changing a single factor at a time does not enable the estimation of the *interaction* between the two factors. Applying linear algebra we can prove that the three-by-four matrix $\boldsymbol{X}$ implied by Table 18.1 must have a column for the interaction $x_1 x_2$ that is linearly dependent on the other three columns. If the vector of simulation outputs is denoted by $\boldsymbol{w}$, then the *ordinary least squares* (OLS) estimator

$$\hat{\boldsymbol{\beta}} = (\mathbf{X}^T \mathbf{X})^{-1} \mathbf{X}^T \mathbf{w} \tag{18.7}$$

does not exist because the inverse in (18.7) does not exist. In mathematical statistics, we say that the estimated effects are *confounded* with or *biased* by each other. One-factor-at-a-time designs are further discussed by Kleijnen (2008, pp. 32–33).

Note that in practice, analysts often study each factor in their one-at-a-time design at *three levels* (which may be denoted by the standardized values $-1, 0, +1$). However, two levels suffice to estimate the parameters of a first-order polynomial (see again Sect. 18.4.1).

To enable the estimation of *interactions*, the analysts must change factors *simultaneously*; for example, the analysts also observe the factor combination (1, 1) besides the three combinations in Table 18.1. It is easy to check that the resulting 2^2

design implies a matrix of regression variables X that is *orthogonal*:

$$\mathbf{X}^T\mathbf{X} = n\mathbf{I} \tag{18.8}$$

where n denotes the number of scenarios simulated; $n = 4$ in the 2^2 design. Hence the OLS estimator (18.7) simplifies to $\hat{\boldsymbol{\beta}} = \mathbf{X}^T\mathbf{w}/4$.

In the general case of k factors, a 2^k design enables the unbiased estimation of the intercept (grand mean) β_0. all k first-order effects (main effects) $\beta_j (j = 1, \ldots, k)$, all the interactions between pairs of factors $\beta_{j;j'}$ $(j < j' = 2, \ldots, k)$, among triplets, etc. The interpretation of interactions among three or more factors is rather difficult, so they are usually assumed to be negligible. This assumption implies that a 2^k design requires the simulation of relatively many (namely, 2^k) factor combinations, as the number of effects to be estimated (say) q in only $1 + k + k(k-1)/2$ in case of a polynomial metamodel with an intercept, first-order effects, and two-factor interactions. In such situations, the analysts need to simulate only a fraction of the 2^k design – as we shall see next.

Let us consider a situation in which the number of factors k increases from two to three. Then Fig. 18.2 becomes Fig. 18.3 (the asterisks will be explained below) and Table 18.1 becomes Table 18.2. The latter table shows that each of the three factors (see the columns with the headings 1, 2, 3) has the two standardized values -1 and 1; to simplify the notation, the table shows only the signs of the factor values so $-$ means -1 and $+$ means $+1$. All 2^3 combinations of these two values or 'scenarios' are simulated (see the first column). The table further shows possible regression variables, using the symbols '0' through '1.2.3' to denote the subscripts of the regression variables x_0 (the dummy variable always equals 1) through $x_1x_2x_3$ (third-order interaction). Further, I point out that each column is *balanced*; i.e., each column has four plusses and four minuses – except for the dummy column.

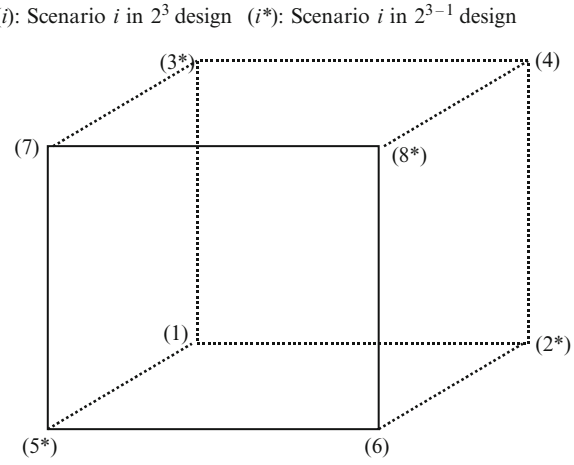


Fig. 18.3 The 2^3 design

The 2^3 design enables the unbiased OLS estimation of all eight parameters of the following regression model, which is a third-order polynomial that is *incomplete*; i.e., some parameters are assumed zero:

$$y = \beta_0 + \sum_{j=1}^{3} \beta_j x_j + \sum_{j=1}^{2} \sum_{j'>j}^{3} \beta_{j;j'} x_j x_{j'} + \beta_{1;2;3} x_1 x_2 x_3 + e. \tag{18.9}$$

Indeed, the 2^3 design implies a matrix of regression variables X that is orthogonal; see (18.8).

The metamodel (18.9) includes the third-order effect $\beta_{1;2;3}$. Assuming that this effect is negligible, the analysts may simulate fewer scenarios than specified by the 2^3 design. Indeed a 2^{3-1} fractional factorial design suffices; a possible 2^{3-1} design is shown in Table 18.2 deleting the four rows (scenarios) that have a minus sign in the 1.2.3 column (i.e., delete the rows 1, 4, 6, 7). In other words, the analysts simulate only a *fraction*; namely, 2^{-1} of the 2^3 full-factorial design. This fraction corresponds with the points denoted by the symbol * in Fig. 18.3 (so each scenario corresponds with a vertex that cannot be reached via a single edge of the cube).

In this 2^{3-1} design two columns are identical; namely, the 1.2.3 column (with four plusses) and the dummy column. Hence, the corresponding two effects are *confounded* – but the high-order interaction $\beta_{1;2;3}$ is assumed zero, so this confounding can be ignored!

Sometimes, (18.9) may be replaced by a *first-order polynomial*; for example, in the (sequential) optimization of black-box simulation models the analysts may use a first-order polynomial to estimate the local gradient; see Angün et al. (2009). Then it suffices to take a 2^{k-p} design with the biggest p value that makes the following condition hold: $2^{k-p} > k$. An example is $k = 7$ and $p = 4$ so only eight scenarios are simulated; see Table 18.3, in which the first three factors (labeled 1, 2, and 3) form a full-factorial 2^3design, and the symbol '4=1.2' means that the values for factor 4 are selected by multiplying the elements of the columns for the factors 1 and 2, etc. Note that the design is still balanced and orthogonal.

Because a 2^{k-p} design is *orthogonal*, its estimators of the regression parameters have smaller variances than one-factor-at-a-time designs have; see Kleijnen (2008,

Table 18.2 The 2^3 design and possible regression variables

Scenario	0	1	2	3	1.2	1.3	2.3	1.2.3
1	+	−	−	−	+	+	+	−
2	+	+	−	−	−	−	+	+
3	+	−	+	−	−	+	−	+
4	+	+	+	−	+	−	−	−
5	+	−	−	+	+	−	−	+
6	+	+	−	+	−	+	−	−
7	+	−	+	+	−	−	+	−
8	+	+	+	+	+	+	+	+

Table 18.3 A 2^{7-4} design

Scenario	1	2	3	4 = 1.2	5 = 1.3	6 = 2.3	7 = 1.2.3
1	−	−	−	+	+	+	−
2	+	−	−	−	−	+	+
3	−	+	−	−	+	−	+
4	+	+	−	+	−	−	−
5	−	−	+	+	−	−	+
6	+	−	+	−	+	−	−
7	−	+	+	−	−	+	−
8	+	+	+	+	+	+	+

p. 32). The actual variances are found on the main diagonal of the *covariance matrix* of the (linear) OLS estimator given by (18.7):

$$\mathbf{cov}(\hat{\boldsymbol{\beta}}) = [(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T]\mathbf{cov}(\mathbf{w})[(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T]^T. \tag{18.10}$$

If we assume *white noise* and *no CRN*, then $\mathbf{cov}(\mathbf{w})$ in (18.10) reduces to

$$\mathbf{cov}(\mathbf{w}) = \sigma^2\mathbf{I}, \tag{18.11}$$

so (18.10) reduces to

$$\mathbf{cov}(\hat{\boldsymbol{\beta}}) = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}. \tag{18.12}$$

I shall return to these assumptions behind (18.11), at the end of this section.

How to select scenarios in 2^{k-p} designs is detailed in many DOE textbooks, including Kleijnen (2008).

Fractional factorial designs of the 2^{k-p} type – with a p value so high that the design still enables the estimation of first-order polynomial regression models – are a subset of *Plackett–Burman designs*. The latter designs consists of $k + 1$ combinations with $k + 1$ rounded upwards to a multiple of four; for example, Table 18.4 gives such a design for $k = 11$. For $7 < k < 11$ we can still use Table 18.4, simply deleting the last $11 - k$ columns. Plackett–Burman designs are tabulated in many DOE textbooks, including Myers et al. (2009). Note that designs for first-order polynomial regression models are called *resolution III* designs.

Resolution IV designs enable unbiased estimators of first-order effects – even if two-factor interactions are important. These designs require double the number of scenarios required by resolution III designs; i.e., after simulating the scenarios of the resolution III design, the analysts simulate the *mirror scenarios* which multiply by -1 the factor values in the original scenarios. For example, the resolution IV design for $k = 11$ factors based on the resolution III design in Table 18.4 has scenario 13 that is the mirror image of scenario 1 so it combines the factor values -+-+++---+-; scenario 24 is the mirror image of scenario 12 so it has all 11 factors at their +value.

Resolution V designs enable unbiased estimators of first-order effects plus all two-factor interactions. This class includes certain 2^{k-p} designs with small enough

Table 18.4 The Placket–Burman design for 11 factors

Scenario	1	2	3	4	5	6	7	8	9	10	11
1	+	−	+	−	−	−	+	+	+	−	+
2	+	+	−	+	−	−	−	+	+	+	−
3	−	+	+	−	+	−	−	−	+	+	+
4	+	+	+	−	+	+	−	−	−	+	+
5	+	−	+	+	−	−	−	−	−	−	+
6	+	+	−	+	+	+	+	+	−	−	−
7	−	+	+	+	−	+	+	−	+	−	−
8	−	−	+	+	+	−	+	+	−	+	−
9	−	−	−	+	+	+	−	+	+	−	+
10	+	−	−	−	+	+	+	−	+	+	−
11	−	+	−	−	−	+	+	+	−	+	+
12	−	−	−	−	−	−	−	−	−	−	−

p values. For example, the resolution V design for $k = 11$ factors has $2^{11-4} = 128$ combinations where the first seven factors form a 2^7 design and the remaining factor values are defined by the following relations (called 'generators'): $8 = 1.2.3.7$, $9 = 2.3.4.5$, $10 = 1.3.4.6$ and $11 = 1.\ 2.3.4.5.6.7$. These designs often require rather many scenarios to be simulated; for example, the number of effects to be estimated for $k = 11$ factors is $1 + 11 + 11 \times 10/2 = 67$ whereas the number of scenarios simulated is 128. Fortunately, there are also *saturated* designs; i.e., designs with the minimum number of scenarios that still allow unbiased estimators of the regression parameters. Saturated designs are attractive for *expensive* simulations; i.e., simulations that require relatively much computer time per scenario. Saturated resolution V designs are discussed by Kleijnen (2008, pp. 48–49).

Central composite designs (CCD) enable the estimation of second-order polynomials. These designs augment resolution V designs with the base scenario (corresponding with the standardized value zero) and $2k$ 'axial' combinations that change each factor one at a time in the base scenario. Saturated variants (smaller than CCD) are discussed in Kleijnen (2008, p. 51).

The main conclusion is that *incomplete designs* for *low-order polynomial regression* are plentiful in both the classic DOE literature and the simulation literature.

Finally, let us return to the assumptions that gave $\mathbf{cov}(\hat{\boldsymbol{\beta}})$ in (18.12). I claim that in practice these assumptions do not hold:

1. The variances of the simulation output w change as the input $\boldsymbol{x}$ changes so the assumed common variance σ^2 in (18.11) does not hold: *variance heterogeneity* does hold. (Well-known examples are Monte Carlo studies of the type-I and type-II errors, which are binomial variables so the estimated variances are $w(1–w)/m$; also see Sect. 18.2)
2. Often the analysts use *common random numbers* (see CRN in Sect. 18.2), so the assumed diagonality of the matrix in (18.11) does not hold.

Several solutions for these two problems are discussed by Kleijnen (2008, pp. 94–97). Here I present the simplest solution, assuming that each scenario is

simulated $m > 1$ times. Each replicate gives the I/O combination $(\boldsymbol{X}, w_r)$ with $r = 1, \ldots, m$ (all m replicates use the same input combination so the matrix of regression variables $\boldsymbol{X}$ does not change with r). From these I/O data for replicate r we compute the OLS estimate (say) $\hat{\boldsymbol{\beta}}_r$. From these m effect estimates we compute the usual Student t statistic and confidence interval for the true $\boldsymbol{\beta}$; i.e., in (18.1) we replace n by m. and x_i by $\hat{\beta}_{j;r}$ with $j = 1, \ldots, q$. In this way we can test statistically whether some factors have zero effects.

A significant factor may be unimportant, practically speaking. If the factors are scaled between -1 and $+1$, then the estimated effects quantify the *order of importance*. For example, in a first-order polynomial metamodel the factor estimated to be the most important factor is the one with the highest absolute value for its estimated effect.

18.4.3 Simulations with Very Many Factors

Most practical, non-academic simulation models have many factors; for example, Kleijnen (2010) discusses a supply-chain simulation model with nearly 100 factors. For this model, even a Plackett-Burman design would require the simulation of 102 scenarios. Because each scenario needs to be replicated several times (to quantify the noise), the total computer time may then be prohibitive. Therefore, many analysts keep many factors fixed (at the base values), and experiment with only a few remaining factors. For example, Horne and Leonardi (2001) present a military simulation that was run millions of times for only a few scenarios that change a few factors only.

However, there are designs that require fewer than k scenarios – called *supersaturated designs*; various types are referenced by Kleijnen (2010) and Kleijnen (2008, p. 159). Some of these designs *aggregate* the k individual factors into groups of factors. It may then happen that the effects of individual factors cancel out, so the analysts would erroneously conclude that all factors within that group are unimportant. The solution is to define the -1 and $+1$ levels of the individual factors (see (18.5)) such that all first-order effects β_j $(j = 1, \ldots, k)$ are *non-negative*. My experience is that in practice the users do know the direction of the first-order effects of individual factors; otherwise they can keep the few factors with unknown signs out of the group-screening.

The most efficient group screening is *sequential bifurcation*. This design type is so efficient because it proceeds *sequentially*, as follows. It starts with only two scenarios; namely, one scenario with all individual factors at -1, and a second scenario with all factors at $+1$. Comparing the outputs of these two extreme scenarios requires only two replications because the aggregated effect of the group factor is huge compared with the intrinsic noise (caused by the pseudorandom numbers). The next step splits or *bifurcates* the factors into two groups; there are several heuristic rules to decide on how to assign factors to groups. Comparing the outputs of the third scenario with the outputs of the preceding two scenarios enables

the estimation of the aggregated effect of the individual factors within a group. Groups – and all its individual factors – are eliminated from further experimentation as soon as the group effect is statistically unimportant. Obviously, the groups get smaller as the analysts proceed sequentially. The analysts stop, once the first-order effects β_j of all the important individual factors are estimated. For example, in the supply-chain simulation, only 11 of the 92 factors are estimated to be important; see Kleijnen (2010) or Kleijnen (2008, p. 167–169).

18.5 Kriging

Let us return to the example in Fig. 18.1. If the analysts are interested in the I/O behavior within 'local area 1', then a first-order polynomial may be adequate. Maybe, a second-order polynomial is required to get a valid approximation in 'local area 2', which is larger and shows non-linear behavior of the I/O function. However, Kleijnen and Van Beers (2005) present an example illustrating that a second-order polynomial may give poor predictions compared with Kriging.

Kriging has been often applied in deterministic simulation models. Such simulations are used for the development of airplanes, automobiles, computer chips, computer monitors, etc.; see Sacks et al. (1989)'s pioneering article, and the update by Kleijnen (2008, pp. 3, 123). For Monte Carlo experiments, I do not know any applications yet. First, I explain the basics of Kriging; then the DOE aspects of Kriging.

18.5.1 Kriging Basics

Kriging is an *interpolation* method that predicts unknown values of a random process; see the classic textbook on Kriging in spatial statistics, Cressie (1993). More precisely, a Kriging prediction is a weighted linear combination of all output values already observed (simulated). These weights depend on the distances between the input for which the output is to be predicted and the inputs already simulated. Kriging assumes that *the closer the inputs are, the more positively correlated the outputs are*. This assumption is modeled through the correlogram or the related variogram, discussed below.

In deterministic simulation, Kriging has an important advantage over regression analysis: Kriging is an *exact* interpolator; that is, predicted values at 'old' observed input values are exactly equal to the observed output values. In random simulation, however, the observed output values are only estimates of the true values, so exact interpolation loses its intuitive appeal. Santner et al. (2003, pp. 215–249) account for the so-called *nugget* effect or measurement error by adding a white-noise term; their Kriging predictor does not interpolate the n old outputs; also see Forrester et al. (2008, p. 143). Recently, Ankenman et al. (2010) and Yin et al. (2009)

introduced Kriging for random simulation, accounting for variance heterogeneity; their Kriging predictors do not interpolate the n old outputs averaged over the m_i ($i = 1 ::: n$) replicates per input combination. Ankenman et al. also account for CRN so the white-noise assumption does not hold; see Sect. 18.2. Unfortunately, there is no well-documented software implementing these nugget effects, so I focus on DACE, the free Matlab Kriging toolbox for deterministic simulation which is well documented in Lophaven et al. (2002). In random simulation, the analysts may use DACE to interpolate the n averages, like I did in my Kriging publications listed in the references.

The simplest type of Kriging (to which I restrict myself in this chapter) assumes the following *metamodel*:

$$w = \mu + \delta(\mathbf{x}) \tag{18.13}$$

where μ is the mean of the stochastic process w, and $\delta(\mathbf{x})$ is a *Gaussian stationary covariance process* with zero mean; i.e., the covariances of $w(\mathbf{x} + \mathbf{h})$ and $w(\mathbf{x})$ depend only on the distance (or 'lag') between their inputs, $|\mathbf{h}| = |(\mathbf{x} + \mathbf{h}) - (\mathbf{x})|$.

The Kriging *predictor* for the 'new' input $\mathbf{x}_0$ is a weighted linear combination of all the n old output data (which are already simulated):

$$\hat{y}(\mathbf{x}_0) = \sum_{i=1}^{n} \lambda_i \cdot w(\mathbf{x}_i) = \boldsymbol{\lambda}^T \cdot \mathbf{w} \quad \text{with} \quad \sum_{i=1}^{n} \lambda_i = \quad 1. \tag{18.14}$$

To quantify the weights $\boldsymbol{\lambda}$ in (18.14), Kriging derives the *best linear unbiased estimator* (BLUE), which minimizes the mean squared error (MSE) of the predictor $\hat{y}(\mathbf{x}_0)$. Obviously, these weights depend on the covariances mentioned below (18.13). The *optimal* weights can be proven to be

$$\boldsymbol{\lambda}^T = \left(\boldsymbol{\gamma} + \mathbf{1} \frac{\mathbf{1} - \mathbf{1}^T \boldsymbol{\Gamma}^{-1} \boldsymbol{\gamma}}{\mathbf{1}^T \boldsymbol{\Gamma}^{-1} \mathbf{1}} \right)^T \boldsymbol{\Gamma}^{-1} \tag{18.15}$$

where $\boldsymbol{\gamma}$ denotes the vector of the n covariances between the output at the new input $\mathbf{x}_0$ and the outputs at the n old inputs so $\boldsymbol{\gamma} = (\gamma(\mathbf{x}_0 - \mathbf{x}_1), \ldots, \gamma(\mathbf{x}_0 - \mathbf{x}_n))^T$; $\boldsymbol{\Gamma}$ denotes the matrix of the covariances between the outputs at the n old inputs with element (i, j) equal to $\gamma(\mathbf{x}_i - \mathbf{x}_j)$; and $\mathbf{1}$ denotes the vector with n ones. Note that these optimal weights vary with the input value for which output is to be predicted (see $\boldsymbol{\gamma}$), whereas linear regression uses the same estimated parameters $\hat{\boldsymbol{\beta}}$ for all inputs to be predicted. The DACE software uses maximum likelihood estimation (MLE) to estimate the optimal weights, and plugs these weights into (18.14) to compute the Kriging predictor; DACE also gives the estimated gradient corresponding with this prediction.

Note that the correlation function for a k-dimensional input vector is assumed to be the product of k one-dimensional correlation functions; a popular one-dimensional correlation function is the Gaussian one, $\rho_j = \exp(\boldsymbol{\theta}_j h_j^2)$ where $\boldsymbol{\theta}_j$ denotes the importance of input j (the higher $\boldsymbol{\theta}_j$ is, the less effect input j has)

and h_j denotes the Euclidean distance between the values of input j in two input combinations.

18.5.2 Designs for Kriging

The most popular design type for Kriging is *Latin hypercube sampling* (LHS). This design type was invented by McKay et al. (1979) for deterministic simulation models. Those authors did not analyze the I/O data by Kriging (but they did assume I/O functions more complicated than the low-order polynomials in classic DOE). Nevertheless, LHS is much applied in Kriging nowadays, because LHS is a simple technique (it is part of popular spreadsheet add-ons such as @Risk).

LHS offers *flexible* design sizes n (number of scenarios simulated) for any number of simulation inputs k. A simplistic example is shown for $k = 2$ and $n = 4$ in Table 18.5 and Fig. 18.4, which are constructed as follows.

1. The table illustrates that LHS divides each input range into n intervals of equal length, numbered from 1 to n (in the example, we have $n = 4$; see the numbers in the last two columns); i.e., the number of values per input can be much larger than in the designs for low-order polynomials (see again Sect. 18.3).
2. Next, LHS places these integers 1,..., n such that each integer appears exactly once in each row and each column of the design. (This characteristic explains the term "Latin hypercube": it resembles Latin squares in classic DOE.)
3. Within each cell of the design in the table, the exact input value may be sampled uniformly; see Fig. 18.4. (Alternatively, these values may be placed systematically in the middle of each cell. In risk analysis, this uniform sampling is replaced by sampling from a prespecified distribution for the input values.)

Because LHS implies randomness, the resulting design may happen to include *outlier* scenarios; for example, it might happen (with small probability) that in Fig. 18.4 all scenarios lie on the main diagonal, so the values of the two inputs have a correlation coefficient of -1. Therefore LHS may be adjusted to give (nearly) orthogonal designs; see Cioppa and Lucas (2007).

Let us compare classic designs and LHS geometrically. Figure 18.3 illustrates that many classic designs consists of corners of k-dimensional cubes; these designs imply simulation of *extreme scenarios*. Figure 18.4 illustrates that LHS has better *space-filling* properties.

Table 18.5 A LHS design for two factors and four scenarios

Scenario	Interval factor 1	Interval factor 2
1	2	1
2	1	4
3	4	3
4	3	2

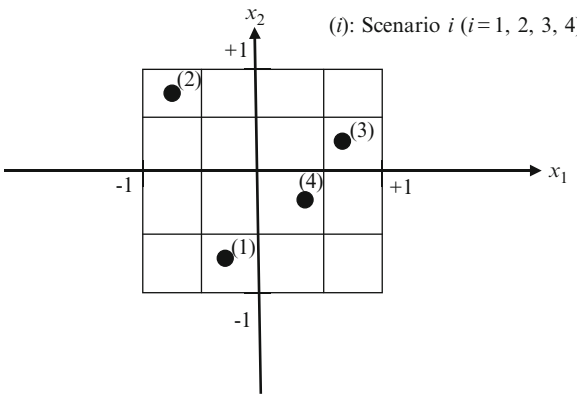


Fig. 18.4 A LHS design for two factors and four scenarios

This property has inspired many statisticians to develop other space-filling designs. One type maximizes the minimum Euclidean distance between any two points in the k-dimensional experimental area. Related designs minimize the maximum distance. Many references and websites are given by Kleijnen (2008, pp. 126–130).

Besides these one-shot designs, simulation may use *sequential* designs, because simulation proceeds sequentially. In sensitivity analysis the design may simulate most scenarios in those subareas in which the I/O behavior is not linear; see Kleijnen and Van Beers (2004). In optimization the design may focus on the most promising area, while also simulating other areas to avoid getting trapped in a local optimum; i.e., the design balances exploration and exploitation (see Forrester et al. (2008) and Kleijnen, Van Beers, and van Nieuwenhuyse (2010)).

18.6 Conclusions

Because simulation implies *experimentation* with a model, design of experiments (DOE) is essential. In this chapter, I treat the simulation model as a black box, and presented both *classic* designs for low-order polynomial regression metamodels and *modern* designs (including Latin hypercube sampling) for other metamodels such as Kriging. The simpler the metamodel is, the fewer scenarios need to be simulated. (Cross validation of the metamodel selected, is discussed in Chap. III.1 by Wang.)

I did not discuss so-called *optimal designs* because these designs use statistical assumptions (such as white noise) that I find too unrealistic. Optimal designs are discussed by Pronzato and Zhigljavsky (2009),; also see Kleijnen (2008, pp. 51–54).

Neither did I discuss the designs in Taguchi (1987), because I think that the classic and modern designs (which I did discuss) are superior. Nevertheless, I think that Taguchi's concepts – as opposed to his statistical techniques – are important. In practice, the 'optimal' solution may break down because the environment turns

out to differ from the environment that the analysts assumed when deriving the optimum. Therefore they should look for a 'robust' solution. For further discussion I refer to Dellino et al. (2011).

I mentioned several more research issues; for example, importance sampling in 'rare event' simulation. Another interesting question is: how much computer time should analysts spend on *replication*; how much on exploring *new* scenarios?

Another challenge is to develop designs and metamodels that explicitly account for *multiple outputs*. In practice, multiple outputs are the rule in simulation.

The application of *Kriging* to *random* simulation models (such models are a focus of this handbook, including this chapter) is also a challenge. Moreover, corresponding software needs to be developed (current software focuses on deterministic simulation).

Comparison of various metamodel types and their designs remains a major problem. Besides linear regression and Kriging, Kleijnen (2008, p. 8) gives references for Classification And Regression Trees (CART), Generalized Linear Models (GLM), Multivariate Adaptive Regression Splines (MARS), neural networks, nonlinear regression models, nonparametric regression analysis, radial functions, rational functions, splines, support vector regression, symbolic regression, and wavelets. Comparison of screening designs has hardly begun; see Kleijnen (2010).

References

Angün, E., den Hertog, D., Gürkan, G., Kleijnen, J.P.C.: Response surface methodology with stochastic constrains for expensive simulation. J. Oper. Res. Soc. **60**, 735–746 (2009)

Ankenman, B., Nelson, B., Staum, J.: Stochastic kriging for simulation metamodeling. Oper. Res. Accepted **58**(2), 371–382 (2010)

Cioppa, T.M., Lucas, T.W.: Efficient nearly orthogonal and space-filling latin hypercubes. Technometrics **49**(1), 45–55 (2007)

Cressie, N.A.C.: Statistics for spatial data. Wiley, New York (1993)

Dellino, G., Kleijnen, J.P.C., Meloni, C.: Robust optimization in simulation: Taguchi and Krige combined. *INFORMS Journal on Computing* (accepted 28 March 2011)

Efron, B., Tibshirani, R.J.: An introduction to the bootstrap. Chapman & Hall, New York (1993)

Forrester, A., Sóbester, A., Keane, A.: Engineering design via surrogate modelling: a practical guide. Wiley, Chichester (2008)

Horne, G., Leonardi, M. (eds.): Maneuver warfare science 2001. Defense Automatic Printing Service, Quantico (2001)

Kleijnen, J.P.C.: Factor screening in simulation experiments: review of sequential bifurcation. In: Alexopoulos, C., Goldsman, D., Wilson, J.R. (eds.) Advancing the frontiers of simulation: a Festschrift in honor of George S. Fishman, pp. 169–173. Springer, New York (2010)

Kleijnen, J.P.C.: Design and analysis of simulation experiments. Springer (2008)

Kleijnen, J.P.C., Sargent, R.G.: A methodology for the fitting and validation of metamodels in simulation. Eur. J. Oper. Res. **120**(1), 14–29 (2000)

Kleijnen, J.P.C., Van Beers, W.C.M.: Robustness of Kriging when interpolating in random simulation with heterogeneous variances: some experiments. Eur. J. Oper. Res. **165**(3), 826–834 (2005)

Kleijnen, J.P.C., Van Beers, W.C.M.: Application-driven sequential designs for simulation experiments: Kriging metamodeling. J. Oper. Res. Soc. **55**(9), 876–883 (2004)

Kleijnen, J.P.C., Van Beers, W.C.M., van Nieuwenhuyse, I.: Constrained optimization in simulation: a novel approach. Eur. J. Oper. Res. **202**, 164–174 (2010)

Lophaven, S.N., Nielsen, H.B., Sondergaard, J.: DACE: a Matlab kriging toolbox, version 2.0. IMM Technical University of Denmark, Lyngby (2002)

McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics. **21**(2), 239–245 (1979) (reprinted in 2000: Technometrics **42**(1), 55–61)

Myers, R.H., Montgomery, D.C., Anderson-Cook, C.M.: Response surface methodology: process and product optimization using designed experiments. Wiley, New York (2009)

Pronzato, L., Zhigljavsky, A. (eds.): Optimal design and related areas in optimization and statistics. Springer (2009)

Sacks, J., Welch, W.J., Mitchell, T.J., Wynn, H.P.: Design and analysis of computer experiments. Stat. Sci. **4**(4), 409–435 (1989)

Santner, T.J., Williams, B.J., Notz, W.I.: The design and analysis of computer experiments. Springer, New York (2003)

Taguchi, G.: System of experimental designs, Volumes 1 and 2. White Plains. UNIPUB/Krauss International, New York (1987)

Yin, J., Ng, S.H., Ng, K.M.: A study on the effects of parameter estimation on Kriging model's prediction error in stochastic simulations. In: Rossini, M.D., Hill, R.R., Johansson, B., Dunkin, A., Ingalls, R.G. (eds.) Proceedings of the 2009 Winter Simulation Conference (2009)

Zeigler, B.P., Praehofer, K., Kim, T.G.: Theory of modeling and simulation. Academic, New York (2000)

Chapter 19
Multivariate Density Estimation and Visualization

David W. Scott

19.1 Introduction

This chapter examines the use of flexible methods to approximate an unknown density function, and techniques appropriate for visualization of densities in up to four dimensions. The statistical analysis of data is a multilayered endeavor. Data must be carefully examined and cleaned to avoid spurious findings. A preliminary examination of data by graphical means is useful for this purpose. Graphical exploration of data was popularized by Tukey (1977) in his book on exploratory data analysis (EDA). Modern data mining packages also include an array of graphical tools such as the histogram, which is the simplest example of a density estimator. Exploring data is particularly challenging when the sample size is massive or if the number of variables exceeds a handful. In either situation, the use of nonparametric density estimation can aid in the fundamental goal of understanding the important features hidden in the data. In the following sections, the algorithms and theory of nonparametric density estimation will be described, as well as descriptions of the visualization of multivariate data and density estimates. For simplicity, the discussion will assume the data and functions are continuous. Extensions to discrete and mixed data are straightforward.

Statistical modeling of data has two general purposes: (1) understanding the shape and features of data through the density function, $f(\mathbf{x})$, and (2) prediction of y through the joint density function, $f(\mathbf{x}, y)$. When the experimental setting is well-known, parametric models may be formulated. For example, if the data are multivariate normal, $N(\mu, \Sigma)$, then the features of the density may be extracted from the maximum likelihood estimates of the parameters μ and Σ. In particular, such data have one feature, which is a single mode located at μ. The shape of the data cloud is elliptical, and the eigenvalues and eigenvectors of the covariance

D.W. Scott (✉)
Department of Statistics, Rice University, Houston, TX, USA
e-mail: scottdw@rice.edu

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_19,

matrix, Σ, indicate the orientation of the data and the spread in those directions. If the experimental setting is not well-known, or if the data do not appear to follow a parsimonious parametric form, then nonparametric density estimation is indicated. The major features of the density may be found by counting and locating the sample modes. The shape of the density cannot easily be determined algebraically, but visualization methodology can assist in this task. Similar remarks apply in the regression setting.

When should parametric methods be used and when should nonparametric methods be used? A parametric model enjoys the advantages of well-known properties and parameters which may be interpreted. However, using parametric methods to explore data makes little sense. The features and shape of a normal fit will always be the same no matter how far from normal the data may be. Nonparametric approaches can fit an almost limitless number of density functional forms. However, at the model, parametric methods are always more statistically accurate than the corresponding nonparametric estimates. This statement can be made more precise by noting that parametric estimators tend to have lower variance, but are susceptible to substantial bias when the wrong parametric form is invoked. Nonparametric methods are not unbiased, but the bias asymptotically vanishes for any continuous target function. Nonparametric algorithms generally have greater variance than a parametric algorithm. Construction of optimal nonparametric estimates requires a data-based approach in order to balance the variance and the bias, and the resulting mean squared error generally converges at a rate slower than the parametric rate of $O(n^{-1})$. In summary, nonparametric approaches are always appropriate for exploratory purposes, and should be used if the data do not follow a simple parametric form.

19.2 Visualization

19.2.1 Data Visualization

Visualization of data is a fundamental task in modern statistical practice. The most common figure for this purpose is the bivariate scatter diagram. Figure 19.1a displays the levels of blood fats in a sample of men with heart disease. The data have been transformed to a logarithm base 10 scale to minimize the effects of skewness. At a first glance, the data appear to follow a bivariate normal distribution. The sample correlation is only 0.22. One might examine each of the two variables separately as a univariate scatter diagram, which is commonly referred to as a "dot plot", but such figures are rarely presented. Tukey advocated the histogram-like stem-and-leaf plot or the box-and-whiskers plot, which displays simple summaries including the median and quartiles. Figure 19.1b displays box-and-whisker plots for these variables. Clearly triglyceride values vary more than cholesterol and may still be right-skewed.

As shown later in Sect. 19.3.3, there may be rather subtle clusters within these data. The eye can readily detect clusters which are well-separated, but the eye is

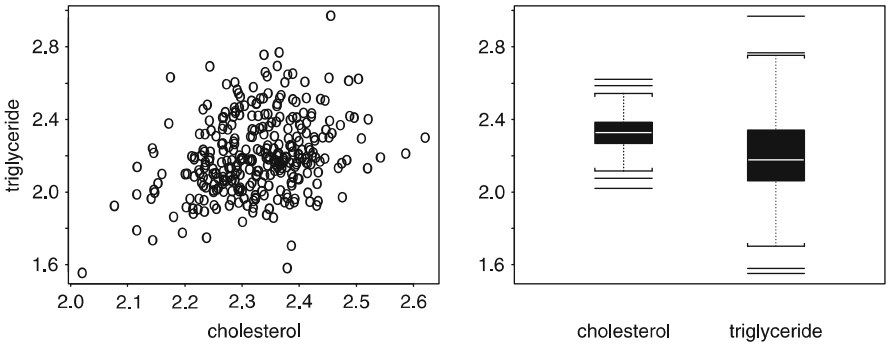


Fig. 19.1 Cholesterol and triglyceride blood levels for 320 males with heart disease

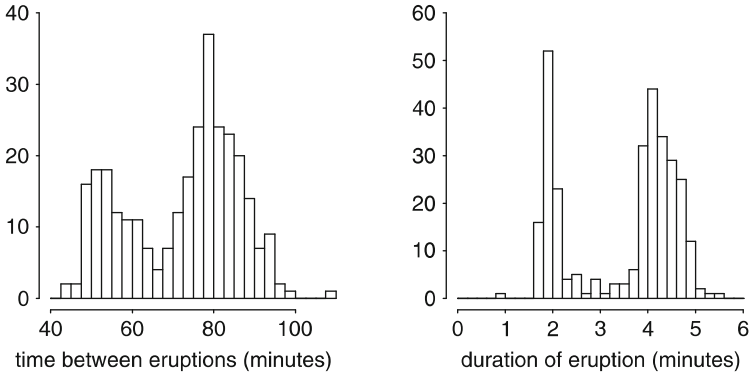


Fig. 19.2 Waiting time and duration of 299 consecutive eruptions of the Old Faithful Geyser

not reliable when the clusters are not well-separated, nor when the sample size is so large that the scatter diagram is too crowded. For example, consider the Old Faithful Geyser data (Azzalini and Bowman 1990), (x_t, y_t), where x_t measures the waiting time between successive eruptions of the geyser, and y_t measures the duration of the subsequent eruption. The data were blurred by adding uniform noise to the nearest minute for x_t and to the nearest second for y_t. Figure 19.2 displays histograms of these two variables. Interestingly, neither appears to follow the normal distribution. The common feature of interest is the appearance of two modes. One group of eruptions is only 2 minutes in duration, while the other averages over 4 minutes in duration. Likewise, the waiting time between eruptions clusters into two groups, one less than an hour and the other greater than one hour. The distribution of eruption durations appears to be a mixture of two normal densities, but the distribution of the waiting times appears more complicated.

Finally, in Fig. 19.3 we examine the scatter diagrams of both (x_t, y_t) as well as the lagged values of eruption duration, (y_{t-1}, y_t). The common feature in these two densities is the presence of three modes. As mentioned earlier, the eye is well-suited to discerning clusters that are well-separated. From Fig. 19.3a, short waiting periods

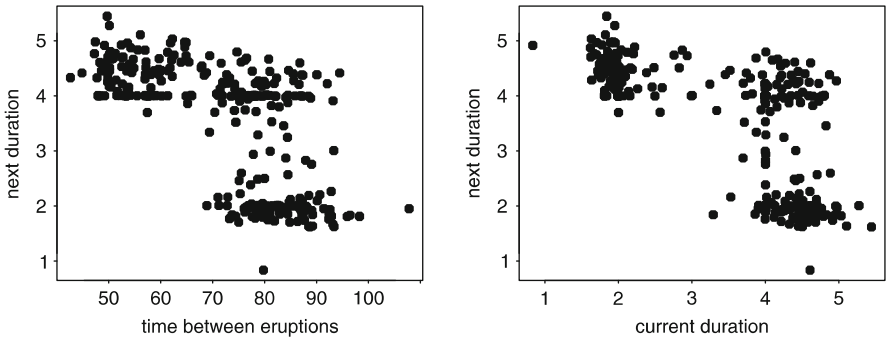


Fig. 19.3 Two scatter diagrams of the Old Faithful Geyser data

are associated with long eruption durations. From Fig. 19.3b, all eruptions of short duration are followed by eruptions of long duration. Missing from Fig. 19.3b are any examples of eruptions of short duration following eruptions of short duration, which should be a plus for the disappointed tourist. The observant reader may notice an odd clustering of points at integer values of the eruption duration. A quick count shows that 23, 2, and 53 of the original 299 values occur exactly at $y = 2, 3$, and 4 minutes, respectively. Examining the original time sequence suggests that these measurements occur in clumps; perhaps accurate measurements were not taken after dark. Exploration of these data has revealed not only interesting features but also suggest possible data collection anomalies.

Massive datasets present different challenges. For example, the Landsat IV remote sensing dataset described by Scott (1992) contains information on 22,932 pixels of a scene imaged in 1977 from North Dakota. The variables displayed in Fig. 19.4 are the time of peak greenness of the crop in each pixel and the derived value of the maximum greenness, scaled to values 0−255 and blurred with uniform noise. Overplotting is apparent. Each successive figure drills down into the boxed region shown. Only 5.6% of the points are eliminated going to the second frame; 35.5% eliminated between the second and third frames; and 38.1% between the third and final frames, still leaving 8624 points. Overplotting is still apparent in the final frame. Generally, gleaning detailed density information from scatter diagrams is difficult at best. Nonparametric density estimation solves this problem.

To see the difficulty of gleaning density information from the graphs in Fig. 19.4, compare the bivariate histogram displayed in Fig. 19.5 for the data in frame (b) from Fig. 19.4. Using only the scatter diagram, there is no way to know the relative frequency of data in the two largest clusters except through the histogram.

The bivariate histogram uses rectangular-shaped bins. An interesting hybrid solution is to use hexagonal-shaped bins and to use a glyph to represent the bin count. Scott (1988) compared the statistical power of using squares, hexagons, and equilateral triangles as shapes for bins of bivariate histograms and concluded that hexagons were the best choice. Carr et al. (1992) examined the

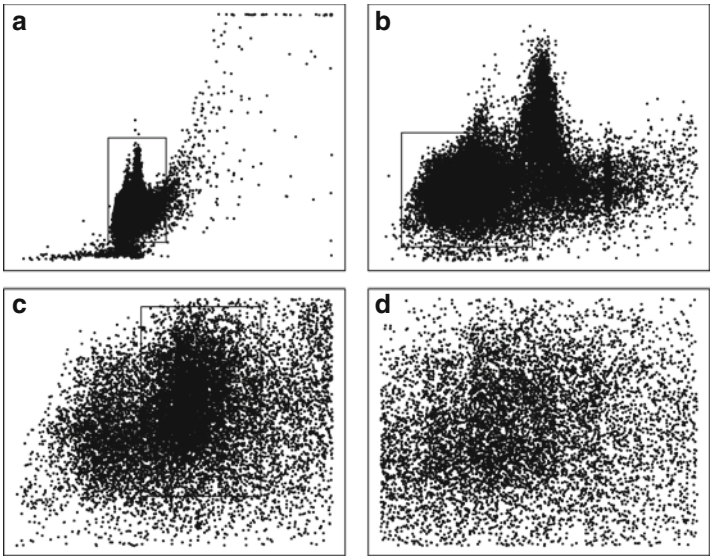
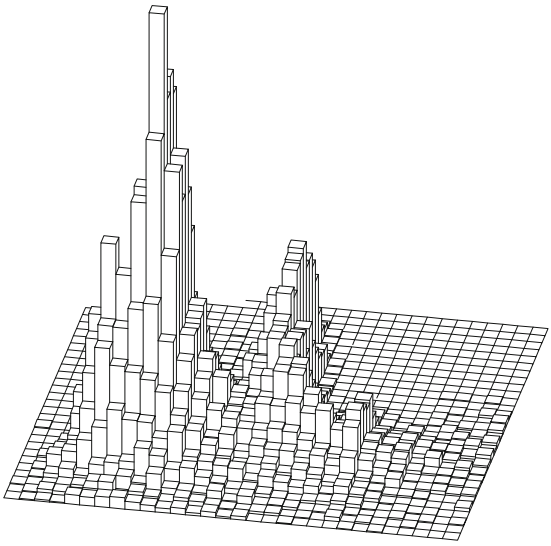


Fig. 19.4 Drilling into the Landsat IV data with $n = 22932$

Fig. 19.5 Histogram of data in Fig. 19.4b



use of drawing a glyph in each bivariate bin rather than the perspective view. For graphical reasons, Carr found hexagonal bins were more effective. The bin count is represented by a hexagonal glyph whose area is proportional to the bin count. Figure 19.6 displays the hexagonal mosaic map of the same data as in Fig. 19.5. This representation gives a quite accurate summary of the density

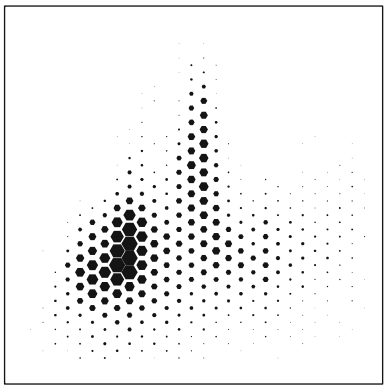


Fig. 19.6 Hexagonal bin glyph of the data in Fig. 19.4b

information. No bin counts are obscured as in the perspective view of the bivariate histogram.

In the next section, some of the algorithms for nonparametric density estimation and their theoretical properties are discussed. We then return to the visualization of data in higher dimensions.

19.3 Density Estimation Algorithms and Theory

This section includes enough algorithms and results to obtain a basic understanding of the opportunities and issues. Fortunately, there have been a number of readable monographs available for the reader interested in pursuing this subject in depth. In rough chronological order, excluding books primarily dealing with nonparametric regression, the list includes Tapia and Thompson (1978), Wertz (1978), Prakasa Rao (1983), Devroye and Gyorfi (1985), Silverman (1986), Devroye (1987), Nadaraya (1989), Hardle (1990), Scott (1992), Tarter and Lock (1993), Wand and Jones (1995), Simonoff (1996), Bowman and Azzalini (1997), and Tarter (2000).

The purpose of the next section is to provide a survey of important results without delving into the theoretical underpinnings and details. The references cited above are well-suited for that purpose.

19.3.1 A High-Level View of Density Theory

Smoothing Parameters

Every algorithm for nonparametric density estimation has one or more design parameters which are called the smoothing parameter(s) or bandwidth(s) of the procedure. The smoothing parameter controls the final appearance of the estimate. For an equally-spaced histogram, the bin width plays the primary role of

a smoothing parameter. Of course, the bins may be shifted and the location of the bin edges is controlled by the bin origin, which plays the role of a secondary smoothing parameter. For a kernel estimator, the scale or width of the kernel serves as the smoothing parameter. For an orthogonal series estimator, the number of basis functions serves as the smoothing parameter. The smoothing parameters of a spline estimator also include the location of the knots. Similarly, a histogram with completely flexible bin widths has many more than two smoothing parameters.

No Unbiased Density Estimators

As a point estimator of $f(x)$, Rosenblatt (1956) proved that every nonparametric density estimator, $\widehat{f}(x)$ is biased. However, it is usually true that the integral of all of the pointwise biases is 0. Thus mean squared error (MSE) and integrated mean squared error (IMSE) are the appropriate criteria to optimize the tradeoff between pointwise/integrated variance and squared bias.

Nonparametric density estimators always underestimate peaks and overestimate valleys in the true density function. Intuitively, the bias is driven by the degree of curvature in the true density. However, since the bias function is continuous and integrates to 0, there must be a few points where the bias does vanish. In fact, letting the smoothing parameter vary pointwise, there are entire intervals where the bias vanishes, including the difficult-to-estimate tail region. This fact has been studied by Hazelton (1996) and Sain and Scott (2002). Since the bias of a kernel estimator does not depend on the sample size, these zero-bias or bias-annihilating estimates have more than a theoretical interest. However, there is much more work required for practical application. Alternatively, in higher dimensions away from peaks and valleys, one can annihilate pointwise bias by balancing directions of positive curvature against directions of negative curvature; see Terrell and Scott (1992). An even more intriguing idea literally adjusts the raw data points towards peaks and away from valleys to reduce bias; see Choi and Hall (1999).

Rates of Convergence

The rate of convergence of a nonparametric density estimator to the true density is much slower than in the parametric setting, assuming in the latter case that the correct parametric model is known. If the correct parametric model is not known, then the parametric estimates will converge but the bias will not vanish. The convergence is slower still in high dimensions, a fact which is often referred to as the *curse of dimensionality*. Estimating the derivative of a density function is even harder than coping with an additional dimension of data.

If the k-th derivative of a density is known to be smooth, then it is theoretically possible to construct an order-k nonparametric density estimation algorithm. The pointwise bias is driven by the k-th derivative at x, $f^{(k)}(x)$. However, if $k > 2$, then the density estimate will take on negative values for some points, x. It is possible to

define higher-order algorithms which are non-negative, but these estimates do not integrate to 1; see Terrell and Scott (1980). Thus higher-order density estimation algorithms violate one of the two conditions for a true density: $f(x) \geq 0$ and $\int_{-\infty}^{\infty} f(x)\,\mathrm{d}x = 1$. Of course, there are cases where the first condition is violated for lower-order estimators. Two such cases include orthogonal series estimators (Kronmal and Tarter 1968; Watson 1969) and boundary-corrected kernel estimators (Rice 1984). Note that positive kernel estimators correspond to $k = 2$. Wahba (1981) studied the efficacy of higher-order procedures and suggested $k = 3$ often provided superior estimates. Scott (1992) also studied this question and found some improvement when $k = 3$, which must be traded off against the disadvantages of negative estimates.

Choosing Bandwidths in Practice

Picking the best smoothing parameter from data is an important task in practice. If the smoothing parameter is too small, the estimate is too noisy, exhibiting high various and extraneous wiggles. If the smoothing parameter is too large, then the estimate may miss key features due to oversmoothing, washing out small details. Such estimates have low variance but high bias in many regions. In practice, bandwidths that do not differ by more than 10−15% from the optimal bandwidth are usually satisfactory.

A statistician experienced in EDA is likely to find all estimates informative for bandwidths ranging from undersmoothed to oversmoothed. With a complicated density function, no single choice for the bandwidth may properly represent the density for all values of x. Thus the same bandwidth may result in undersmoothing for some intervals of x, oversmoothing in another interval, and yet near optimal smoothing elsewhere. However, the practical difficulty of constructing locally adaptive estimators makes the single-bandwidth case of most importance. Simple transformations of the data scale can often be an effective strategy (Wand et al. 1991). This strategy was used with the lipid data in Fig. 19.1, which were transformed to a $\log_{10}$ scale.

Consider the 21640 x points shown in frame (b) of Fig. 19.4. Histograms of these data with various numbers of bins are shown in Fig. 19.7. With so much data, the

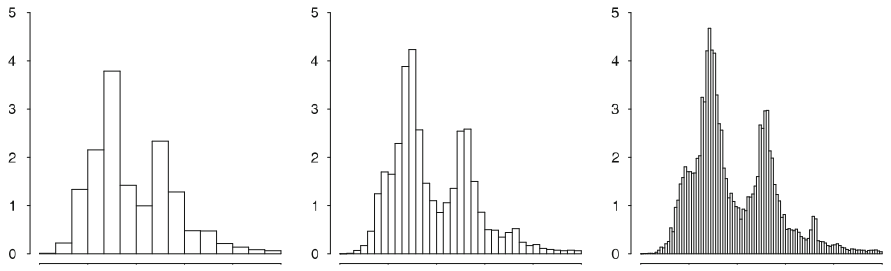


Fig. 19.7 Histograms of x variable in Fig. 19.4b with 15, 35, and 100 bins

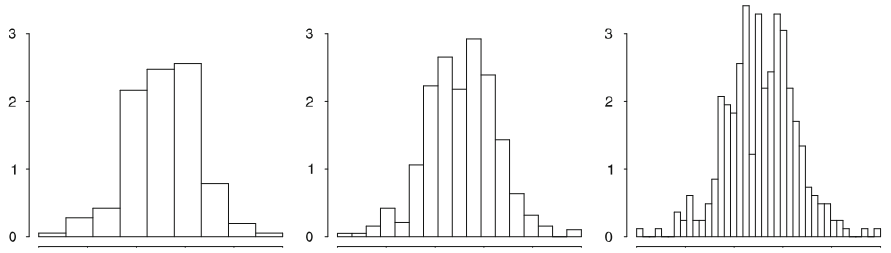


Fig. 19.8 Histograms of $\log_{10}$-cholesterol variable in Fig. 19.1 with 9, 19, and 39 bins

oversmoothed histogram Fig. 19.7a captures the major features, but seems biased downwards at the peaks. The final frame shows a histogram that is more useful for finding data anomalies than as a good density estimate.

The differences are more apparent with a smaller sample size. Consider the 320 $\log_{10}$-cholesterol levels shown in Fig. 19.1. Three histograms are shown in Fig. 19.8. The extra one or two modes are at least suggested in the middle panel, while the histogram in the first panel only suggests a rather unusual non-normal appearance. The third panel has many large spurious peaks. We conclude from these two figures that while an oversmoothed estimator may have a large error relative to the optimal estimator, the absolute error may still be reasonably small for very large data samples.

Oversmoothed Bandwidths

While there is no limit on how complicated a density may be (for which $\int f^{(k)}(x)^2 \, \mathrm{d}x$ may grow without bound), the converse is not true. Terrell and Scott (1985) and Terrell (1990) show that for a particular scale of a density (for example, the range, standard deviation, or interquartile range), there is in fact a lower bound among continuous densities for the roughness quantity

$$R\left(f^{(k)}\right) = \int_{-\infty}^{\infty} f^{(k)}(x)^2 \, \mathrm{d}x \, . \tag{19.1}$$

In a real sense, such densities are the smoothest possible and are the easiest to estimate. The optimal bandwidth for these "oversmoothed densities" serves as an upper bound. Specifically, any other density with the same scale will have more complicated structure and will require a smaller bandwidth to more accurately estimate those features. Since oversmoothed bandwidths (and reference bandwidths as well) only use the data to estimate the scale (variance, for example), these data-based estimates are quite stable. Obtaining similar highly stable data-based nearly optimal bandwidth estimators requires very sophisticated estimates of the roughness function given in 19.1. One algorithm by Hall et al. (1991) is often highly

rated in practice (Jones et al. 1996). seemed closely related to the oversmoothed bandwidths. These approaches all rely on asymptotic expansions of IMSE rather than an unbiased risk estimate, which underlies the least-squares or unbiased cross-validation algorithm introduced by Rudemo (1982) and Bowman (1984). However, the unbiased risk approach has numerous extensions; see Sain and Scott (1996) and Scott (2001). Another algorithm that should be mentioned is the bootstrap bandwidth. For a Gaussian kernel, the bootstrap with an infinite number of repetitions has a closed form expression; see Taylor (1989). Multivariate extensions are discussed by Sain et al. (1994).

Many details of these ideas may be found in the literature and in the many textbooks available. The following section provides some indication of this research.

19.3.2 The Theory of Histograms

The basic results of density estimation are perhaps most easily understood with the ordinary histogram. Thus more time will be spent on the histogram with only an outline of results for more sophisticated and more modern algorithms.

Given an equally spaced mesh $\{t_k\}$ over the entire real line with $t_{k+1} - t_k = h$, the density histogram is given by

$$\widehat{f}(x) = \frac{\nu_k}{nh} \qquad \text{for } t_k < x < t_{k+1}\,, \tag{19.2}$$

where ν_k is the number of data points falling in the k-th bin. Clearly, $\sum_k \nu_k = n$ and ν_k is a Binomial random variable with mean $p_k = \int_{t_k}^{t_{k+1}} f(x)\,\mathrm{d}x$; hence, $\mathrm{E}\nu_k = np_k$ and $\mathrm{Var}\nu_k = np_k(1-p_k)$. Thus the pointwise variance of the histogram (19.2) is $np_k(1-p_k)/(nh)^2$, which is constant for all x in the k-th bin. Thus, the integrated variance (IV) over $(-\infty, \infty)$ may be found by integrating the pointwise variance over the k-th bin (i.e., multiply by the bin width h), and summing over all bins:

$$\mathrm{IV} = \sum_{k=-\infty}^{\infty} \frac{np_k(1-p_k)}{(nh)^2} \times h = \sum_{k=-\infty}^{\infty} \frac{p_k(1-p_k)}{nh} = \frac{1}{nh} - \sum_k \frac{p_k^2}{nh}\,, \tag{19.3}$$

since $\sum p_k = \int_{-\infty}^{\infty} f(x)\,\mathrm{d}x = 1$. The final term may be shown to approximate $n^{-1}\int f(x)^2\,\mathrm{d}x$, which is asymptotically negligible. Thus the global integrated variance of the histogram can be controlled by collecting more data or choosing a wider bin width.

Next consider the bias of $\widehat{f}$ at a fixed point, x, which is located in the k-th bin. Note that $\mathrm{E}\,\widehat{f}(x) = np_k/nh = p_k/h$. A useful approximation to the bin probability is

$$p_k = \int_{t_k}^{t_{k+1}} f(y)\,\mathrm{d}y = h\,f(x) + h^2\left(\frac{1}{2} - \frac{x-t_k}{h}\right) f'(x) + \ldots\,, \tag{19.4}$$

replacing the integrand $f(y)$ by $f(x) + (y - x) f'(x) + \ldots$. Thus the pointwise bias may be approximated by

$$\text{Bias}\, \widehat{f}(x) = \text{E}\, \widehat{f}(x) - f(x) = \frac{p_k}{h} - f(x) = h \left(\frac{1}{2} - \frac{x - t_k}{h} \right) f'(x) + \ldots . \tag{19.5}$$

Therefore, the bias is controlled by the first derivative of the unknown density at x. Since $t_k < x < t_{k+1}$, then the factor $(1/2 - (x - t_k)/h)$ in (19.5) varies from $-1/2$ to $1/2$. Thus the bias is also directly proportional to the bandwidth, h. To control the bias of the histogram estimate, the bandwidth h should be small. Comparing (19.3) and (19.5), the global consistency of the histogram can be guaranteed if, as $n \to \infty$, $h \to 0$ while ensuring that the product $nh \to \infty$ as well, for example, if $h = 1/\sqrt{n}$ (see Duda and Hart 1973).

A more complete analysis of the bias (Scott 1979) shows that the integrated squared bias is approximately $h^2\, R(f')/12$, where $R(f') = \int f'(x)^2\, \mathrm{d}x$, so that the IMSE is given by

$$\text{IMSE}\left[\widehat{f}_k \right] = \frac{1}{nh} + \frac{1}{12} h^2 R(f') + O(n^{-1}) . \tag{19.6}$$

From this equation, the optimal bandwidth is seen to be

$$h^* = \left[\frac{6}{nR(f')} \right]^{1/3} \quad \text{and} \quad \text{IMSE}^* = \left(\frac{9}{16} \right)^{1/3} R(f')^{1/3}\, n^{-2/3} . \tag{19.7}$$

Thus the optimal bandwidth approaches zero at the rate $O(n^{-1/3})$ and not the rate $O(n^{-1/2})$ as suggested by Duda and Hart (1973) nor the rate $O(1/\log n)$ as suggested by Sturges (1926). With regards to IMSE, the best rate a histogram can achieve is of order $O(n^{-2/3})$, which falls well short of the parametric rate of $O(n^{-1})$. From (19.7), the larger the value of the roughness $R(f')$ of the true density, the smaller the optimal bandwidth and the larger the average error.

Finally, the smoothest density with variance σ^2 is

$$g(x) = \frac{15}{16\sqrt{7}\sigma} \left(1 - \frac{x^2}{7\sigma^2} \right)^2 \qquad -\sqrt{7}\sigma < x < \sqrt{7}\sigma \tag{19.8}$$

and zero elsewhere, for which $R(g') = 15\sqrt{7}/(343\sigma^3)$. Since $R(f') \geq R(g')$ for any other continuous density, f,

$$h^* = \left[\frac{6}{nR(f')} \right]^{1/3} \leq \left[\frac{6}{nR(g')} \right]^{1/3} = \left[\frac{686\, \sigma^3}{5\sqrt{7}n} \right]^{1/3} = 3.73\, \sigma\, n^{-1/3} , \tag{19.9}$$

which is the "oversmoothed bandwidth" rule. Consider the normal reference rule, $f = \phi = N(\mu, \sigma^2)$, for which $R(\phi') = 1/(4\sqrt{\pi}\sigma^3)$, which when substituted

into (19.7) gives $h^* = 3.49\,\sigma\, n^{-1/3}$, a value that is only 6.4% narrower than the oversmoothed bandwidth.

The oversmoothing rule (19.9) may be inverted when the scale is the range of the density to obtain a lower bound of $\sqrt[3]{2n}$ on the number of bins in the optimal histogram. This formula should be compared to Sturges' rule of $1 + \log_2 n$, which is in common use in many statistical packages (Sturges 1926). In fact, the histograms in the first frames of Figs. 19.7 and 19.8 correspond to Sturges' rule, while the second frames of these figures correspond to the oversmoothed bandwidths. Presumably the optimal bandwidth would occur somewhere between the second and third frames of these figures. Clearly Sturges' rule results in oversmoothed graphs since the optimal number of bins is severely underestimated.

19.3.3 ASH and Kernel Estimators

The histogram is an order-one density estimator, since the bias is determined by the first derivative. The estimators used most in practice are order-two estimators. (Recall that order-three estimators are not non-negative.) Perhaps the most unexpected member of the order-two class is the frequency polygon (FP), which is the piecewise linear interpolant of the midpoints of a histogram. (Scott 1985a) showed that

$$\mathrm{IMSE}\left[\widehat{f}_{\mathrm{FP}}\right] = \frac{2}{3nh} + \frac{49}{2880}h^4 R(f'') + O\left(n^{-1}\right). \tag{19.10}$$

Compared to Equation (19.6), the integrated variance of a FP is 33% smaller and the integrated squared bias is two orders of magnitude smaller. Clearly, $h^* = O(n^{-1/5})$ and $\mathrm{IMSE}^* = O(n^{-4/5})$. Thus the error converges at a faster rate than the histogram, by using bins which are wider and an estimator which is not discontinuous. Examples and other results such as oversmoothing may be found in Scott (1992).

The use of wider bins means that the choice of the bin origin has a larger impact, at least subjectively. Given a set of m shifted histogram, $\widehat{f}_1(x), \ldots, \widehat{f}_m(x)$, one might use cross-validation to try to pick the best one. Alternatively, Scott (1985b) suggested the averaged shifted histogram (ASH), which is literally defined:

$$\widehat{f}_{\mathrm{ASH}}(x) = \frac{1}{m}\sum_{k=1}^{m} \widehat{f}_k(x)\,. \tag{19.11}$$

To be specific, suppose the collection of m histograms has meshes shifted by an amount $\delta = h/m$ from each other. Recompute the bin counts, ν_k, on the finer mesh, $t'_k = k\delta, -\infty < k < \infty$. Then a bin count for one of the histograms with bin width h may be computed by adding m of the bin counts on the finer mesh. For x in the ℓ-th (narrow) bin, there are m shifted histograms that include the (narrow) bin

count, ν_ℓ. Adding these m shifted histograms together and averaging gives:

$$\frac{\nu_{\ell+1-m} + 2\nu_{\ell+2-m} + \ldots + m\,\nu_\ell + \ldots + 2\nu_{\ell+m-2} + \nu_{\ell+m-1}}{m \times nh}\,, \tag{19.12}$$

or after re-arranging

$$\widehat{f}_{\mathrm{ASH}}(x) = \frac{1}{nh} \sum_{k=1-m}^{m-1} \left(1 - \frac{|k|}{m}\right) \nu_{\ell+k}\,. \tag{19.13}$$

As the number of shifted histograms $m \to \infty$, the weights on the bin counts approaches the triangular kernel given by $K(t) = 1 - |t|$ for $|t| < 1$ and zero elsewhere. The ASH may be generalized to handle general weights by sampling from an arbitrary kernel function, $K(t)$, which is any symmetric probability density defined on the interval $[-1, 1]$. In this case,

$$\widehat{f}_{\mathrm{ASH}}(x) = \frac{1}{nh} \sum_{k=1-m}^{m-1} w_m(k)\,\nu_{\ell+k} \qquad \text{where} \quad w_m(k) \propto K(k/m)\,. \tag{19.14}$$

Like the FP, the ASH is an order-two algorithm, but more efficient in the statistical sense.

In Fig. 19.9, two ASHs of the $\log_{10}$-cholesterol data are shown. The bin edge effects and discontinuities apparent in the ordinary histogram in Fig. 19.8 are removed. The extra features in the distribution are hinted at.

The extension of the ASH to bivariate (and multivariate) data is straightforward. A number of bivariate (multivariate) histograms are constructed with equal shifts along the coordinate axes and then averaged together. Figure 19.10 displays a bivariate ASH of the same lipid data displayed in Fig. 19.1. The strong bimodal and weak trimodal features are evident. The third mode is perhaps more clearly represented in a perspective plot; see Fig. 19.11. (Note that for convenience, the data were rescaled to the intervals $(0, 1)$ for these plots, unlike Fig. 19.1.) The precise location of the third mode above (and between) the two primary modes results in the masking of the multiple modes when viewed along the cholesterol axis alone. This masking feature is commonplace and a primary reason for trying to extend the dimensions available for visualization of the density function.

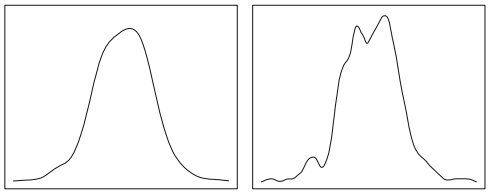


Fig. 19.9 Averaged shifted histograms of the $\log_{10}$-cholesterol data

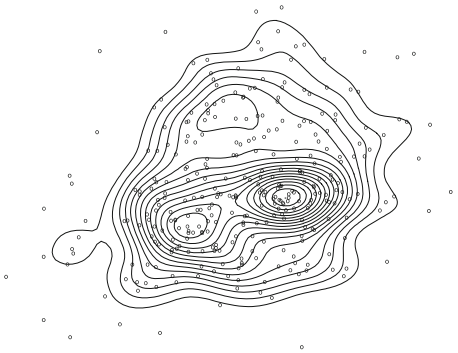


Fig. 19.10 Bivariate ASH of the $\log_{10}$-cholesterol and $\log_{10}$-triglyceride data

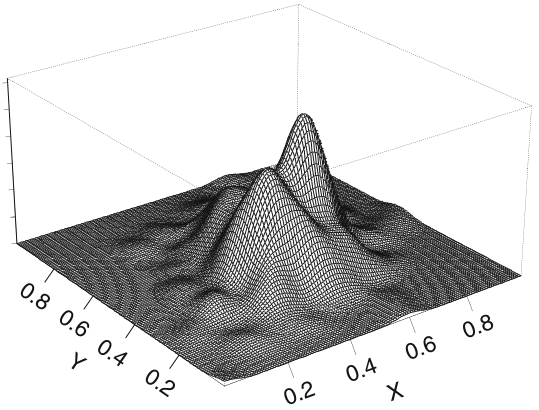


Fig. 19.11 Perspective view of the Bivariate ASH in Fig. 19.10

19.3.4 Kernel and Other Estimators

The ASH is a discretized representation of a kernel estimator. Binned kernel estimators are of great interest to reduce the computational burden. An alternative to the ASH is the fast Fourier transform approach of Silverman (1982). Kernel methods were introduced by Rosenblatt (1956) and Parzen (1962) with earlier work by Evelyn Fix and Joe Hodges completed by 1951 in San Antonio, Texas (see Silverman and Jones 1989).

Given a kernel function, $K(t)$, which is generally taken to be a symmetric probability density function, the kernel density estimate is defined by

$$\widehat{f}_K(x) = \frac{1}{nh}\sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) = \frac{1}{n}\sum_{i=1}^{n} K_h(x - x_i)\,, \tag{19.15}$$

letting K_h denote the kernel density transformed by the scale factor, h; that is, $K_h(t) = K(t/h)/h$. Among kernels with finite support, Beta densities shifted to

the interval $(-1, 1)$ are popular choices. Among kernels with infinite support, the normal density is by far the most common choice. An important paper by Silverman (1981) showed that the normal kernel has the unique property that the number of modes in the kernel estimate monotonically decreases as the smoothing parameter increases. For many exploratory purposes, this property alone is reason to use only the normal kernel. Minnotte and Scott (1993) proposed graphing the locations of all modes at all bandwidths in the "mode tree." Minnotte (1997) proposed an extension of Silverman's bootstrap test (Silverman 1981) for the number of modes to test individual modes. Software for the ASH, kernel estimates, and the various mode tests may be found on the web; see statlib at www.stat.cmu.edu, for example.

Multivariate extensions of the kernel approach generally rely upon the product kernel. For example, with bivariate data $\{(x_i, y_i),\ i = 1, \ldots, n\}$, the bivariate (product) kernel estimator is

$$\widehat{f}_K(x, y) = \frac{1}{n} \sum_{i=1}^{n} K_{h_x}(x - x_i) K_{h_y}(y - y_i)\,. \tag{19.16}$$

A different smoothing parameter for each variable generally gives sufficient control. A full bivariate normal kernel may be used in special circumstances, effectively adding one additional smoothing parameter in the form of the correlation coefficient. However, an equivalent estimate may be obtained by rotating the data so that the correlation in the kernel vanishes, so that the product kernel may be used on the transformed data.

In higher dimensions, some care must be exercised to minimize the effects of the curse of dimensionality. First, marginal variable transformations should be explored to avoid a heavily skewed appearance or heavy tails. Second, a principal components analysis should be performed to determine if the data are of full rank. If so, the data should be projected into an appropriate subspace. No nonparametric procedure works well if the data are not of full rank. Finally, if the data do not have many significant digits, the data should be carefully blurred. Otherwise the data may have many repeated values, and cross-validation algorithms may believe the data are discrete and suggest using $h = 0$. Next several kernel or ASH estimates may be calculated and explored to gain an understanding of the data, as a preliminary step towards further analyses.

An extensive body of work also exists for orthogonal series density estimators. Originally, the Fourier basis was studied, but more modern choices for the basis functions include wavelets. These can be re-expressed as kernel estimators, so we do not pursue these further. In fact, a number of workers have shown how almost any nonparametric density algorithm can be put into the form of a kernel estimator; see Walter and Blum (1979) and Terrell and Scott (1992), for example. More recent work on local likelihood algorithms for density estimation further shows how closely related parametric and nonparametric thinking really is; see Loader (1999) for details and literature.

19.4 Visualization of Trivariate Functionals

The field of scientific visualization has greatly enhanced the set of tools available for the statistician interested in exploring the features of a density estimate in more than two dimensions. In this section, we demonstrate by example the exploration of trivariate data.

We continue our analysis of the data given by the duration of 299 consecutive eruptions of the Old Faithful geyser. A graph of the histogram of these data is displayed in Fig. 19.2b. We further modified the data as follows: the 105 values that were only recorded to the nearest minute were blurred by adding uniform noise of 30 seconds in duration. (The remaining data points were recorded to the nearest second). An easy way to generate high-dimensional data from a univariate time series is to group adjacent values. In Fig. 19.12, ASH's of the univariate data $\{y_t\}$ and the lagged data $\{(y_{t-1}, y_t)\}$ are shown. The obvious question is whether knowledge of y_{t-1} is useful for predicting the value of y_t. Clearly, the answer is in the affirmative, but the structure would not be well-represented by an autoregressive model.

Next, we computed the ASH for the trivariate lagged data $\{(y_{t-2}, y_{t-1}, y_t)\}$. The resulting estimate, $\widehat{f}_{\mathrm{ASH}}(y_{t-2}, y_{t-1}, y_t)$, may be explored in several fashions. The question is whether knowing y_{t-2} can be used to predict the joint behavior of (y_{t-1}, y_t). This may be accomplished, for example, by examining *slices* of the trivariate density. Since the (univariate) density has two modes at $x = 1.88$ and 4.33 minutes, we examine the slices $\widehat{f}_{\mathrm{ASH}}(1.88, y_{t-1}, y_t)$ and $\widehat{f}_{\mathrm{ASH}}(4.33, y_{t-1}, y_t)$; see Fig. 19.13. The 297 data points were divided into two groups, depending on whether $y_{t-2} < 3.0$ or not. The first group of points was added to Fig. 19.13a, while the second group was added to Fig. 19.13b.

Since each axis was divided into 100 bins, there are 98 other views one might examine like Fig. 19.13. (An animation is actually quite informative.) However, one may obtain a holistic view by examining level sets of the full trivariate density. A level set is the set of all points $\mathbf{x}$ such that $\widehat{f}_{\mathrm{ASH}}(\mathbf{x}) = \alpha \widehat{f}_{\max}$, where $\widehat{f}_{\max}$ is the

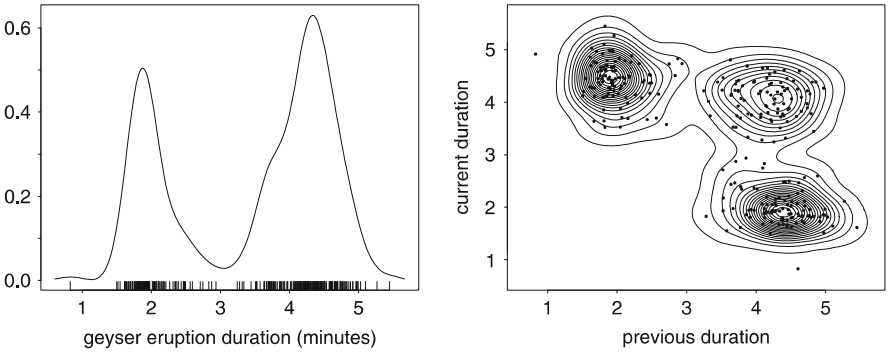


Fig. 19.12 Averaged shifted histograms of the Old Faithful geyser duration data

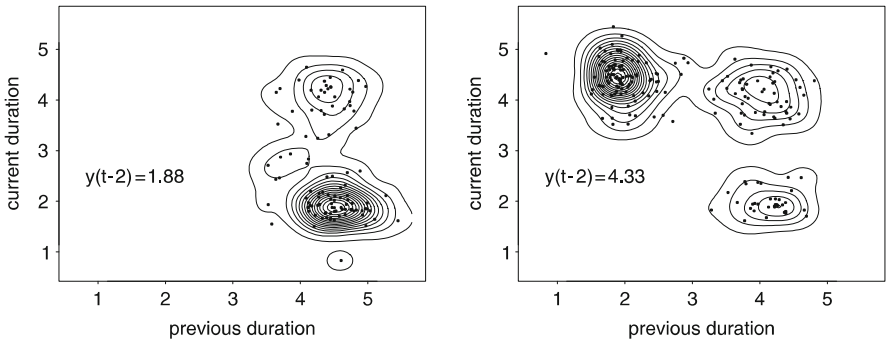
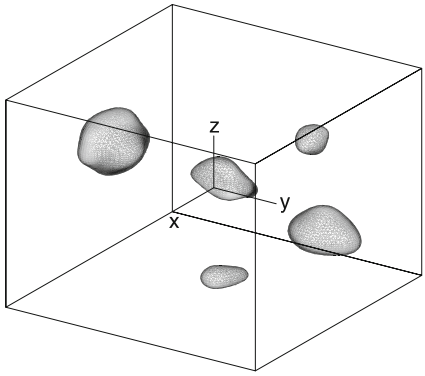


Fig. 19.13 Slices of the trivariate averaged shifted histogram of lagged values of the Old Faithful geyser duration data

Fig. 19.14 Visualization of the $\alpha = 58\%$ contour of the trivariate ASH of the lagged geyser duration data



maximum or modal value of the density estimate, and $\alpha \in (0, 1)$ is a constant that determines the contour level. Such contours are typically smooth surfaces in $\Re^3$. When $\alpha = 1$, then the "contour" is simply the modal location point. In Fig. 19.14, the contour corresponding to $\alpha = 58\%$ is displayed. Clearly these data are multimodal, as five well-separated high-density regions are apparent. Each cluster corresponds to a different sequence of eruption durations, such as long-long-long. The five clusters are now also quite apparent in both frames of Fig. 19.13. Of the eight possible sequences, three are not observed in this sequence of 299 eruptions.

A single contour does not convey as much information as several. Depending on the display device, one may reasonably view three to five contours, using transparency to see the higher density contours that are "inside" the lower density contours. Consider adding a second contour corresponding to $\alpha = 28\%$ to that in Fig. 19.14. Rather than attempt to use transparency, we choose an alternative representation which emphasizes the underlying algorithms. The software which produced these figures is called *ashn* and is available at the author's website. ASH values are computed on a three-dimensional lattice. The surfaces are constructed using the marching cubes algorithm (Lorensen and Cline 1987), which generates

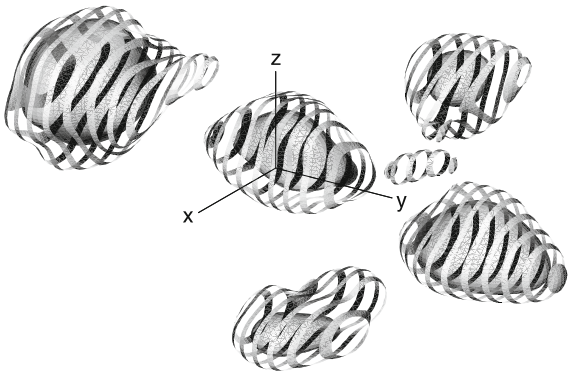


Fig. 19.15 Visualization of the $\alpha = 28\%$ and 58% contours of the trivariate ASH of the lagged geyser duration data

thousands of triangles that make up each surface. In Fig. 19.15, we choose not to plot all of the triangles but only every other "row" along the second axis. The striped effect allows one to interpolate and complete the low-density contour, while allowing one to look inside and see the high-density contour. Since there are five clusters, this is repeated five times. A smaller sixth cluster is suggested as well.

19.5 Conclusions

Exploring data is an important part of successful statistical model building. General discussions of graphical tools may be found in Tufte (1983), Wainer (1997), Cleveland (1985, 1993), Wegman and Depriest (1986) and Buja and Tukey (1991), for example. Advanced exploratory software may be found in many commercial packages, but of special note is the XGobi (Swayne et al. 1991) system and successors. Immersive environments are also of growing interest (Cook et al. 1997). A general visualization overview may be found in Wolff and Yaeger (1993).

Especially when the data size grows, point-oriented methods should be supplemented by indirect visualization techniques based upon nonparametric density estimation or by parallel coordinates (Inselberg 1985; Wegman 1990). Many density algorithms are available. The use of order-two algorithms is generally to be recommended. These should be calibrated by several techniques, starting with an oversmoothed bandwidth and the normal reference rule.

For data beyond three dimensions, density estimates may be computed and slices such as $\widehat{f}(x, y, z, t = t_0)$ visualized. If advanced hardware is available, the surfaces can be animated as t varies continuously over an interval (t_0, t_1); see Scott (1986; 2000). Obviously, this is most useful for data in four and five dimensions. In any case, multivariate density estimation and visualization are important modern tools for EDA.

Acknowledgements This research was supported in part by the National Science Foundation grants NSF EIA-9983459 (digital government) and DMS 09-07491 (non-parametric methodology).

References

Azzalini, A., Bowman, A.W.: A look at some data on the old faithful geyser. Appl. Stat. **39**, 357–365 (1990)

Bowman, A.W.: An alternative method of cross-validation for the smoothing of density estimates. Biometrika **71**, 353–360 (1984)

Bowman, A.W., Azzalini, A.: Applied Smoothing Techniques For Data Analysis: The Kernel Approach With S-Plus Illustrations, Oxford University Press, Oxford (London/Melbourne) (1990)

Buja, A., Tukey, P.A. ed.: Computing and Graphics in Statistics, Springer, New York (1991)

Carr, D.B., Olsen, A.R., White, D.: Hexagon mosaic maps for the display of univariate and bivariate geographical data. Cartograph. Geograph. Inform. Syst. **19**, 228–231 (1992)

Choi, E., Hall, P.: Data sharpening as a prelude to density estimation. Biometrika **86**, 941–947 (1999)

Cleveland, W.S.: The Elements of Graphing Data, Wadsworth, Monterey, CA (1985)

Cleveland, W.S.: Visualizing Data, Hobart Press, Summit, NJ (1993)

Cook, D., Cruz-Neira, C., Kohlmeyer, B.D., Lechner, U., Lewin, N., Elson, L., Olsen, A., Pierson, S., Symanzik, J.: Exploring environmental data in a highly immersive virtual reality environment. Inter. J. Environ. Monit. Assess. **51**(1–2), 441–450 (1997)

Devroye, L.: A Course in Density Estimation, Birkhï£¡user, Boston (1987)

Devroye, L., Györfi, L.: Nonparametric Density Estimation: The L_1 View, John Wiley, New York (1985)

Duda, R.O., Hart, P.E.: Pattern Classification and Scene Analysis, John Wiley, New York (1973)

Hall, P., Sheather, S.J., Jones, M.C., Marron, J.S.: On optimal data-based bandwidth selection in kernel density estimation. Biometrika **78**, 263–270 (1991)

Härdle, W.: Smoothing Techniques With Implementation in S, Springer, New York (1990)

Hazelton, M.: Bandwidth selection for local density estimation. Scand. J. Stat. **23**, 221–232 (1996)

Inselberg, A.: The plane with parallel coordinates. Visual comput. **1**, 69–91 (1985)

Jones, M.C., Marron, J.S., Sheather, S.J.: A brief survey of bandwidth selection for density estimation. J. Am. Stat. Assoc. **91**, 401–407 (1996)

Kronmal, R.A., Tarter, M.E.: The Estimation of Probability Densities and Cumulatives by Fourier Series Methods. J. Amer. Statist. Assoc. **63**, 925–952 (1968)

Loader, C.: Local Regression and Likelihood, Springer, New York (1999)

Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3D surface construction algorithm. Comput. Graph. **21**, 163–169 (1987)

Minnotte, M.C.: Nonparametric testing of the existence of modes. Ann. Stat. 25:1646–1660 (1997)

Minnotte, M.C., Scott, D.W.: The mode tree: A tool for visualization of nonparametric density features. J. Comput. Graph. Stat. **2**, 51–68 (1993)

Nadaraya, E.A., Kotz, S. Translator: Nonparametric Estimation of Probability Densities and Regression Curves, Kluwer Academic Publishers Group, Dordrecht (Hingham, MA) (1989)

Parzen, E.: On estimation of probability density function and mode. Ann. Math. Stat. **33**, 1065–1076 (1962)

Prakasa Rao, B.L.S.: Nonparametric Functional Estimation, Academic Press, Orlando, FL (1983)

Rice, J.A.: Boundary modification for kernel regression. Commun. Stat. **13**, 893–900 (1984)

Rosenblatt, M.: Remarks on some nonparametric estimates of a density function. Ann. Math. Stat. **27**, 832–837 (1956)

Rudemo, M.: Empirical choice of histograms and kernel density estimators. Scand. J. Stat. **9**, 65–78 (1982)

Sain, S.R., Baggerly, K.A., Scott, D.W.: Cross-validation of multivariate densities. J. Am. Stat. Assoc. **89**, 807–817 (1994)

Sain, S.R., Scott, D.W.: On locally adaptive density estimation. J. Am. Stat. Assoc. **91**, 1525–1534 (1996)

Sain, S.R., Scott, D.W.: Zero-Bias bandwidths for locally adaptive kernel density estimation. Scand. J. Stat. **29**, 441–460 (2002)
Scott, D.W.: On Optimal and data-based Histograms. Biometrika **66**, 605–610 (1979)
Scott, D.W.: On optimal and data-based frequency polygons. J. Amer. Stat. Assoc. **80**, 348–354 (1985a)
Scott, D.W.: Averaged shifted histograms: Effective nonparametric density estimators in several dimensions. Ann. Stat. **13**, 1024–1040 (1985b)
Scott, D.W.: Data Analysis in 3 and 4 Dimensions with Nonparametric Density Estimation. In: Wegman, E.J., DePriest, D. (eds.) Statistical Image Processing and Graphics, pp. 291–305. Marcel Dekker, New York (1986)
Scott, D.W.: A note on choice of bivariate histogram bin shape. J. Offic. Stat. **4**, 47–51 (1988)
Scott, D.W.: Multivariate Density Estimation: Theory, Practice, and Visualization. Wiley, New York (1992)
Scott, D.W.: Multidimensional Smoothing and Visualization. In: Schimek, M.G. (eds.) Smoothing and Regression. Approaches, Computation and Application, pp. 451–470. Wiley, New York (2000)
Scott, D.W.: Parametric statistical modeling by minimum integrated square error. Technometrics **43**, 274–285 (2001)
Silverman, B.W.: Using kernel density estimates to investigate multimodality. J. Roy. Stat. Soc. B **43**, 97–99 (1981)
Silverman, B.W.: Algorithm AS176. Kernel density estimation using the fast fourier transform. Appl. Stat. **31**, 93–99 (1982)
Silverman, B.W.: Density Estimation for Statistics and Data Analysis, Chapman and Hall, London (1986)
Silverman, B.W., Jones, M.C.: Fix, E., Hodges, J. L. (1951): An important contribution to nonparametric discriminant analysis and density estimation: Commentary on Fix and Hodges (1951). Int. Stat. Rev. **57**, 233–247 (1989)
Simonoff, J.S.: Smoothing Methods in Statistics, Springer, New York (1996)
Sturges, H.A.: The choice of a class interval. J. Amer. Stat. Assoc. **21**, 65–66 (1926)
Swayne, D., Cook, D., Buja, A.: XGobi: Interactive Dynamic Graphics in the X Window System with a Link to S. ASA Proceedings of the Section on Statistical Graphics, pp. 1–8, ASA, Alexandria, VA (1991)
Tapia, R.A., Thompson, J.R.: Nonparametric Probability Density Estimation, John Hopkins University Press, Baltimore (1978)
Tarter, M.E.: Statistical Curves and Parameters, AK Peters, Natick, MA (2000)
Tarter, M.E., Lock, M.D.: Model-Free Curve Estimation, Chapman & Hall, London (1993)
Taylor, C.C.: Bootstrap choice of the smoothing parameter in kernel density estimation. Biometrika **76**, 705–712 (1989)
Terrell, G.R.: The maximal smoothing principle in density estimation. J. Am. Stat. Assoc. **85**, 470–477 (1990)
Terrell, G.R., Scott, D.W.: On improving convergence rates for nonnegative kernel density estimators. Ann. Stat. **8**, 1160–1163 (1980)
Terrell, G.R., Scott, D.W.: Oversmoothed nonparametric density estimates. J. Am. Stat. Assoc. **80**, 209–214 (1985)
Terrell, G.R., Scott, D.W.: Variable kernel density estimation. Ann. Stat. **20**, 1236–1265 (1992)
Tufte, E.R.: The Visual Display of Quantitative Information, Graphics Press, Cheshire, CT (1983)
Tukey, J.W.: Exploratory Data Analysis, Addison-Wesley, Reading, MA (1977)
Wahba, G.: Data-based optimal smoothing of orthogonal series density estimates. Ann. Stat. **9**, 146–156 (1981)
Wainer, H.: Visual Revelations, Springer, New York (1997)
Walter, G., Blum, J.R.: Probability density estimation using delta sequences. Ann. Stat. **7**, 328–340 (1979)
Wand, M.P., Jones, M.C.: Kernel Smoothing, Chapman & Hall, London (1995)

Wand, M.P., Marron, J.S., Ruppert, D.: Transformations in density estimation J. Am. Stat. Assoc. **86**, 343–353 (1991)
Watson, G.S.: Density estimation by orthogonal series. Ann. Math. Stat. **40**, 1496–1498 (1969)
Wegman, E.J.: Hyperdimensional Data Analysis Using Parallel Coordinates. J. Amer. Stat. Assoc. **85**, 664–675 (1990)
Wegman, E.J., Depriest, D.J.: Statistical Image Processing and Graphics, Marcel Dekker Inc, New York (1986)
Wertz, W.: Statistical Density Estimation: A Survey, Vandenhoeck & Ruprecht, Göttingen (1978)
Wolff, R.S., Yaeger, L.: Visualization Nat. Phenom. Springer, New York (1993)

Chapter 20
Smoothing: Local Regression Techniques

Catherine Loader

Smoothing methods attempt to find functional relationships between different measurements. As in the standard regression setting, the data is assumed to consist of measurements of a response variable, and one or more predictor variables. Standard regression techniques (Chap. III8.) specify a functional form (such as a straight line) to describe the relation between the predictor and response variables. Smoothing methods take a more flexible approach, allowing the data points themselves to determine the form of the fitted curve.

This article begins by describing several different approaches to smoothing, including kernel methods, local regression, spline methods and orthogonal series. A general theory of linear smoothing is presented, which allows us to develop methods for statistical inference, model diagnostics and choice of smoothing parameters.

The theory is then extended to more general settings, including multivariate smoothing and likelihood models.

20.1 Smoothing

Given a dataset consisting of several variables and multiple observations, the goal of smoothing is to construct a functional relationship among the variables.

The most common situation for smoothing is that of a classical regression setting, where one assumes that observations occur in (predictor, response) pairs. That is, the available data has the form

$$\{(x_i, Y_i)\,; \quad i = 1, \ldots, n\}\,,$$

C. Loader (✉)
Department of Statistics, Case Western Reserve University, Cleveland, OH, USA
e-mail: catherine@case.edu

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_20,

where x_i is a measurement of the predictor (or independent) variable, and Y_i is the corresponding response. A functional model relating the variables takes the form

$$Y_i = \mu(x_i) + \epsilon_i \ , \tag{20.1}$$

where $\mu(x_i)$ is the mean function, and ϵ_i is a random error term. In classical regression analysis, one assumes a parametric form for the mean function; for example, $\mu(x) = a_0 + a_1 x$. The problem of estimating the mean function then reduces to estimating the coefficients a_0 and a_1.

The idea of smoothing methods is not to specify a parametric model for the mean function, but to allow the data to determine an appropriate functional form. Loosely stated, one assumes only that the mean function is smooth. Formal mathematical analysis may state the smoothness condition as a bound on derivatives of μ; for example, $|\mu''(x)| \leq M$ for all x and a specified constant M.

Section 20.2 describes some of the most important smoothing methods. These all fall into a class of linear smoothers, and Sect. 20.3 develops important properties, including bias and variance. These results are applied to derive statistical procedures, including bandwidth selection, model diagnostics and goodness-of-fit testing in Sect. 20.4. Multivariate smoothing, when there are multiple predictor variables, is discussed in Sect. 20.5. Finally, Sect. 20.5.2 discusses extensions to likelihood smoothing.

20.2 Linear Smoothing

In this section, some of the most common smoothing methods are introduced and discussed.

20.2.1 Kernel Smoothers

The simplest of smoothing methods is a kernel smoother. A point x is fixed in the domain of the mean function $\mu(\cdot)$, and a smoothing window is defined around that point. Most often, the smoothing window is simply an interval $(x-h, x+h)$, where h is a fixed parameter known as the *bandwidth*.

The kernel estimate is a weighted average of the observations within the smoothing window:

$$\hat{\mu}(x) = \frac{\sum_{i=1}^{n} W\left(\frac{x_i - x}{h}\right) Y_i}{\sum_{j=1}^{n} W\left(\frac{x_j - x}{h}\right)} \ , \tag{20.2}$$

where $W(\cdot)$ is a weight function. The weight function is chosen so that most weight is given to those observations close to the fitting point x. One common choice is the bisquare function,

$$W(x) = \begin{cases} \left(1 - x^2\right)^2 & -1 \le x \le 1 \\ 0 & x > 1 \quad \text{or} \quad x < -1 \end{cases} .$$

The kernel smoother can be represented as

$$\hat{\mu}(x) = \sum_{i=1}^{n} l_i(x) Y_i \ , \tag{20.3}$$

where the coefficients $l_i(x)$ are given by

$$l_i(x) = \frac{W\left(\frac{x_i - x}{h}\right)}{\sum_{j=1}^{n} W\left(\frac{x_j - x}{h}\right)} .$$

A *linear smoother* is a smoother that can be represented in the form (20.3) for appropriately defined weights $l_i(x)$. This linear representation leads to many nice statistical and computational properties, which will be discussed later.

The kernel estimate (20.2) is sometimes called the Nadaraya–Watson estimate (Nadaraya (1964); Watson (1964)). Its simplicity makes it easy to understand and implement, and it is available in many statistical software packages. But its simplicity leads to a number of weaknesses, the most obvious of which is boundary bias. This can be illustrated through an example.

The fuel economy dataset consists of measurements of fuel usage (in miles per gallon) for sixty different vehicles. The predictor variable is the weight (in pounds) of the vehicle. Figure 20.1 shows a scatterplot of the sixty data points, together with a kernel smooth. The smooth is constructed using the bisquare kernel and bandwidth $h = 600$ pounds.

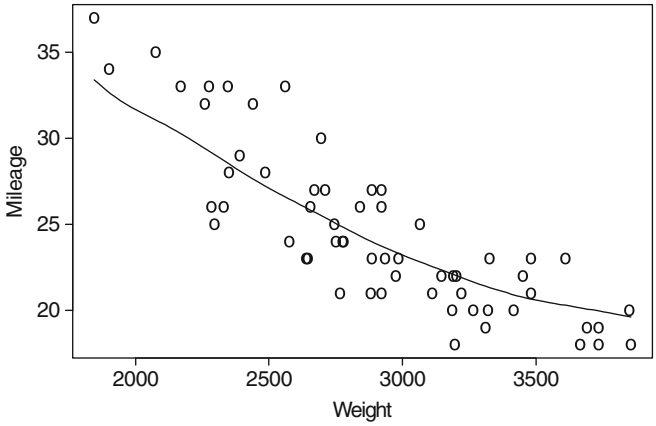


Fig. 20.1 Kernel smooth of the fuel economy dataset. The bisquare kernel is used, with bandwidth $h = 600$ pounds

Over much of the domain of Fig. 20.1, the smooth fit captures the main trend of the data, as required. But consider the left boundary region; in particular, vehicles weighing less than 2200 pounds. All these data points lie *above* the fitted curve; the fitted curve will underestimate the economy of vehicles in this weight range. When the kernel estimate is applied at the left boundary (say, at Weight $=$ 1800), all the data points used to form the average have Weight $>$ 1800, and correspondingly slope of the true relation induces boundary bias into the estimate.

More discussion of this and other weaknesses of the kernel smoother can be found in Hastie and Loader (1993). Many modified kernel estimates have been proposed, but one obtains more parsimonious solutions by considering alternative estimation procedures.

20.2.2 Local Regression

Local regression estimation was independently introduced in several different fields in the late nineteenth and early twentieth century (Henderson (1916); Schiaparelli (1866)). In the statistical literature, the method was independently introduced from different viewpoints in the late 1970's (Cleveland (1979); Katkovnik (1979); Stone (1977)). Books on the topic include Fan and Gijbels (1996) and Loader (1999b).

The underlying principle is that a smooth function can be well approximated by a low degree polynomial in the neighborhood of any point x. For example, a local linear approximation is

$$\mu(x_i) \approx a_0 + a_1(x_i - x) \tag{20.4}$$

for $x - h \le x_i \le x + h$. A local quadratic approximation is

$$\mu(x_i) \approx a_0 + a_1(x_i - x) + \frac{a_2}{2}(x_i - x)^2 .$$

The local approximation can be fitted by locally weighted least squares. A weight function and bandwidth are defined as for kernel regression. In the case of local linear regression, coefficient estimates $\hat{a}_0, \hat{a}_1$ are chosen to minimize

$$\sum_{i=1}^{n} W\left(\frac{x_i - x}{h}\right)(Y_i - (a_0 + a_1(x_i - x)))^2 . \tag{20.5}$$

The local linear regression estimate is defined as

$$\hat{\mu}(x) = \hat{a}_0 . \tag{20.6}$$

Each local least squares problem defines $\hat{\mu}(x)$ at one point x; if x is changed, the smoothing weights $W\left(\frac{x_i - x}{h}\right)$ change, and so the estimates $\hat{a}_0$ and $\hat{a}_1$ change.

Since (20.5) is a weighted least squares problem, one can obtain the coefficient estimates by solving the normal equations

$$\boldsymbol{X}^\top \boldsymbol{W} \left(Y - \boldsymbol{X} \begin{pmatrix} \hat{a}_0 \\ \hat{a}_1 \end{pmatrix} \right) = 0 \,, \tag{20.7}$$

where $\boldsymbol{X}$ is the *design matrix*:

$$\boldsymbol{X} = \begin{pmatrix} 1 & x_1 - x \\ \vdots & \vdots \\ 1 & x_n - x \end{pmatrix}$$

for local linear regression, $\boldsymbol{W}$ is a diagonal matrix with entries $W\left(\frac{x_i - x}{h}\right)$ and $Y = \left(Y_1 \ \ldots \ Y_n\right)^\top$.

When $\boldsymbol{X}^\top \boldsymbol{W}\, \boldsymbol{X}$ is invertible, one has the explicit representation

$$\begin{pmatrix} \hat{a}_0 \\ \hat{a}_1 \end{pmatrix} = \left(\boldsymbol{X}^\top \boldsymbol{W}\, \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{W}\, Y \,. \tag{20.8}$$

This shows that the local regression estimate is a linear estimate, as defined by (20.3). Explicitly, the coefficients $l_i(x)$ are given by

$$l(x)^\top = \left(l_1(x) \ \ldots \ l_n(x)\right) = e_1^\top \left(\boldsymbol{X}^\top \boldsymbol{W}\, \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{W} \,, \tag{20.9}$$

where $e_1^\top$ is the unit vector $\left(1 \ 0\right)$.

For local quadratic regression and higher order fits, one simply adds additional columns to the design matrix $\boldsymbol{X}$ and vector $e_1^\top$.

Figure 20.2 shows a local linear regression fit to the fuel economy dataset. This has clearly fixed the boundary bias problem observed in Fig. 20.1. With

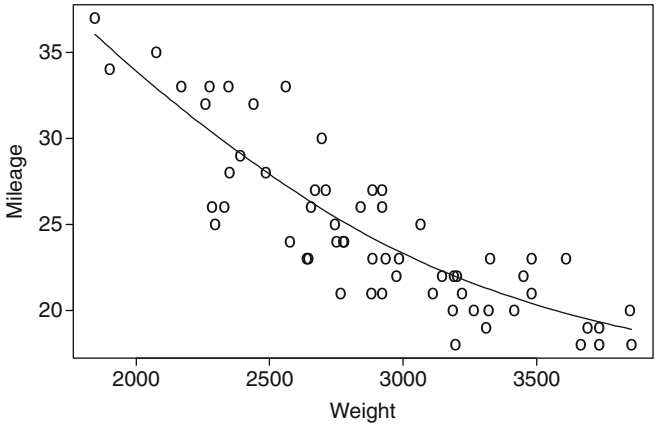


Fig. 20.2 Local Linear Regression fitted to the fuel economy dataset. A bandwidth $h = 1000$ pounds is used

the reduction in boundary bias, it is also possible to substantially increase the bandwidth, from $h = 600$ pounds to $h = 1000$ bounds. As a result, the local linear fit is using much more data, meaning the estimate has less noise.

20.2.3 Penalized Least Squares (Smoothing Splines)

An entirely different approach to smoothing is through optimization of a penalized least squares criterion, such as

$$\sum_{i=1}^{n} (Y_i - \mu(x_i))^2 + \lambda \int \mu''(x)^2 \mathrm{d}x \ , \tag{20.10}$$

where λ is specified constant. This criterion trades off fidelity to the data (measured by the residual sum-of-squares) versus roughness of the mean function (measured by the penalty term). The penalized least squares method chooses $\hat{\mu}$ from the class of twice differentiable functions to minimize the penalized least squares criterion.

The solution to this optimization problem is a piecewise polynomial, or spline function, and so penalized least squares methods are also known as smoothing splines. The idea was first considered in the early twentieth century (Whitaker (1923)). Modern statistical literature on smoothing splines began with work including Wahba and Wold (1975) and Silverman (1985). Books devoted to spline smoothing include Green and Silverman (1994) and Wahba (1990).

Suppose the data are ordered; $x_i \leq x_{i+1}$ for all i. Let $\hat{a}_i = \hat{\mu}(x_i)$, and $\hat{b}_i = \hat{\mu}'(x_i)$, for $i = 1, \ldots, n$. Given these values, it is easy to show that between successive data points, $\hat{\mu}(x)$ must be the unique cubic polynomial interpolating these values:

$$\hat{\mu}(x) = a_i \phi_0(u) + b_i \Delta_i \psi_0(u) + a_{i+1} \phi_1(u) + b_{i+1} \Delta_i \psi_1(u) \ ,$$

where $\Delta_i = x_{i+1} - x_i$; $u = (x - x_i)/\Delta_i$ and

$$\begin{aligned} \phi_0(u) &= 1 - u^2(3 - 2u) \\ \psi_0(u) &= u(1 - u(2 - u)) \\ \phi_1(u) &= u^2(3 - 2u) \\ \psi_1(u) &= u^2(u - 1) \ . \end{aligned}$$

Letting $\alpha^\top = \begin{pmatrix} a_1 & b_1 & \ldots & a_n & b_n \end{pmatrix}$, the penalty term $\int \mu''(x)^2 \mathrm{d}x$ is a quadratic function of the parameters, and so (20.10) can be written as

$$\| Y - \boldsymbol{X}\alpha \|^2 + \lambda \alpha^\top \boldsymbol{M} \alpha \ ,$$

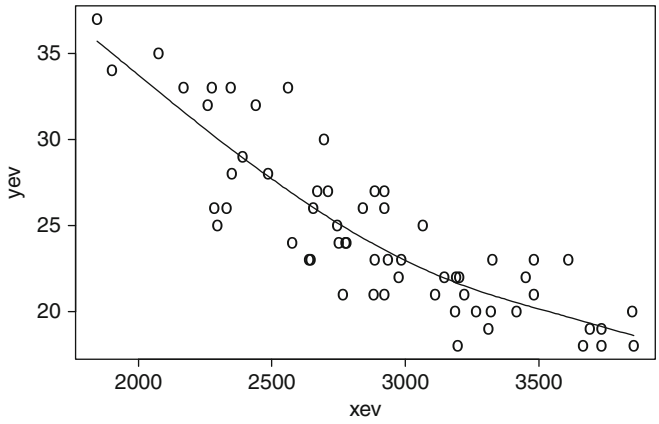


Fig. 20.3 Smoothing Spline fitted to the fuel economy dataset. The penalty is $\lambda = 1.5E8$ $pounds^3$

for appropriate matrices $\boldsymbol{M}$ and $\boldsymbol{X}$. The parameter estimates are given by

$$\widehat{\alpha} = \left(\boldsymbol{X}^\top \boldsymbol{X} + \lambda \boldsymbol{M}\right)^{-1} \boldsymbol{X}^\top Y \ .$$

Figure 20.3 shows a smoothing spline fitted to the fuel economy dataset. Clearly, the fit is very similar to the local regression fit in Fig. 20.2. This situation is common for smoothing problems with a single predictor variable; with comparably chosen smoothing parameters, local regression and smoothing spline methods produce similar results. On the other hand, kernel methods can struggle to produce acceptable results, even on relatively simple datasets.

20.2.4 Regression Splines

Regression splines begin by choosing a set of knots (typically, much smaller than the number of data points), and a set of basis functions spanning a set of piecewise polynomials satisfying continuity and smoothness constraints.

Let the knots be $v_1 < \ldots < v_k$ with $v_1 = \min(x_i)$ and $v_k = \max(x_i)$. A linear spline basis is

$$f_j(x) = \begin{cases} \dfrac{x - v_{j-1}}{v_j - v_{j-1}} & v_{j-1} \leq x \leq v_j \\ \dfrac{v_{j+1} - x}{v_{j+1} - v_j} & v_j < x \leq v_{j+1} \\ 0 & \text{otherwise} \end{cases} ;$$

note that these functions span the space of piecewise linear functions with knots at $v_1, \ldots, v_k$. The piecewise linear spline function is constructed by regressing the data onto these basis functions.

The linear spline basis functions have discontinuous derivatives, and so the resulting fit may have a jagged appearance. It is more common to use piecewise cubic splines, with the basis functions having two continuous derivatives. See Chap. 3 of Ruppert et al. (2003) for a more detailed discussion of regression splines and basis functions.

20.2.5 Orthogonal Series

Orthogonal series methods represent the data with respect to a series of orthogonal basis functions, such as sines and cosines. Only the low frequency terms are retained. The book Efromovich (1999) provides a detailed discussion of this approach to smoothing.

Suppose the x_i are equally spaced; $x_i = i/n$. Consider the basis functions

$$f_\omega(x) = a_\omega \cos(2\pi\omega x) \; ; \quad \omega = 0, 1, \ldots, \lfloor n/2 \rfloor$$

$$g_\omega(x) = b_\omega \sin(2\pi\omega x) \; ; \quad \omega = 1, \ldots, \lfloor (n-1)/2 \rfloor \; ,$$

where the constants a_ω, b_ω are chosen so that $\sum_{i=1}^n f_\omega(x_i)^2 = \sum_{i=1}^n g_\omega(x_i)^2 = 1$. Then the regression coefficients are

$$c_\omega = \sum_{i=1}^n f_\omega(x_i) Y_i$$

$$s_\omega = \sum_{i=1}^n g_\omega(x_i) Y_i$$

and the corresponding smooth estimate is

$$\hat{\mu}(x) = \sum_\omega h(\omega) \left(c_\omega f_\omega(x) + s_\omega g_\omega(x) \right) \; .$$

Here, $h(\omega)$ is chosen to 'damp' high frequencies in the observations; for example,

$$h(\omega) = \begin{cases} 1 & \omega \le \omega_0 \\ 0 & \omega > \omega_0 \end{cases}$$

is a low-pass filter, passing all frequencies less than or equal to ω_0.

Orthogonal series are widely used to model time series, where the coefficients c_ω and s_ω may have a physical interpretation: non-zero coefficients indicate the presence of cycles in the data. A limitation of orthogonal series approaches is that they are more difficult to apply when the x_i are not equally spaced.

20.3 Statistical Properties of Linear Smoothers

Each of the smoothing methods discussed in the previous section has one or more 'smoothing parameters' that control the amount of smoothing being performed. For example, the bandwidth h in the kernel smoother or local regression methods, and the parameter λ in the penalized likelihood criterion. In implementing the smoothers, the first question to be asked is how should the smoothing parameters be chosen? More generally, how can the performance of a smoother with given smoothing parameters be assessed? A deeper question is in comparing fits from different smoothers. For example, we have seen for the fuel economy dataset that a local linear fit with $h = 1000$ (Fig. 20.2) produces a fit similar to a smoothing spline with $\lambda = 1.5 \times 10^8$ (Fig. 20.3). Somehow, we want to be able to say these two smoothing parameters are equivalent.

As a prelude to studying methods for bandwidth selection and other statistical inference procedures, we must first study some of the properties of linear smoothers. We can consider measures of goodness-of-fit, such as the mean squared error,

$$\text{MSE}(x) = E\left((\hat{\mu}(x) - \mu(x))^2\right) = \text{var}\left(\hat{\mu}(x)\right) + \text{bias}\left(\hat{\mu}(x)\right)^2 ,$$

where $\text{bias}(\hat{\mu}(x)) = E(\hat{\mu}(x)) - \mu(x)$.

Intuitively, as the bandwidth h increases, more data is used to construct the estimate $\hat{\mu}(x)$, and so the variance $\text{var}(\hat{\mu}(x))$ decreases. On the other hand, the local polynomial approximation is best over small intervals, so we expect the bias to increase as the bandwidth increases. Choosing h is a tradeoff between small bias and small variance, but we need more precise characterizations to derive and study selection procedures.

20.3.1 Bias

The bias of a linear smoother is given by

$$E(\hat{\mu}(x)) - \mu(x) = \sum_{i=1}^{n} l_i(x) E(Y_i) - \mu(x) = \sum_{i=1}^{n} l_i(x)\mu(x_i) - \mu(x) . \quad (20.11)$$

As this depends on the unknown mean function $\mu(x)$, it is not very useful by itself, although it may be possible to estimate the bias by substituting an estimate for $\mu(x)$. To gain more insight, approximations to the bias are derived. The basic tools are

1. A low order Taylor series expansion of $\mu(\cdot)$ around the fitting point x.
2. Approximation of the sums by integrals.

For illustration, consider the bias of the local linear regression estimate defined by (20.6). A three-term Taylor series gives

$$\mu(x_i) = \mu(x) + (x_i - x)\mu'(x) + \frac{(x_i - x)^2}{2}\mu''(x) + o\left(h^2\right)$$

for $|x_i - x| \le h$. Substituting this into (20.11) gives

$$E(\hat{\mu}(x)) - \mu(x) = \mu(x)\sum_{i=1}^{n} l_i(x) + \mu'(x)\sum_{i=1}^{n}(x_i - x)l_i(x) + \frac{\mu''(x)}{2}\sum_{i=1}^{n}(x_i - x)^2 l_i(x) - \mu(x) + o\left(h^2\right) .$$

For local linear regression, it can be shown that

$$\sum_{i=1}^{n} l_i(x) = 1$$

$$\sum_{i=1}^{n}(x_i - x)l_i(x) = 0 .$$

This is a mathematical statement of the heuristically obvious property of the local linear regression: if data Y_i fall on a straight line, the local linear regression will reproduce that line. See Loader (1999b), p. 37, for a formal proof. With this simplification, the bias reduces to

$$E(\hat{\mu}(x)) - \mu(x) = \frac{\mu''(x)}{2}\sum_{i=1}^{n}(x_i - x)^2 l_i(x) + o\left(h^2\right) . \tag{20.12}$$

This expression characterizes the dependence of the bias on the mean function: the dominant term of the bias is proportional to the second derivative of the mean function.

The next step is to approximate summations by integrals, both in (20.12) and in the matrix equation (20.9) defining $l_i(x)$. This leads to

$$E(\hat{\mu}(x)) - \mu(x) \approx \mu''(x)h^2\frac{\int v^2 W(v)\mathrm{d}v}{2\int W(v)\mathrm{d}v} . \tag{20.13}$$

In addition to the dependence on $\mu''(x)$, we now see the dependence on h: as the bandwidth h increases, the bias increases quadratically with the bandwidth.

Bias expansions like (20.13) are derived much more generally by Ruppert and Wand (1994); their results cover arbitrary degree local polynomials and multidimensional fits also. Their results imply that when p, the degree of the local polynomial, is odd, the dominant term of the bias is proportional to $h^{p+1}\mu^{(p+1)}(x)$. When p is even, the first-order term can disappear, leading to bias of order h^{p+2}.

20.3.2 Variance

To derive the variance of a linear smoother, we need to make assumptions about the random errors ϵ_i in (20.1). The most common assumption is that the errors are independent and identically distributed, with variance $\text{var}(\epsilon_i) = \sigma^2$. The variance of a linear smoother (20.3) is

$$\text{var}(\hat{\mu}(x)) = \sum_{i=1}^{n} l_i(x)^2 \text{var}(Y_i) = \sigma^2 \|l(x)\|^2 \,. \tag{20.14}$$

As with bias, informative approximations to the variance can be derived by replacing sums by integrals. For local linear regression, this leads to

$$\text{var}(\hat{\mu}(x)) \approx \frac{\sigma^2}{nhf(x)} \frac{\int W(v)^2 \mathrm{d}v}{\left(\int W(v)\mathrm{d}v\right)^2} \,, \tag{20.15}$$

where $f(x)$ is the density of the design points x_i. The dependence on the sample size, bandwidth and design density through $1/(nhf(x))$ is universal, holding for any degree of local polynomial. The term depending on the weight function varies according to the degree of local polynomial, but generally increases as the degree of the polynomials increases. See Ruppert and Wand (1994) for details.

20.3.3 Degrees of Freedom

Under the model (20.1) the observation Y_i has variance σ^2, while the estimate $\hat{\mu}(x_i)$ has variance $\sigma^2\|l(x_i)\|^2$. The quantity $\|l(x_i)\|^2$ measures the variance reduction of the smoother at a data point x_i. At one extreme, if the 'smoother' interpolates the data, then $\hat{\mu}(x_i) = Y_i$ and $\|l(x_i)\|^2 = 1$. At the other extreme, if $\hat{\mu}(x_i) = \bar{Y}$, $\|l(x_i)\|^2 = 1/n$. Under mild conditions on the weight function, a local polynomial smoother satisfies

$$\frac{1}{n} \le \|l(x_i)\|^2 \le 1 \,,$$

and $\|l(x_i)\|^2$ is usually a decreasing function of the bandwidth h.

A global measure of the amount of smoothing is provided by

$$\nu_2 = \sum_{i=1}^{n} \|l(x_i)\|^2 \ .$$

This is one definition of the 'degrees of freedom' or 'effective number of parameters' of the smoother. It satisfies the inequalities

$$1 \leq \nu_2 \leq n \ .$$

An alternative representation of ν_2 is as follows. Let $\boldsymbol{H}$ be the 'hat matrix', which maps the data to fitted values:

$$\begin{pmatrix} \hat{\mu}(x_1) \\ \vdots \\ \hat{\mu}(x_n) \end{pmatrix} = \boldsymbol{H}Y \ .$$

For a linear smoother, $\boldsymbol{H}$ has rows $l(x_i)^\top$, and $\nu_2 = \text{trace}(\boldsymbol{H}^\top \boldsymbol{H})$.

The diagonal elements of $\boldsymbol{H}$, $l_i(x_i)$ provide another measure of the amount of smoothing at x_i. If the smooth interpolates the data, then $l(x_i)$ is the corresponding unit vector with $l_i(x_i) = 1$. If the smooth is simply the global average, $l_i(x_i) = 1/n$. The corresponding definition of degrees of freedom is

$$\nu_1 = \sum_{i=1}^{n} l_i(x_i) = \text{trace}(\boldsymbol{H}) \ .$$

For a least-squares fit, the hat matrix is a perpendicular projection operator, which is symmetric and idempotent. In this case, $\boldsymbol{H} = \boldsymbol{H}^\top \boldsymbol{H}$, and $\nu_1 = \nu_2$. For linear smoothers, the two definitions of degrees-of-freedom are usually not equal, but they are often of similar magnitude.

For the local linear regression in Fig. 20.2, the degrees of freedom are $\nu_1 = 3.54$ and $\nu_2 = 3.09$. For the smoothing spline smoother in Fig. 20.3, $\nu_1 = 3.66$ and $\nu_2 = 2.98$. By either measure the degrees of freedom are similar for the two fits. The degrees of freedom provides a mechanism by which different smoothers, with different smoothing parameters, can be compared: we simply choose smoothing parameters producing the same number of degrees of freedom. More extensive discussion of the degrees of freedom of a smoother can be found in Cleveland and Devlin (1988) and Hastie and Tibshirani (1990).

Variance Estimation.

The final component needed for many statistical procedures is an estimate of the error variance σ^2. One such estimate is

$$\hat{\sigma}^2 = \frac{1}{n - 2\nu_1 + \nu_2} \sum_{i=1}^{n} (Y_i - \hat{\mu}(x_i))^2 \,. \tag{20.16}$$

The normalizing constant is chosen so that if the bias of $\hat{\mu}(x_i)$ is neglected, $\hat{\sigma}^2$ is unbiased. See Cleveland and Devlin (1988).

20.4 Statistics for Linear Smoothers: Bandwidth Selection and Inference

We also want to perform statistical inference based on the smoothers. As for parametric regression, we want to construct confidence bands and prediction intervals based on the smooth curve. Given a new car that weighs 2800 pounds, what is its fuel economy? Tests of hypotheses can also be posed: for example, is the curvature observed in Fig. 20.2 significant, or would a linear regression be adequate? Given different classifications of car (compact, sporty, minivan etc.) is there differences among the categories that cannot be explained by weight alone?

20.4.1 Choosing Smoothing Parameters

All smoothing methods have one or more smoothing parameters: parameters that control the 'amount' of smoothing being performed. For example, the bandwidth h in the kernel and local regression estimates. Typically, bandwidth selection methods are based on an estimate of some goodness-of-fit criterion. Bandwidth selection is a special case of model selection, discussed more deeply in Chap. III.1.

How should smoothing parameters be used? At one extreme, there is full automation: optimization of the goodness-of-fit criterion produces a single 'best' bandwidth. At the other extreme is purely exploratory and graphical methods, using goodness-of-fit as a guide to help choose the best method.

Automation has the advantage that it requires much less work; a computer can be programmed to perform the optimization. But the price is a lack of reliability: fits with very different bandwidths can produce similar values of the goodness-of-fit criterion. The result is either high variability (producing fits that look undersmoothed) or high bias (producing fits that miss obvious features in the data).

Cross Validation.

Cross validation (CV) focuses on the prediction problem: if the fitted regression curve is used to predict new observations, how good will the prediction be? If a new

observation is made at $x = x_0$, and the response Y_0 is predicted by $\widehat{Y}_0 = \hat{\mu}(x_0)$, what is the prediction error? One measure is

$$E((Y_0 - \widehat{Y}_0)^2) \, .$$

The method of CV can be used to estimate this quantity. In turn, each observation (x_i, Y_i) is omitted from the dataset, and is 'predicted' by smoothing the remaining $n - 1$ observations. This leads to the CV score

$$\mathrm{CV}(\hat{\mu}) = \frac{1}{n} \sum_{i=1}^{n} (Y_i - \hat{\mu}_{-i}(x_i))^2 \, , \tag{20.17}$$

where $\hat{\mu}_{-i}(\cdot)$ denotes the smoothed estimate when the single data point (x_i, Y_i) are omitted from the dataset; only the remaining $n - 1$ data points are used to compute the estimate.

Formally computing each of the leave-one-out regression estimates $\hat{\mu}_{-i}(\cdot)$ would be highly computational, and so at a first glance computation of the CV score (20.17) looks prohibitively expensive. But there is a remarkable simplification, valid for nearly all common linear smoothers (and all those discussed in Sect. 20.2):

$$\hat{\mu}_{-i}(x_i) = \frac{\hat{\mu}(x_i) - l_i(x_i) Y_i}{1 - l_i(x_i)} \, .$$

With this simplification, the CV criterion becomes

$$\mathrm{CV}(\hat{\mu}) = \frac{1}{n} \sum_{i=1}^{n} \frac{(Y_i - \hat{\mu}(x_i))^2}{(1 - l_i(x_i))^2} \, .$$

Generalized cross validation (GCV) replaces each of the influence values $l_i(x_i)$ by the average, ν_1 / n. This leads to

$$\mathrm{GCV}(\hat{\mu}) = n \frac{\sum_{i=1}^{n} (Y_i - \hat{\mu}(x_i))^2}{(n - \nu_1)^2} \, .$$

Figure 20.4 shows the GCV scores for the fuel economy dataset, and using kernel and local linear smoothers with a range of bandwidths. Note the construction of the plot: the fitted degrees of freedom ν_1 are used as the x axis. This allows us to meaningfully superimpose and compare the GCV curves arising from different smoothing methods. From right to left, the points marked '0' represent a kernel smoother with $h = 300, 400, 500, 600, 800$ and 1000, and points marked '1' represent a local linear smoother with $h = 400, 500, 700, 1{,}000, 1{,}500, 2{,}000$ and ∞.

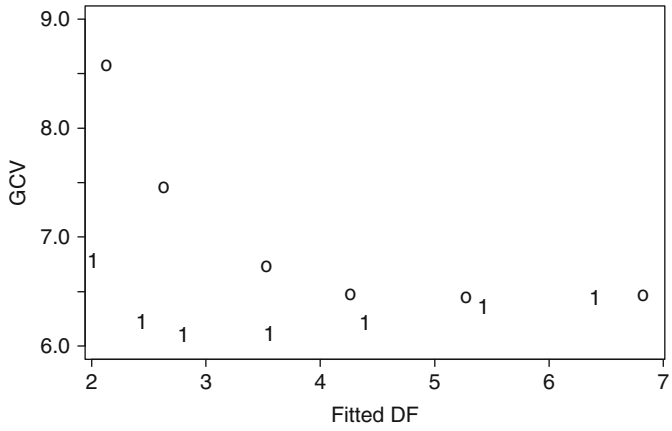


Fig. 20.4 GCV scores for the fuel economy dataset. Points marked 0 are for kernel smoothers with a range of bandwidths h, and points marked 1 are for a local linear smoother

The interpretation of Fig. 20.4 is that for any fixed degrees of freedom, the local linear fit outperforms the kernel fit. The best fits obtained are the local linear, with 3 to 3.5 degrees of freedom, or h between 1000 and 1500.

Unbiased Risk Estimation.

A risk function measures the distance between the true regression function and the estimate; for example,

$$R(\mu, \hat{\mu}) = \frac{1}{\sigma^2} \sum_{i=1}^{n} E\left((\hat{\mu}(x_i) - \mu(x_i))^2\right) . \tag{20.18}$$

Ideally, a good estimate would be one with low risk. But since μ is unknown, $R(\mu, \hat{\mu})$ cannot be evaluated directly.

Instead, the risk must be estimated. An unbiased estimate is

$$\hat{R}(\mu, \hat{\mu}) = \frac{1}{\sigma^2} \sum_{i=1}^{n} (Y_i - \hat{\mu}(x_i))^2 - n + 2\nu_1$$

(Mallows (1973); Cleveland and Devlin (1988)). The unbiased risk estimate is equivalent to Akaike's Information Criterion (Akaike (1972), 1974). To implement the unbiased risk estimate one needs to substitute an estimate for σ^2; Cleveland and Devlin recommend using (20.16) with a small bandwidth.

The unbiased risk estimate can be used similarly to GDV. One computes $\hat{R}(\mu, \hat{\mu})$ for a range of different fits $\hat{\mu}$, and plots the resulting risk estimates versus the degrees of freedom. Fits producing a small risk estimate are considered best.

Bias Estimation and Plug-in Methods.

An entirely different class of bandwidth selection methods, often termed plug-in methods, attempt to directly estimate a risk measure by estimating the bias and variance. The method has been developed mostly in the context of kernel density estimation, but adaptations to kernel regression and local polynomial regression can be found in Fan and Gijbels (1995) and Ruppert et al. (1995).

Again focusing on the squared-error risk, we have the bias-variance decomposition

$$\begin{aligned}\sigma^2 R(\mu, \hat{\mu}) &= \sum_{i=1}^{n} \operatorname{bias}(\hat{\mu}(x_i))^2 + \sum_{i=1}^{n} \operatorname{var}(\hat{\mu}(x_i)) \\ &= \sum_{i=1}^{n} \left(\sum_{j=1}^{n} l_j(x_i)\mu(x_j) - \mu(x_i) \right)^2 + \sigma^2 \sum_{i=1}^{n} \|l(x_i)\|^2 . \qquad (20.19)\end{aligned}$$

A plug-in estimate begins by constructing a preliminary *pilot estimate* of the mean function $\mu(\cdot)$. This is then substituted into the risk estimate (20.19), which can then be minimized over the bandwidth h.

There are many variants of the plug-in idea in the statistics literature. Most simplify the risk function using asymptotic approximations such as (20.13) and (20.15) for the bias and variance; making these substitutions in (20.19) gives

$$\sigma^2 R(\mu, \hat{\mu}) \approx h^4 \left(\frac{\int v^2 W(v) \mathrm{d}v}{2 \int W(v) \mathrm{d}v} \right)^2 \sum_{i=1}^{n} \mu''(x_i)^2 + \frac{\sigma^2}{nh} \frac{\int W(v)^2 \mathrm{d}v}{\left(\int W(v) \mathrm{d}v \right)^2} \sum_{i=1}^{n} \frac{1}{f(x_i)} .$$

If the design points are uniformly distributed on an interval $[a, b]$ say, then approximating the sums by integrals gives

$$\sigma^2 R(\mu, \hat{\mu}) \approx nh^4 \left(\frac{\int v^2 W(v) \mathrm{d}v}{2 \int W(v) \mathrm{d}v} \right)^2 \frac{1}{b-a} \int_a^b \mu''(x)^2 \mathrm{d}x + \frac{(b-a)\sigma^2}{h} \frac{\int W(v)^2 \mathrm{d}v}{\left(\int W(v) \mathrm{d}v \right)^2} .$$

Minimizing this expression over h yields an asymptotically optimal bandwidth:

$$h_{\mathrm{opt}}^5 = \frac{\sigma^2 (b-a)^2 \int W(v)^2 \mathrm{d}v}{n \left(\int v^2 W(v) \mathrm{d}v \right)^2 \int_a^b \mu''(x)^2 \mathrm{d}x} .$$

Evaluation of h_{opt} requires substitution of estimates for $\int_a^b \mu''(x)^2 \mathrm{d}x$ and of σ^2. The estimate (20.16) can be used to estimate σ^2, but estimating $\int_a^b \mu''(x)^2 \mathrm{d}x$ is more problematic. One technique is to estimate the second derivative using a 'pilot' estimate of the smooth, and then use the estimate

$$\int_a^b \hat{\mu}''(x)^2 \mathrm{d}x \ .$$

If a local quadratic estimate is used at the pilot stage, the curvature coefficient $\hat{a}_2$ can be used as an estimate of $\mu''(x)$.

But the use of a pilot estimate to estimate the second derivative is problematic. The pilot estimate itself has a bandwidth that has to be selected, and the estimated optimal bandwidth $\hat{h}_{\text{opt}}$ is highly sensitive to the choice of pilot bandwidth. Roughly, if the pilot estimate smooths out important features of μ, so will the estimate $\hat{\mu}$ with bandwidth $\hat{h}_{\text{opt}}$. More discussion of this point may be found in Loader (1999a).

20.4.2 Normal-based Inference

Inferential procedures for smoothers include the construction of confidence bands for the true mean function, and procedures to test the adequacy of simpler models. In this section, some of the main ideas are briefly introduced; more extensive discussion can be found in the books Azzalini and Bowman (1997), Hardle (1990), Hart (1997) and Loader (1999b).

Confidence Intervals.

If the errors ϵ_i are normally distributed, then confidence intervals for the true mean can be constructed as

$$\hat{\mu}(x) \pm c\hat{\sigma} \|l(x)\| \ .$$

The constant c can be chosen from the Student's t distribution with degrees of freedom equal to $n - 2\nu_1 + \nu_2$ (alternative choices are discussed below in the context of testing). These confidence intervals are pointwise intervals for $E(\hat{\mu}(x))$:

$$P\left(|\hat{\mu}(x) - E(\hat{\mu}(x))| < c\hat{\sigma}\|l(x)\|\right) = 1 - \alpha \ .$$

To construct confidence intervals for $\mu(x)$, one must either choose the bandwidth sufficiently small so that the bias can be ignored, or explicitly estimate the bias. The latter approach suffers from the same weaknesses observed in plug-in bandwidth selection.

Tests of Hypothesis

Consider the problem of testing for the adequacy of a linear model. For example, in the fuel economy dataset of Figs. 20.1 and 20.2, one may be interested in knowing whether a linear regression, $\mu(x) = a + bx$ is adequate, or alternatively whether the departure from linearity indicated by the smooth is significant. This hypothesis testing problem can be stated as

$$H_0 : \mu(x) = a + bx \quad \text{for some} \quad a, b$$
$$H_1 : \text{otherwise} .$$

In analogy with the theory of linear models, an F ratio can be formed by fitting both the null and alternative models, and considering the difference between the fits. Under the null model, parametric least squares is used; the corresponding fitted values are $\boldsymbol{M}Y$ where $\boldsymbol{M}$ is the hat matrix for the least squares fit. Under the alternative model, the fitted values are $\boldsymbol{H}Y$, where $\boldsymbol{H}$ is the hat matrix for a local linear regression. An F ratio can then be formed as

$$F = \frac{\|\boldsymbol{H}Y - \boldsymbol{M}Y\|^2/\nu}{\hat{\sigma}^2} ,$$

where $\nu = \text{trace}((\boldsymbol{H} - \boldsymbol{M})^\top(\boldsymbol{H} - \boldsymbol{M}))$.

What is the distribution of F when H_0 is true? Since $\boldsymbol{H}$ is not a perpendicular projection operator, the numerator does not have a χ^2 distribution, and F does not have an exact F distribution. None-the-less, we can use an approximating F distribution. Based on a one-moment approximation, the degrees of freedom are ν and $n - 2\nu_1 + \nu_2$.

Better approximations are obtained using the two-moment Satterwaite approximation, as described in Cleveland and Devlin (1988). This method matches both the mean and variance of chi-square approximations to the numerator and denominator. Letting $\Lambda = (\boldsymbol{H} - \boldsymbol{M})^\top(\boldsymbol{H} - \boldsymbol{M})$, the numerator degrees of freedom for the F distribution are given by $\text{trace}(\Lambda)^2/\text{trace}(\Lambda^2)$. A similar adjustment is made to the denominator degrees of freedom. Simulations reported in Cleveland and Devlin (1988) suggest the two-moment approximation is adequate for setting critical values.

For the fuel economy dataset, we obtain $F = 7.247$, $\nu = 1.0866$ and $n - 2\nu_1 + \nu_2 = 55.997$. Using the one-moment approximation, the p-value is 0.0079. The two-moment approximation gives a p-value of 0.0019. Both methods indicate that the nonlinearity is significant, although there is some discrepancy between the P-values.

20.4.3 Bootstrapping

The F-tests in the previous section are approximate, even when the errors ϵ_i are normally distributed. Additionally, the degrees-of-freedom computations (particularly for the two-moment approximation) require $O(n^3)$ computations, which is prohibitively expensive for n more than a few hundred.

An alternative to the F approximations is to simulate the null distribution of the F ratio. A bootstrap method (Chap. III.2) performs the simulations using the empirical residuals to approximate the true error distribution:

- Let $r_i = Y_i - \hat{\mu}(x_i)$.
- Resample: $Y_i^* = \hat{\mu}(x_i) + \epsilon_i^*, i = 1, \ldots, n$, where ϵ_i^* is drawn from $r_1, \ldots, r_n$.
- Compute the F statistic based on the resampled data:

$$F^* = \frac{\|\boldsymbol{H}Y^* - \boldsymbol{M}Y^*\|^2/\nu}{(\hat{\sigma}^*)^2} .$$

This procedure is repeated a large number of times (say $B = 1000$) and tabulation of the resulting F^* values provides an estimate of the true distribution of the F ratio.

Remark. Since the degrees of freedom do not change with the replication, there is no need to actually compute the normalizing constant. Instead, one can simply work with the modified F ratio,

$$F_B = \frac{\|\boldsymbol{H}Y^* - \boldsymbol{M}Y^*\|^2}{\|(I - \boldsymbol{H})Y^*\|^2} .$$

Figure 20.5 compares the bootstrap distribution of the F ratio and the 1 and 2 moment F approximations for the fuel economy dataset. The bootstrap method uses 10000 bootstrap replications, and the density is estimated using the Local Likelihood method (Sect. 20.5.2 below). Except at the left end-point, there is generally good agreement between the bootstrap density and the two-moment density. The upper 5% quantiles are 3.21 based on the two-moment approximation, and 3.30 based on the bootstrap sample. The one-moment approximation has a critical value of 3.90. Based on the observed $F = 7.248$, the bootstrap p-value is 0.0023, again in close agreement with the two-moment method.

20.5 Multivariate Smoothers

When there are multiple predictor variables, the smoothing problem becomes multivariate: $\mu(x)$ is now a surface. The definition of kernel and local regression smoothers can be extended to estimate a regression surface with any number of

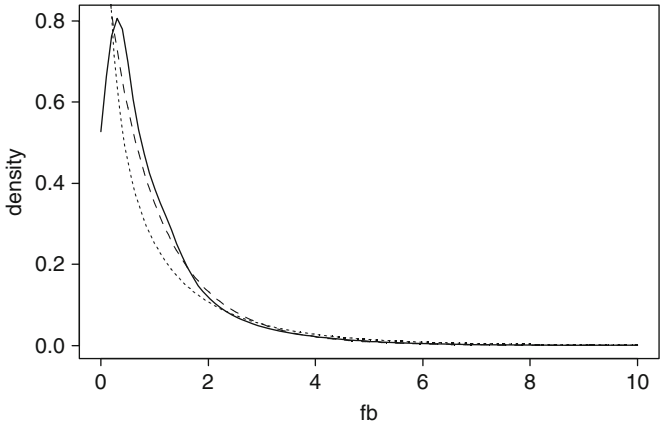


Fig. 20.5 Estimated density of the F ratio, based on the bootstrap method (*solid line*); 1-moment F approximation (*short dashed line*) and 2-moment F approximation (*long dashed line*)

predictor variables, although the methods become less useful for more than 2 or 3 variables. There are several reasons for this:

- Data sparsity – the curse of dimensionality.
- Visualization issues – how does one view and interpret a high dimensional smooth regression surface?
- Computation is often much more expensive in high dimensions.

For these reasons, use of local polynomials and other smoothers to model high dimensional surfaces is rarely recommended, and the presentation here is restricted to the two-dimensional case. In higher dimensions, smoothers can be used in conjunction with dimension reduction procedures (Chap. III.6), which attempt to model the high-dimensional surface through low-dimensional components. Examples of this type of procedure include Projection Pursuit (Friedman and Stuetzle (1981)), Additive Models (Hastie and Tibshirani (1990)), Semiparametric Models (Ruppert et al. (2003) and Chap. III.10) and recursive partitioning (Chap. III.14).

20.5.1 Two Predictor Variables

Suppose the dataset consists of n vectors (u_i, v_i, Y_i), where u_i and v_i are considered predictor variables, and Y_i is the response. For simplicity, we'll use $x_i = \begin{pmatrix} u_i & v_i \end{pmatrix}^\top$ to denote a vector of the predictor variables. The data are modeled as

$$Y_i = \mu(u_i, v_i) + \epsilon_i = \mu(x_i) + \epsilon_i \ .$$

Bivariate smoothers attempt to estimate the surface $\mu(u_i, v_i)$. Kernel and local regression methods can be extended to the bivariate case, simply by defining smoothing weights on a plane rather than on a line. Formally, a bivariate local regression estimate at a point $x = (u, v)^\top$ can be constructed as follows:

1. Define a distance measure $\rho(x, x_i)$ between the data points and fitting point. A common choice is Euclidean distance,

$$\rho(x, x_i) = \sqrt{(u_i - u)^2 + (v_i - v)^2} \, .$$

2. Define the smoothing weights using a kernel function and bandwidth:

$$w_i(x) = W\left(\frac{\rho(x, x_i)}{h}\right) \, .$$

3. Define a local polynomial approximation, such as a local linear approximation

$$\mu(u_i, v_i) \approx a_0 + a_1(u_i - u) + a_2(v_i - v)$$

when (u_i, v_i) is close to (u, v). More generally, a local polynomial approximation can be written

$$\mu(x_i) \approx \langle a, A(x_i - x) \rangle \, ,$$

where a is a vector of coefficients, and $A(\cdot)$ is a vector of basis polynomials.

4. Estimate the coefficient vector by local least squares. That is, choose $\hat{a}$ to minimize

$$\sum_{i=1}^{n} w_i(x) \left(Y_i - \langle a, A(x_i - x) \rangle\right)^2 \, .$$

5. The local polynomial estimate is then

$$\hat{\mu}(x) = \hat{a}_0 \, .$$

20.5.2 Likelihood Smoothing

A likelihood smoother replaces the model (20.1) with a distributional assumption

$$Y_i \sim f(y, \mu_i) \, ,$$

where $f(y, \mu)$ is a specified family of densities, parameterized so that $E(Y_i) = \mu_i$. The family may be chosen depending on the response variable. If Y_i is a count, then the Poisson family is a natural choice:

$$f(y,\mu) = \frac{\mu^y \mathrm{e}^{-\mu}}{y!}\ ; \quad y = 0, 1, 2, \ldots\ .$$

If Y_i is a 0/1 (or no/yes) response, then the Bernoulli family is appropriate:

$$f(y,\mu) = \mu^y (1-\mu)^{1-y}\ ; \quad y = 0, 1\ .$$

Given the data, the log-likelihood is

$$\mathcal{L}(\mu_1, \ldots, \mu_n) = \sum_{i=1}^{n} \log\ f(Y_i, \mu_i)\ .$$

The goal is to estimate the mean function, $\mu_i = \mu(x_i)$ for an observed set of covariates x_i. A generalized linear model (Chap. III.7) uses a parametric model for the mean function. Likelihood smoothers assume only that the mean is a smooth function of the covariates.

The earliest work on likelihood smoothing is Henderson (1924), who used a penalized binomial likelihood to estimate mortality rates. The local likelihood method described below can be viewed as an extension of local polynomial regression, and was introduced by Tibshirani and Hastie (1987).

Local Likelihood Estimation.

Local likelihood estimation is based on a locally weighted version of the log-likelihood:

$$\mathcal{L}_x(\mu_1, \ldots, \mu_n) = \sum_{i=1}^{n} w_i(x) \log\ f(Y_i, \mu_i)\ .$$

A local polynomial approximation is then used for a transformation of the mean function. For example, a local quadratic approximation is

$$\begin{aligned}\boldsymbol{\theta}(x_i) &= g(\mu(x_i)) \\ &\approx a_0 + a_1(x_i - x) + \frac{a_2}{2}(x_i - x)^2\ .\end{aligned}$$

The function $g(\mu)$ is the link function. Its primary goal is to remove constraints on the mean by mapping the parameter space to $(-\infty, \infty)$. For example, in the Poisson case, the parameter space is $0 < \mu < \infty$. If the log transformation $\boldsymbol{\theta} = \log(\mu)$ is used, then the parameter space becomes $-\infty < \boldsymbol{\theta} < \infty$.

Let $l(y, \boldsymbol{\theta}) = \log f(y, \mu)$ where $\boldsymbol{\theta} = g(\mu)$, so that the locally weighted log-likelihood becomes

$$\mathcal{L}_x = \sum_{i=1}^{n} w_i(x) l\left(Y_i, \boldsymbol{\theta}(x_i)\right) \ .$$

The maximizer satisfies the likelihood equations,

$$\sum_{i=1}^{n} w_i(x) \begin{pmatrix} 1 \\ x_i - x \\ \frac{1}{2}(x_i - x)^2 \end{pmatrix} \dot{l}(Y_i, \boldsymbol{\theta}(x_i)) = 0 \ , \tag{20.20}$$

where

$$\dot{l} = \frac{\partial}{\partial \boldsymbol{\theta}} l(y, \boldsymbol{\theta}) \ .$$

In matrix notation, this system of equations can be written in a form similar to (20.7):

$$\boldsymbol{X}^\top \boldsymbol{W} \dot{l}(Y, \boldsymbol{X}a) = 0 \ . \tag{20.21}$$

This system of equations is solved to find parameter estimates $\hat{a}_0, \hat{a}_1$ and $\hat{a}_2$. The local likelihood estimate is defined as

$$\hat{\mu}(x) = g^{-1}(\hat{a}_0) \ .$$

Solving the Local Likelihood Equations.

The local likelihood equations (20.20) are usually non-linear, and so the solution must be obtained through iterative methods. The Newton–Raphson updating formula is

$$\hat{a}^{(j+1)} = \hat{a}^{(j)} + \left(\boldsymbol{X}^\top \boldsymbol{W} \boldsymbol{V} \boldsymbol{X}\right)^{-1} \boldsymbol{X}^\top \boldsymbol{W} \dot{l}\left(Y, \boldsymbol{X}\hat{a}^{(j)}\right) \ , \tag{20.22}$$

where $\boldsymbol{V}$ is a diagonal matrix with entries

$$-\frac{\partial^2}{\partial \boldsymbol{\theta}^2} l(y, \boldsymbol{\theta}) \ .$$

For many common likelihoods $l(Y, \boldsymbol{\theta})$ is concave. Under mild conditions on the design points, this implies that the local likelihood is also concave, and has a unique global maximizer. If the Newton–Raphson algorithm converges, it must converge to this global maximizer.

The Newton–Raphson algorithm (20.22) cannot be guaranteed to converge from arbitrary starting values. But for concave likelihoods, $\hat{a}^{(j+1)} - \hat{a}^{(j)}$ is guaranteed to be an ascent direction, and convergence can be ensured by controlling the step size.

Statistics for the Local Likelihood Estimate.

Since the local likelihood estimate does not have an explicit representation, statistical properties cannot be derived as easily as in the local regression case. But a Taylor series expansion of the local likelihood gives an approximate linearization of the estimate, leading to theory parallel to that developed in Sects. 20.3 and 20.4 for local regression. See Chap. 4 of Loader (1999b).

20.5.3 Extensions of Local Likelihood

The local likelihood method has been formulated for regression models. But variants of the method have been derived for numerous other settings, including robust regression, survival models, censored data, proportional hazards models, and density estimation. References include Tibshirani and Hastie (1987), Hjort and Jones (1996), Loader (1996, 1999b).

Robust Smoothing.

Robust smoothing combines the ideas of robust estimation (Chap. III.9) with smoothing. One method is local M-estimation: choose $\hat{a}$ to minimize

$$\sum_{i=1}^{n} w_i(x)\rho\left(Y_i - \langle a, A(x_i - x)\rangle\right) ,$$

and estimate $\hat{\mu}(x) = \hat{a}_0$. If $\rho(u) = u^2$, this corresponds to local least squares estimation. If $\rho(u)$ is a symmetric function that increases more slowly than u^2, then the resulting estimate is more robust to outliers in the data. One popular choice of $\rho(u)$ is the Huber function:

$$\rho(u) = \begin{cases} u^2 & |u| \le c \\ c(2|u| - c) & |u| > c \end{cases} .$$

References include Hardle (1990) and Loader (1999b). Another variant of M-estimation for local regression is the iterative procedure of Cleveland (1979).

Density Estimation.

Suppose $X_1, \ldots, X_n$ are an independent sample from a density $f(x)$. The goal is to estimate $f(x)$. The local likelihood for this problem is

$$\mathcal{L}_x(a) = \sum_{i=1}^{n} w_i(x)\,\langle a, A(x_i - x)\rangle - n\int_{\mathcal{X}} W\left(\frac{u-x}{h}\right) \mathrm{e}^{\langle a, A(u-x)\rangle}\mathrm{d}u\ .$$

Letting $\hat{a}$ be the maximizer of the local log-likelihood, the local likelihood estimate is $\hat{f}(x) = \exp(\hat{a}_0)$. See Hjort and Jones (1996) and Loader (1996).

The density estimation problem is discussed in detail, together with graphical techniques for visualizing densities, in Chap. III.4.

Acknowledgements This work was supported by National Science Foundation Grant DMS 0306202.

References

Akaike, H.: Information theory and an extension of the maximum likelihood principle. In: Petrov, B.N., Csàki, F. (eds.) Second International Symposium on Information Theory, pp. 267–281. Budapest, Akademia Kiadó (1972)

Akaike, H.: A new look at the statistical model identification. IEEE Trans. Automat. Contr. **19**, 716–723 (1974)

Azzalini, A., Bowman, A.W.: Applied Smoothing Techniques for Data Analysis. Oxford University Press, Oxford (1997)

Cleveland, W.S.: Robust locally weighted regression and smoothing scatterplots. J. Am. Stat. Assoc. **74**, 829–836 (1979)

Cleveland, W.S., Devlin, S.J.: Locally weighted regression: An approach to regression analysis by local fitting. J. Am. Stat. Assoc. **83**, 596–610 (1988)

Efromovich, S.: Nonparametric Curve Estimation. Springer, New York (1999)

Fan, J., Gijbels, I.: Data-driven bandwidth selection in local polynomial fitting: variable bandwidth and spatial adaptation. J. Roy. Stat. Soc. B **57**, 371–394 (1995)

Fan, J., Gijbels, I.: Local Polynomial Modelling and its Applications. Chapman and Hall, London (1996)

Friedman, J., Stuetzle, W.: Projection pursuit regression. J. Am. Stat. Assoc. **76**, 817–823 (1981)

Green, P.J., Silverman, B.: Nonparametric regression and Generalized Linear Models: A Roughness Penalty Approach. Chapman and Hall, London (1994)

Härdle, W.: Applied Nonparametric Regression. Cambridge University Press, Cambridge (1990)

Hart, J.D.: Nonparametric Smoothing and Lack-of-Fit Tests. Springer, New York (1997)

Hastie, T.J., Loader, C.R.: Local regression: Automatic kernel carpentry (with discussion). Stat. Sci. **8**, 120–143 (1993)

Hastie, T.J., Tibshirani, R.J.: Generalized Additive Models. Chapman and Hall, London (1990)

Henderson, R.: Note on graduation by adjusted average. Trans. Actuarial Soc. Am. **17**, 43–48 (1916)

Henderson, R.: A new method of graduation. Trans. Actuarial Soc. Am. **25**, 29–40 (1924)

Hjort, N.L., Jones, M.C.: Locally parametric nonparametric density estimation. Ann. Stat. **24**, 1619–1647 (1996)

Katkovnik, V.Y.: Linear and nonlinear methods of nonparametric regression analysis. Soviet Autom. Contr. **5**, 35–46 (25–34) (1979)

Loader, C.: Local likelihood density estimation. Ann. Stat. **24**, 1602–1618 (1996)

Loader, C.: Bandwidth selection: Classical or plug-in? Ann. Stat. **27**, 415–438 (1999a)

Loader, C.: Local Regression and Likelihood. Springer, New York (1999b)

Mallows, C.L.: Some comments on c_p. Technometrics **15**, 661–675 (1973)

Nadaraya, E.A.: On estimating regression. Theor. Probab. Appl. **9**, 157–159 (141–142) (1964)
Ruppert, D., Sheather, S.J., and Wand, M.P.: An effective bandwidth selector for local least squares regression. J. Am. Stat. Assoc. **90**, 1257–1270 (1995)
Ruppert, D., Wand, M.P.: Multivariate locally weighted least squares regression. Ann. Stat. **22**, 1346–1370 (1994)
Ruppert, D., Wand, M.P., Carroll, R.J.: Semiparametric Regression. Cambridge University Press, Cambridge (2003)
Schiaparelli, G.V.: Sul modo di ricavare la vera espressione delle leggi delta natura dalle curve empiricae. Effemeridi Astronomiche di Milano per l'Arno **857**, 3–56 (1866)
Silverman, B.W.: Some aspects of the spline smoothing approach to nonparametric regression curve fitting (with discussion). J. Roy. Stat. Soc. B **47**, 1–52 (1985)
Stone, C.J.: Consistent nonparametric regression (with discussion). Ann. Stat. **5**, 595–645 (1977)
Tibshirani, R.J., Hastie, T.J.: Local likelihood estimation. J. Am. Stat. Assoc. **82**, 559–567 (1987)
Wahba, G.: Spline Models for Observational Data, SIAM, Philadelphia (1990)
Wahba, G., Wold, S.: A completely automatic French curve: Fitting spline functions by cross-validation. Comm. Stat. **4**, 1–17 (1975)
Watson, G.S.: Smooth regression analysis. Sankhya A **26**, 359–372 (1964)
Whitaker, E.T.: On a new method of graduation. Proc. Edinb. Math. Soc. **41**, 62–75 (1923)

Chapter 21
Semiparametric Models

Joel L. Horowitz

21.1 Introduction

Much empirical research is concerned with estimating conditional mean, median, or hazard functions. For example, labor economists are interested in estimating the mean wages of employed individuals conditional on characteristics such as years of work experience and education. The most frequently used estimation methods assume that the function of interest is known up to a set of constant parameters that can be estimated from data. Models in which the only unknown quantities are a finite set of constant parameters are called *parametric*. The use of a parametric model greatly simplifies estimation, statistical inference, and interpretation of the estimation results but is rarely justified by theoretical or other a priori considerations. Estimation and inference based on convenient but incorrect assumptions about the form of the conditional mean function can be highly misleading.

As an illustration, the solid line in Fig. 21.1 shows an estimate of the mean of the logarithm of weekly wages, $\log W$, conditional on years of work experience, EXP, for white males with 12 years of education who work full time and live in urban areas of the North Central U.S. The estimate was obtained by applying kernel nonparametric regression (see, e.g., Fan and Gijbels 1996; Härdle 1990) to data from the 1993 Current Population Survey (CPS). The estimated conditional mean of $\log W$ increases steadily up to approximately 30 years of experience and is flat thereafter. The dashed and dotted lines in Fig. 21.1 show two parametric estimates of the mean of the logarithm of weekly wages conditional on years of work experience. The dashed line is the ordinary least squares (OLS) estimate that is obtained by assuming that the mean of $\log W$ conditional on EXP is the linear function $E(\log W|EXP) = \beta_0 + \beta_1 EXP$. The dotted line is the OLS estimate that is obtained by assuming that $E(\log W|EXP)$ is the quadratic function

J.L. Horowitz (✉)
Department of Economics, Northwestern University, Evanston, IL, USA
e-mail: joel-horowitz@northwestern.edu

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_21,

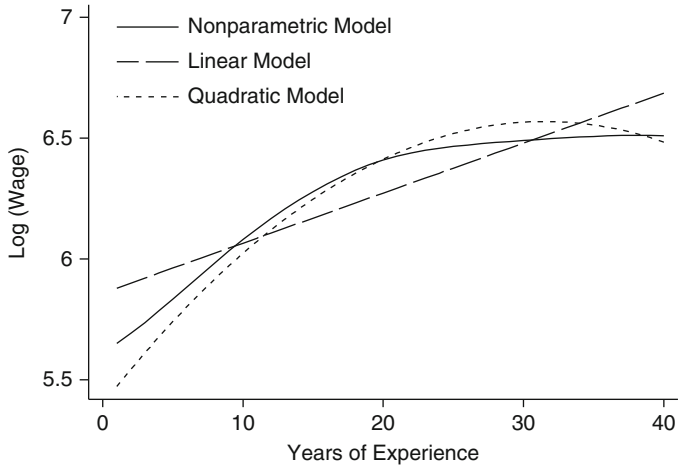


Fig. 21.1 Nonparametric and parametric estimates of mean log wages

$E(\log W|EXP) = \beta_0 + \beta_1 EXP + \beta_2 EXP^2$. The nonparametric estimate (solid line) places no restrictions on the shape of $E(\log W|EXP)$. The linear and quadratic models give misleading estimates of $E(\log W|EXP)$. The linear model indicates that $E(\log W|EXP)$ increases steadily as experience increases. The quadratic model indicates that $E(\log W|EXP)$ decreases after 32 years of experience. In contrast, the nonparametric estimate of $E(\log W|EXP)$ becomes nearly flat at approximately 30 years of experience. Because the nonparametric estimate does not restrict the conditional mean function to be linear or quadratic, it is more likely to represent the true conditional mean function.

The opportunities for specification error increase if Y is binary. For example, consider a model of the choice of travel mode for the trip to work. Suppose that the available modes are automobile and transit. Let $Y = 1$ if an individual chooses automobile and $Y = 0$ if the individual chooses transit. Let X be a vector of explanatory variables such as the travel times and costs by automobile and transit. Then $E(Y|x)$ is the probability that $Y = 1$ (the probability that the individual chooses automobile) conditional on $X = x$. This probability will be denoted $P(Y = 1|x)$. In applications of binary response models, it is often assumed that $P(Y|x) = G(\beta' x)$, where β is a vector of constant coefficients and G is a known probability distribution function. Often, G is assumed to be the cumulative standard normal distribution function, which yields a *binary probit* model, or the cumulative logistic distribution function, which yields a *binary logit* model. The coefficients β can then be estimated by the method of maximum likelihood (Amemiya 1985). However, there are now two potential sources of specification error. First, the dependence of Y on x may not be through the linear *index* $\beta' x$. Second, even if the index $\beta' x$ is correct, the *response function* G may not be the normal or logistic distribution function. See Horowitz (1993a, 1998) for examples of specification errors in binary response models and their consequences.

Many investigators attempt to minimize the risk of specification error by carrying out a *specification search* in which several different models are estimated and conclusions are based on the one that appears to fit the data best. Specification searches may be unavoidable in some applications, but they have many undesirable properties and their use should be minimized. There is no guarantee that a specification search will include the correct model or a good approximation to it. If the search includes the correct model, there is no guarantee that it will be selected by the investigator's model selection criteria. Moreover, the search process invalidates the statistical theory on which inference is based.

The rest of this chapter describes methods that deal with the problem of specification error by relaxing the assumptions about functional form that are made by parametric models. The possibility of specification error can be essentially eliminated through the use of nonparametric estimation methods. They assume that the function of interest is smooth but make no other assumptions about its shape or functional form. However, nonparametric methods have important disadvantages that seriously limit their usefulness in applications. One important problem is that the precision of a nonparametric estimator decreases rapidly as the dimension of the explanatory variable X increases. This phenomenon is called the *curse of dimensionality*. As a result of it, impracticably large samples are usually needed to obtain acceptable estimation precision if X is multidimensional, as it often is in applications. For example, a labor economist may want to estimate mean log wages conditional on years of work experience, years of education, and one or more indicators of skill levels, thus making the dimension of X at least 3.

Another problem is that nonparametric estimates can be difficult to display, communicate, and interpret when X is multidimensional. Nonparametric estimates do not have simple analytic forms. If X is one- or two-dimensional, then the estimate of the function of interest can be displayed graphically as in Fig. 21.1, but only reduced-dimension projections can be displayed when X has three or more components. Many such displays and much skill in interpreting them can be needed to fully convey and comprehend the shape of an estimate.

A further problem with nonparametric estimation is that it does not permit extrapolation. For example, in the case of a conditional mean function it does not provide predictions of $E(Y|x)$ at points x that are outside of the support (or range) of the random variable X. This is a serious drawback in policy analysis and forecasting, where it is often important to predict what might happen under conditions that do not exist in the available data. Finally, in nonparametric estimation, it can be difficult to impose restrictions suggested by economic or other theory. Matzkin (1994) discusses this issue.

Semiparametric methods offer a compromise. They make assumptions about functional form that are stronger than those of a nonparametric model but less restrictive than the assumptions of a parametric model, thereby reducing (though not eliminating) the possibility of specification error. Semiparametric methods permit greater estimation precision than do nonparametric methods when X is multidimensional. They are easier to display and interpret than nonparametric ones and provide limited capabilities for extrapolation and imposing restrictions derived

from economic or other theory models. Section 21.2 of this chapter describes some semiparametric models for conditional mean functions. Section 21.3 describes semiparametric estimators for an important class of hazard models. Section 21.4 is concerned with semiparametric estimation of a certain binary response model.

21.2 Semiparametric Models for Conditional Mean Functions

The term *semiparametric* refers to models in which there is an unknown function in addition to an unknown finite dimensional parameter. For example, the binary response model $P(Y = 1|x) = G(\beta' x)$ is semiparametric if the function G and the vector of coefficients β are both treated as unknown quantities. This section describes two semiparametric models of conditional mean functions that are important in applications. The section also describes a related class of models that has no unknown finite-dimensional parameters but, like semiparametric models, mitigates the disadvantages of fully nonparametric models. Finally, this section describes a class of transformation models that is important in estimation of hazard functions among other applications. Powell (1994) discusses additional semiparametric models.

21.2.1 Single Index Models

In a semiparametric single index model, the conditional mean function has the form

$$E(Y|x) = G(\beta' x) , \tag{21.1}$$

where β is an unknown constant vector and G is an *unknown* function. The quantity $\beta' x$ is called an *index*. The inferential problem is to estimate G and β from observations of (Y, X). G in (21.1) is analogous to a link function in a generalized linear model, except in (21.1) G is unknown and must be estimated.

Model (21.1) contains many widely used parametric models as special cases. For example, if G is the identity function, then (21.1) is a linear model. If G is the cumulative normal or logistic distribution function, then (21.1) is a binary probit or logit model. When G is unknown, (21.1) provides a specification that is more flexible than a parametric model but retains many of the desirable features of parametric models, as will now be explained.

One important property of single index models is that they avoid the curse of dimensionality. This is because the index $\beta' x$ aggregates the dimensions of x, thereby achieving *dimension reduction*. Consequently, the difference between the estimator of G and the true function can be made to converge to zero at the same rate that would be achieved if $\beta' x$ were observable. Moreover, β can be estimated with the same rate of convergence that is achieved in a parametric model. Thus, in

terms of the rates of convergence of estimators, a single index model is as accurate as a parametric model for estimating β and as accurate as a one-dimensional nonparametric model for estimating G. This dimension reduction feature of single index models gives them a considerable advantage over nonparametric methods in applications where X is multidimensional and the single index structure is plausible.

A single-index model permits limited extrapolation. Specifically, it yields predictions of $E(Y|x)$ at values of x that are not in the support of X but are in the support of $\beta'X$. Of course, there is a price that must be paid for the ability to extrapolate. A single index model makes assumptions that are stronger than those of a nonparametric model. These assumptions are testable on the support of X but not outside of it. Thus, extrapolation (unavoidably) relies on untestable assumptions about the behavior of $E(Y|x)$ beyond the support of X.

Before β and G can be estimated, restrictions must be imposed that insure their identification. That is, β and G must be uniquely determined by the population distribution of (Y, X). Identification of single index models has been investigated by Ichimura (1993) and, for the special case of binary response models, Manski (1988). It is clear that β is not identified if G is a constant function or there is an exact linear relation among the components of X (perfect multicollinearity). In addition, (21.1) is observationally equivalent to the model $E(Y|X) = G^*(\gamma + \delta\beta'x)$, where γ and $\delta \neq 0$ are arbitrary and G^* is defined by the relation $G^*(\gamma + \delta v) = G(v)$ for all v in the support of $\beta'X$. Therefore, β and G are not identified unless restrictions are imposed that uniquely specify γ and δ. The restriction on γ is called *location normalization* and can be imposed by requiring X to contain no constant (intercept) component. The restriction on δ is called *scale normalization*. Scale normalization can be achieved by setting the β coefficient of one component of X equal to one. A further identification requirement is that X must include at least one continuously distributed component whose β coefficient is non-zero. Horowitz (1998) gives an example that illustrates the need for this requirement. Other more technical identification requirements are discussed by Ichimura (1993) and Manski (1988).

The main estimation challenge in single index models is estimating β. Given an estimator b_n of β, G can be estimated by carrying out the nonparametric regression of Y on $b_n'X$ (e.g, by using kernel estimation). Several estimators of β are available. Ichimura (1993) describes a nonlinear least squares estimator. Klein and Spady (1993) describe a semiparametric maximum likelihood estimator for the case in which Y is binary. These estimators are difficult to compute because they require solving complicated nonlinear optimization problems. Powell et al. (1989) describe a *density-weighted average derivative estimator* (DWADE) that is non-iterative and easily computed. The DWADE applies when all components of X are continuous random variables. It is based on the relation

$$\beta \propto E\left[p(X)\partial G(\beta'X)/\partial X\right] = -2E\left[Y\,\partial p(X)/\partial X\right] , \tag{21.2}$$

where p is the probability density function of X and the second equality follows from integrating the first by parts. Thus, β can be estimated up to scale by estimating

the expression on the right-hand side of the second equality. Powell et al. (1989) show that this can be done by replacing p with a nonparametric estimator and replacing the population expectation $\boldsymbol{E}$ with a sample average. Horowitz and Härdle (1996) extend this method to models in which some components of X are discrete. Hristache et al. (2001a) developed an iterated average derivative estimator that performs well when X is high-dimensional. Ichimura and Lee (1991) and Hristache et al. (2001b) investigate multiple-index generalizations of (21.1).

The usefulness of single-index models can be illustrated with an example that is taken from Horowitz and Härdle (1996). The example consists of estimating a model of product innovation by German manufacturers of investment goods. The data, assembled in 1989 by the IFO Institute of Munich, consist of observations on 1,100 manufacturers. The dependent variable is $Y = 1$ if a manufacturer realized an innovation during 1989 in a specific product category and 0 otherwise. The independent variables are the number of employees in the product category ($EMPLP$), the number of employees in the entire firm ($EMPLF$), an indicator of the firm's production capacity utilization (CAP), and a variable DEM, which is 1 if a firm expected increasing demand in the product category and 0 otherwise. The first three independent variables are standardized so that they have units of standard deviations from their means. Scale normalization was achieved by setting $\beta_{EMPLP} = 1$.

Table 21.1 shows the parameter estimates obtained using a binary probit model and the semiparametric method of Horowitz and Härdle (1996). Figure 21.2 shows a kernel estimate of $G'(v)$. There are two important differences between the semiparametric and probit estimates. First, the semiparametric estimate of β_{EMPLF} is small and statistically nonsignificant, whereas the probit estimate is significant at the 0.05 level and similar in size to β_{CAP}. Second, in the binary probit model, G is a cumulative normal distribution function, so G' is a normal density function. Figure 21.2 reveals, however, that G' is bimodal. This bimodality suggests that the data may be a mixture of two populations. An obvious next step in the analysis of the data would be to search for variables that characterize these populations. Standard diagnostic techniques for binary probit models would provide no indication that G' is bimodal. Thus, the semiparametric estimate has revealed an important feature of the data that could not easily be found using standard parametric methods.

Table 21.1 Estimated coefficients (Standard Errors) for model of product innovation

EMPLP	EMPLF	CAP	DEM
	Semiparametric model		
1	0.032	0.346	1.732
	(0.023)	(0.078)	(0.509)
	Probit model		
1	0.516	0.520	1.895
	(0.024)	(0.163)	(0.387)

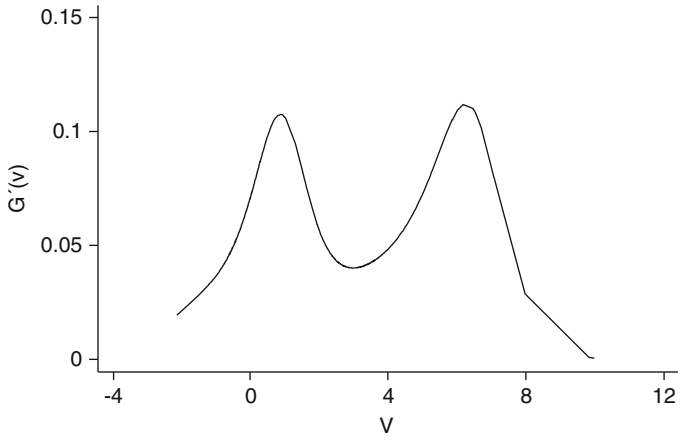


Fig. 21.2 Estimate of $G'(v)$ for model of product innovation

21.2.2 Partially Linear Models

In a partially linear model, X is partitioned into two non-overlapping subvectors, X_1 and X_2. The model has the form

$$E(Y|x_1, x_2) = \beta' x_1 + G(x_2) \ , \tag{21.3}$$

where β is an unknown constant vector and G is an unknown function. This model is distinct from the class of single index models. A single index model is not partially linear unless G is a linear function. Conversely, a partially linear model is a single index model only in this case. Stock (1989, 1991) and Engle et al. (1986) illustrate the use of (21.3) in applications. Identification of β requires the *exclusion restriction* that none of the components of X_1 are perfectly predictable by components of X_2. When β is identified, it can be estimated with an $n^{-1/2}$ rate of convergence regardless of the dimensions of X_1 and X_2. Thus, the curse of dimensionality is avoided in estimating β.

An estimator of β can be obtained by observing that (21.3) implies

$$Y - E(Y|x_2) = \beta' [X_1 - E(X_1|x_2)] + U \ , \tag{21.4}$$

where U is an unobserved random variable satisfying $E(U|x_1, x_2) = 0$. Robinson (1988) shows that under regularity conditions, β can be estimated by applying OLS to (21.4) after replacing $E(Y|x_2)$ and $E(X_1|x_2)$ with nonparametric estimators. The estimator of β, b_n, converges at rate $n^{-1/2}$ and is asymptotically normally distributed. G can be estimated by carrying out the nonparametric regression of $Y - b_n' X_1$ on X_2. Unlike b_n, the estimator of G suffers from the curse of dimensionality; its rate of convergence decreases as the dimension of X_2 increases.

21.2.3 Nonparametric Additive Models

Let X have d continuously distributed components that are denoted $X_1, \ldots, X_d$. In a nonparametric additive model of the conditional mean function,

$$E(Y|x) = \mu + f_1(x_1) + \ldots + f_d(x_d) \ , \tag{21.5}$$

where μ is a constant and $f_1, \ldots, f_d$ are unknown functions that satisfy a location normalization condition such as

$$\int f_k(v) w_k(v) \mathrm{d}v = 0 \ , \quad k = 1, \ldots, d \ , \tag{21.6}$$

where w_k is a non-negative weight function. An additive model is distinct from a single index model unless $E(Y|x)$ is a linear function of x. Additive and partially linear models are distinct unless $E(Y|x)$ is partially linear and G in (21.3) is additive.

An estimator of f_k $(k = 1, \ldots, d)$ can be obtained by observing that (21.5) and (21.6) imply

$$f_k(x_k) = \int E(Y|x) w_{-k}(x_{-k}) \mathrm{d}x_{-k} \ , \tag{21.7}$$

where x_{-k} is the vector consisting of all components of x except the k'th and w_{-k} is a weight function that satisfies $\int w_{-k}(x_{-k}) \mathrm{d}x_{-k} = 1$. The estimator of f_k is obtained by replacing $E(Y|x)$ on the right-hand side of (21.7) with nonparametric estimators. Linton and Nielsen (1995) and Linton (1997) present the details of the procedure and extensions of it. Under suitable conditions, the estimator of f_k converges to the true f_k at rate $n^{-2/5}$ regardless of the dimension of X. Thus, the additive model provides dimension reduction. It also permits extrapolation of $E(Y|x)$ within the rectangle formed by the supports of the individual components of X. Mammen et al. (1999) describe a backfitting procedure that is likely to be more precise than the estimator based on (21.7) when d is large. See Hastie and Tibshirani (1990) for an early discussion of backfitting.

Linton and Härdle (1996) describe a generalized additive model whose form is

$$E(Y|x) = G\left[\mu + f_1(x_1) + \ldots + f_K(x_d)\right] \ , \tag{21.8}$$

where $f_1, \ldots, f_d$ are unknown functions and G is a known, strictly increasing (or decreasing) function. Horowitz (2001) describes a version of (21.8) in which G is unknown. Both forms of (21.8) achieve dimension reduction. When G is unknown, (21.8) nests additive and single index models and, under certain conditions, partially linear models.

The use of the nonparametric additive specification (21.5) can be illustrated by estimating the model $E(\log W|EXP, EDUC) = \mu + f_{EXP}(EXP) + f_{EDUC}(EDUC)$, where W and EXP are defined as in Sect. 21.1, and $EDUC$ denotes years of

education. The data are taken from the 1993 CPS and are for white males with 14 or fewer years of education who work full time and live in urban areas of the North Central U.S. The results are shown in Fig. 21.3. The unknown functions f_{EXP} and f_{EDUC} are estimated by the method of Linton and Nielsen (1995) and are normalized so that $f_{EXP}(2) = f_{EDCU}(5) = 0$. The estimates of f_{EXP} (Fig. 21.3a) and f_{EDUC} (Fig. 21.3b) are nonlinear and differently shaped. Functions f_{EXP} and f_{EDUC} with different shapes cannot be produced by a single index model, and a lengthy specification search might be needed to find a parametric model that produces the shapes shown in Fig. 21.3. Some of the fluctuations of the estimates of f_{EXP} and f_{EDUC} may be artifacts of random sampling error rather than features of $E(\log W|EXP, EDUC)$. However, a more elaborate analysis that takes account

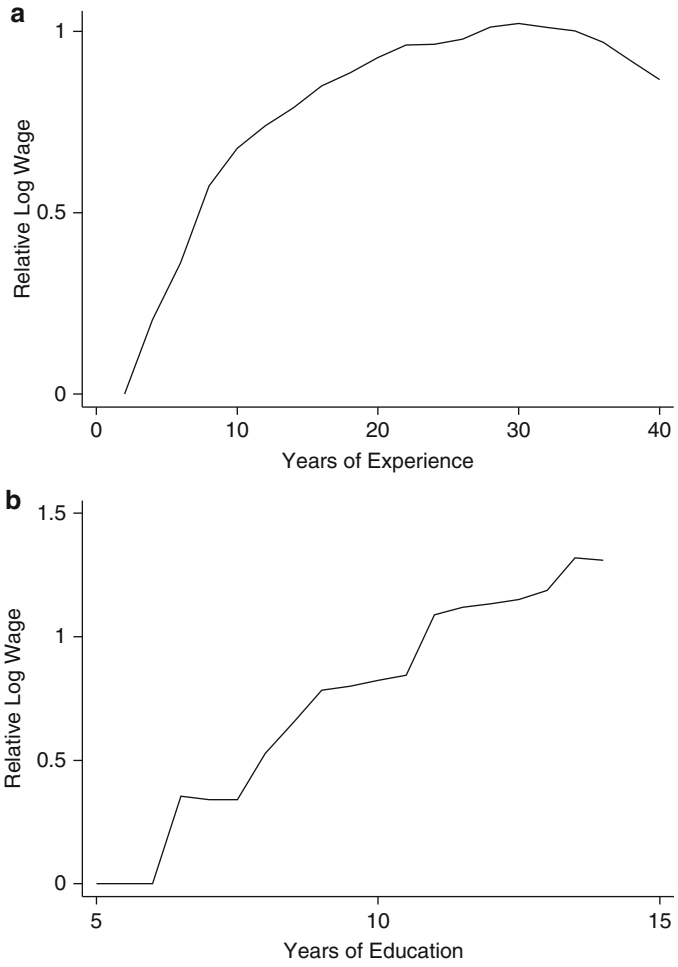


Fig. 21.3 Components of nonparametric, additive wage equation

of the effects of random sampling error rejects the hypothesis that either function is linear.

21.2.4 Transformation Models

A transformation model has the form

$$H(Y) = \beta' X + U \ , \tag{21.9}$$

where H is an unknown increasing function, β is an unknown finite dimensional vector of constants, and U is an unobserved random variable. It is assumed here that U is statistically independent of X. The aim is to estimate H and β. One possibility is to assume that H is known up to a finite-dimensional parameter. For example, H could be the Box-Cox transformation

$$H(y) = \begin{cases} (y^\tau - 1)/\tau & \text{if } \tau > 0 \\ \log y & \text{if } \tau = 0 \end{cases}$$

where τ is an unknown parameter. Methods for estimating transformation models in which H is parametric have been developed by Amemiya and Powell (1981) and Foster et al. (2001) among others.

Another possibility is to assume that H is unknown but that the distribution of U is known. Cheng et al. (1995); Cheng et al. (1997) have developed estimators for this version of (21.9). Consider, first, the problem of estimating β. Let F denote the (known) cumulative distribution function (CDF) of U. Let (Y_i, X_i) and (Y_j, X_j) $(i \neq j)$ be two distinct, independent observations of (Y, X). Then it follows from (21.9) that

$$E\left[I(Y_i > Y_j)|X_i = x_i, X_j = x_j\right] = P\left[U_i - U_j > -(x_i - x_j)\right] \ . \tag{21.10}$$

Let $G(z) = P(U_i - U_j > z)$ for any real z. Then

$$G(z) = \int_{-\infty}^{\infty} [1 - F(u + z)] \, \mathrm{d}F(u) \ .$$

G is a known function because F is assumed known. Substituting G into (21.10) gives

$$E\left[I(Y_i > Y_j)|X_i = x_i, X_j = x_j\right] = G\left[-\beta'(x_i - x_j)\right] \ .$$

Define $X_{ij} = X_i - X_j$. Then it follows that β satisfies the moment condition

$$E\left\{w\left(\beta' X_{ij}\right) X_{ij}\left[I\left(Y_i > Y_j\right) - G\left(-\beta' X_{ij}\right)\right]\right\} = 0 \tag{21.11}$$

where w is a weight function. Cheng et al. (1995) propose estimating β by replacing the population moment condition (21.11) with the sample analog

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\left\{w\left(b'X_{ij}\right)X_{ij}\left[I\left(Y_i > Y_j\right) - G\left(-b'X_{ij}\right)\right]\right\} = 0\,. \tag{21.12}$$

The estimator of β, b_n, is the solution to (21.12). Equation (21.12) has a unique solution if $w(z) = 1$ for all z and the matrix $\sum_i \sum_j X'_{ij} X_{ij}$ is positive definite. It also has a unique solution asymptotically if w is positive everywhere (Cheng et al. 1995). Moreover, b_n converges almost surely to β. Cheng et al. (1995) also give conditions under which $n^{1/2}(b_n - \beta)$ is asymptotically normally distributed with a mean of 0.

The problem of estimating the transformation function is addressed by Cheng et al. (1997). Equation (21.11) implies that for any real y and vector x that is conformable with X, $EI[I(Y \leq y)|X = x] - F[H(y) - \beta' x] = 0$. Cheng et al. (1997) propose estimating $H(y)$ by the solution to the sample analog of this equation. That is, the estimator $H_n(y)$ solves

$$n^{-1}\sum_{i=1}^{n}\left\{I\left(Y_i \leq y\right) - F\left[H_n(y) - b_n' X_i\right]\right\} = 0\,,$$

where b_n is the solution to (21.12). Cheng et al. (1997) show that if F is strictly increasing on its support, then $H_n(y)$ converges to $H(y)$ almost surely uniformly over any interval $[0, t]$ such that $P(Y > t) > 0$. Moreover, $n^{1/2}(H_n - H)$ converges to a mean-zero Gaussian process over this interval.

A third possibility is to assume that H and F are both nonparametric in (21.9). In this case, certain normalizations are needed to make identification of (21.9) possible. First, observe that (21.9) continues to hold if H is replaced by cH, β is replaced by $c\beta$, and U is replaced by cU for any positive constant c. Therefore, a scale normalization is needed to make identification possible. This will be done here by setting $|\beta_1| = 1$, where β_1 is the first component of β. Observe, also, that when H and F are nonparametric, (21.9) is a semiparametric single-index model. Therefore, identification of β requires X to have at least one component whose distribution conditional on the others is continuous and whose β coefficient is non-zero. Assume without loss of generality that the components of X are ordered so that the first satisfies this condition.

It can also be seen that (21.9) is unchanged if H is replaced by $H + d$ and U is replaced by $U + d$ for any positive or negative constant d. Therefore, a location normalization is also needed to achieve identification when and F are nonparametric. Location normalization will be carried out here by assuming that $H(y_0) = 0$ for some finite y_0 With this location normalization, there is no centering assumption on U and no intercept term in X.

Now consider the problem of estimating H, β, and F. Because (21.9) is a single-index model in this case, β can be estimated using the methods described in Sect. 21.1. Let b_n denote the estimator of β. One approach to estimating H and F is given by Horowitz and Härdle (1996). To describe this approach, define $Z = \beta' X$. Let $G(\cdot|z)$ denote the CDF of Y conditional on $Z = z$. Set $G_y(y|z) = \partial G(y|z)/\partial z$ and $G_z(y|z) = \partial G(y|z)/\partial z$. Then it follows from (21.9) that $H'(y) =$

$-G_y(y|z)/G_z(y|z)$ and that

$$H(y) = -\int_{y_0}^{y} \left[G_y(v|z)/G_z(v|z)\right] \mathrm{d}v \tag{21.13}$$

for any z such that the denominator of the integrand is non-zero. Now let $w(\cdot)$ be a scalar-valued, non-negative weight function with compact support S_w such that the denominator of $G_z(v|z)$ is bounded away from 0 for all $v \in [y_0, y]$ and $z \in S_w$. Also assume that

$$\int_{S_w} w(z)\mathrm{d}z = 1 \ .$$

Then

$$H(y) = -\int_{y_0}^{y} \int_{S_w} w(z) \left[G_y(v|z)/G_z(v|z)\right] \mathrm{d}z \, \mathrm{d}v \ . \tag{21.14}$$

Horowitz and Härdle (1996) obtains an estimator of H from (21.14) by replacing G_y and G_z by kernel estimators. Specifically, G_y is replaced by a kernel estimator of the probability density function of Y conditional on $b_n'X = z$, and G_z is replaced by a kernel estimator of the derivative with respect to z of the CDF of Y conditional on $b_n'X = z$. Denote these estimators by G_{ny} and G_{nz}. Then the estimator of H is

$$H_n(y) = -\int_{y_0}^{y} \int_{S_w} w(z) \left[G_{ny}(v|z)/G_{nz}(v|z)\right] \mathrm{d}z \, \mathrm{d}v \ . \tag{21.15}$$

Horowitz and Härdle (1996) gives conditions under which H_n is uniformly consistent for H and $n^{1/2}(H_n - H)$ converges weakly to a mean-zero Gaussian process. Horowitz and Härdle (1996) also shows how to estimate F, the CDF of U, and gives conditions under which $n^{1/2}(F_n - F)$ converges to a mean-zero Gaussian process, where F_n is the estimator. Gorgens and Horowitz (1999) extend these results to a censored version of (21.9). Integration over z in (21.14) and (21.15) accelerates the convergence of H_n to H. Kernel estimators converge in probability at rates slower than $n^{-1/2}$. Therefore, $G_{ny}(v|z)/G_{nz}(v|z)$ is not $n^{-1/2}$-consistent for $G_y(v|z)/G_z(v|z)$. However, integration over z and v in (21.15) creates an averaging effect that causes the integral and, therefore, H_n to converge at the rate $n^{-1/2}$. This is the reason for basing the estimator on (21.14) instead of (21.13).

Other estimators of H when and F are both nonparametric have been proposed by Ye (1997) and Chen (2002). Chen uses a rank-based approach that is in some ways simpler than that of Horowitz and Härdle (1996) and may have better finite-sample performance. To describe this approach, define $d_{iy} = I(Y_i > y)$ and $d_{jy_0} = I(Y_j > y_0)$. Let $i \neq j$. Then $E(d_{iy} - d_{jy0}|X_i, X_j) \geq 0$ whenever $Z_i - Z_j \geq H(y)$. This suggests that if β were known, then $H(y)$ could be estimated by

$$H_n(y) = \arg\max_{\tau} \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} (d_{iy} - d_{iy_0}) I(Z_i - Z_j \geq \tau) \ .$$

Since β is unknown, Chen (2002) proposes

$$H_n(y) = \arg\max_{\tau} \frac{1}{n(n-1)} \sum_{i=1}^{n} \sum_{\substack{j=1 \\ j \neq i}}^{n} (d_{iy} - d_{iy_0}) I(b_n' X_i - b_n' X_j \geq \tau) \ .$$

Chen (2002) gives conditions under which H_n is uniformly consistent for H and $n^{1/2}(H_n - H)$ converges to a mean-zero Gaussian process. Chen (2002) also shows how this method can be extended to a censored version of (21.9).

21.3 The Proportional Hazards Model with Unobserved Heterogeneity

Let T denote a duration such as that of a spell of employment or unemployment. Let $F(t|x) = P(T \leq t|X = x)$ where X is a vector of covariates. Let $f(t|x)$ denote the corresponding conditional probability density function. The conditional hazard function is defined as

$$\lambda(t|x) = \frac{f(t|x)}{1 - F(t|x)} \ .$$

This section is concerned with an approach to modeling $\lambda(t|x)$ that is based on the proportional hazards model of Cox (1972).

The proportional hazards model is widely used for the analysis of duration data. Its form is

$$\lambda(t|x) = \lambda_0(t) \mathrm{e}^{-x'\beta} \ , \tag{21.16}$$

where β is a vector of constant parameters that is conformable with X and λ_0 is a non-negative function that is called the baseline hazard function.indexBaseline hazard function The essential characteristic of (21.16) that distinguishes it from other models is that $\lambda(t|x)$ is the product of a function of t alone and a function of x alone. Cox (1972) developed a partial likelihood estimator of β and a nonparametric estimator of λ_0. Tsiatis (1981) derived the asymptotic properties of these estimators.

In the proportional hazards model with unobserved heterogeneity, the hazard function is conditioned on the covariates X and an unobserved random variable U that is assumed to be independent of X. The form of the model is

$$\lambda(t|x, u) = \lambda_0(t) \mathrm{e}^{-(\beta' x + u)} \ , \tag{21.17}$$

where $\lambda(\cdot|x,u)$ is the hazard conditional on $X = x$ and $U = u$. In a model of the duration of employment U might represent unobserved attributes of an individual (possibly ability) that affect employment duration. A variety of estimators of λ_0 and β have been proposed under the assumption that λ_0 or the distribution of U or both are known up to a finite-dimensional parameter. See, for example, Lancaster (1979), Heckman and Singer (1984a), Meyer (1990), Nielsen et al. (1992), and Murphy (1994, 1995). However, λ_0 and the distribution of U are nonparametrically identified (Elbers and Ridder 1982; Heckman and Singer 1984a), which suggests that they can be estimated nonparametrically.

Horowitz (1999) describes a nonparametric estimator of λ_0 and the density of U in model (21.17). His estimator is based on expressing (21.17) as a type of transformation model. To do this, define the integrated baseline hazard funtion, Λ_0 by

$$\Lambda_0(t) = \int_0^t \lambda_0(\tau)\mathrm{d}\tau \ .$$

Then it is not difficult to show that (21.17) is equivalent to the transformation model

$$\log \Lambda_0(T) = X'\beta + U + \varepsilon \ , \tag{21.18}$$

where ε is a random variable that is independent of X and U and has the CDF $F_\varepsilon(y) = 1 - \exp(-\mathrm{e}^y)$. Now define $\sigma = |\beta_1|$, where β_1 is the first component of β and is assumed to be non-zero. Then β/σ and $H = \sigma^{-1}\log\Lambda_0$ can be estimated by using the methods of Sect. 21.4. Denote the resulting estimators of β/σ and H by α_n and H_n. If σ were known, then β and Λ_0 could be estimated by $b_n = \sigma\alpha_n$ and $\Lambda_{n0} = \exp(\sigma H_n)$. The baseline hazard function λ_0 could be estimated by differentiating Λ_{n0}. Thus, it is necessary only to find an estimator of the scale parameter σ.

To do this, define $Z = \beta' X$, and let $G(\cdot|z)$ denote the CDF of T conditional on $Z = z$. It can be shown that

$$G(t|z) = 1 - \int \exp\left[-\Lambda_0(t)\mathrm{e}^{-(\beta' x + u)}\right]\mathrm{d}F(u) \ ,$$

where F is the CDF of U. Let p denote the probability density function of Z. Define $G_z(t|z) = \partial G(t|z)/\partial z$ and

$$\sigma(t) = \frac{\int G_z(t|z)p(z)^2\mathrm{d}z}{\int G(t|z)p(z)^2\mathrm{d}z} \ .$$

Then it can be shown using l'Hospital's rule that if $\Lambda_0(t) > 0$ for all $t > 0$, then

$$\sigma = \lim_{t\to 0}\sigma(t) \ .$$

To estimate σ, let p_n, G_{nz} and G_n be kernel estimators of p, G_z and G, respectively, that are based on a simple random sample of (T, X). Define

$$\sigma_n(t) = \frac{\int G_{nz}(t|z)\, p_n(z)^2 \mathrm{d}z}{\int G_n(t|z)\, p_n(z)^2 \mathrm{d}z} .$$

Let c, d, and δ be constants satisfying $0 < c < \infty$, $1/5 < d < 1/4$, and $1/(2d) < \delta < 1$. Let $\{t_{n1}\}$ and $\{t_{n2}\}$ be sequences of positive numbers such that $\Lambda_0(t_{n1}) = cn^{-d}$ and $\Lambda_0(t_{n2}) = cn^{-\delta d}$. Then σ is estimated consistently by

$$\sigma_n = \frac{\sigma_n(t_{n1}) - n^{-d(1-\delta)}\sigma_n(t_{n2})}{n^{-d(1-\delta)}} .$$

Horowitz (1999) gives conditions under which $n^{(1-d)/2}(\sigma_n - \sigma)$ is asymptotically normally distributed with a mean of zero. By choosing d to be close to $1/5$, the rate of convergence in probability of σ_n to σ can be made arbitrarily close to $n^{-2/5}$, which is the fastest possible rate (Ishwaran 1996). It follows from an application of the delta method that the estimators of β, Λ_0, and λ_0 that are given by $b_n = \sigma_n \alpha_n$, $\Lambda_{n0} = \exp(\sigma_n H_n)$, and $\lambda_{n0} = \mathrm{d}\Lambda_{n0}/\mathrm{d}t$ are also asymptotically normally distributed with means of zero and $n^{-(1-d)/2}$ rates of convergence. The probability density function of U can be estimated consistently by solving the deconvolution problem $W_n = U + \varepsilon$, where $W_n = \log \Lambda_{n0}(T) - X'\beta_n$. Because the distribution of ε is "supersmooth," the resulting rate of convergence of the estimator of the density of U is $(\log n)^{-m}$, where m is the number of times that the density is differentiable. This is the fastest possible rate. Horowitz (1999) also shows how to obtain data-based values for t_{n1} and t_{n2} and extends the estimation method to models with censoring.

If panel data on (T, X) are available, then Λ_0 can be estimated with a $n^{-1/2}$ rate of convergence, and the assumption of independence of U from X can be dropped. Suppose that each individual in a random sample of individuals is observed for exactly two spells. Let $(T_j, X_j : j = 1, 2)$ denote the values of (T, X) in the two spells. Define $Z_j = \beta' X_j$. Then the joint survivor function of T_1 and T_2 conditional on $Z_1 = z_1$ and $Z_2 = z_2$ is

$$\begin{aligned} S(t_1, t_2|Z_1, Z_2) &\equiv P(T_1 > t_1, T_2 > t_2|Z_1, Z_2) \\ &= \int \exp\left[-\Lambda_0(t_1)\, \mathrm{e}^{z_1+u} - \Lambda_0(t_2)\, \mathrm{e}^{z_2+u}\right] \mathrm{d}P(u|Z_1 = z_1, Z_2 = z_2). \end{aligned}$$

Honoré (1993) showed that

$$R(t_1, t_2|z_1, z_2) \equiv \frac{\partial S(t_1, t_2|z_1, z_2)/\partial t_1}{\partial S(t_1, t_2|z_1, z_2)/\partial t_2} = \frac{\lambda_0(t_1)}{\lambda_0(t_2)} \exp(z_1 - z_2) .$$

Adopt the scale normalization

$$\int_{S_T} \frac{w_t(\tau)}{\lambda_0(\tau)} \mathrm{d}\tau = 1 \ ,$$

where w_t is a non-negative weight function and S_T is its support. Then

$$\lambda_0(t) = \int_{S_T} w_t(\tau) \exp(z_2 - z_1) \, R(t, \tau | z_2, z_1) \, \mathrm{d}\tau \ .$$

Now for a weight function ω_z with support S_Z, define

$$w(\tau, z_1, z_2) = w_t(\tau) w_z(z_1) \, w_z(z_2) \ .$$

Then,

$$\lambda_0(t) = \int_{S_T} \mathrm{d}\tau \int_{S_Z} \mathrm{d}z_1 \int_{S_Z} \mathrm{d}z_2 w(\tau, z_1, z_2) \exp(z_2 - z_1) \, R(t, \tau | z_1, z_2) \ . \tag{21.19}$$

The baseline hazard function can now be estimated by replacing R with an estimator, R_n, in (21.19). This can be done by replacing Z with $X'b_n$, where b_n is a consistent estimator of β such as a marginal likelihood estimator (Chamberlain 1985; Kalbfleisch and Prentice 1980; Lancaster 2000; Ridder and Tunali 1999), and replacing S with a kernel estimator of the joint survivor function conditional $X_1'b_n = z_1$ and $X_2'b_n = z_2$. The resulting estimator of λ_0 is

$$\lambda_{n0}(t) =$$
$$\int_{S_T} \mathrm{d}\tau \int_{S_Z} \mathrm{d}z_1 \int_{S_Z} \mathrm{d}z_2 w(\tau, z_1, z_2) \exp(z_2 - z_1) \, R_n(t, \tau | z_1, z_2) \ .$$

The integrated baseline hazard function is estimated by

$$\Lambda_{n0}(t) = \int_0^t \lambda_{n0}(\tau) \mathrm{d}\tau \ .$$

Horowitz and Lee (2004) give conditions under which $n^{1/2}(\Lambda_{n0} - \Lambda_0)$ converges weakly to a tight, mean-zero Gaussian process. The estimated baseline hazard function λ_{n0} converges at the rate $n^{-q/(2q+1)}$, where $q \geq 2$ is the number of times that λ_0 is continuously differentiable. Horowitz and Lee (2004) also show how to estimate a censored version of the model.

21.4 A Binary Response Model

The general binary response model has the form

$$Y = I(\beta' X + U > 0) , \tag{21.20}$$

where U is an unobserved random variable. If the distribution of U is unknown but depends on X only through the index $\beta' X$, then (21.20) is a single-index model, and β can be estimated by the methods described in Sect. 21.1. An alternative model that is non-nested with single-index models can be obtained by assuming that $median(U|X = x) = 0$ for all x. This assumption places only weak restrictions on the relation between X and the distribution of U. Among other things, it accommodates fairly general types of heteroskedasticity of unknown form, including random coefficients. Under median centering, the inferential problem is to estimate β. The response function, $P(Y = 1|X = x)$ is not identified without making assumptions about the distribution of U that are stronger than those needed to identify and estimate β. Without such assumptions, the only restriction on $P(Y = 1|X = x)$ under median centering is that

$$P(Y = 1|X = x) \begin{cases} > 0.5 & \text{if } \beta' x > 0 \\ = 0.5 & \text{if } \beta' x = 0 \\ < 0.5 & \text{if } \beta' x < 0 \end{cases}$$

Manski (1975, 1985) proposed the first estimator of β under median centering. Let the data be the simple random sample $\{Y_i, X_i : i = 1, \ldots, n\}$. The estimator is called the *maximum score* estimator and is

$$b_n = \arg \max_{\|b\|=1} n^{-1} \sum_{i=1}^{n} (2Y_i - 1) I(b' X_i \geq 0) , \tag{21.21}$$

where $\|b\|$ denotes the Euclidean norm of the vector b. The restriction $\|b\| = 1$ is a scale normalization. Scale normalization is needed for identification because (21.20) identifies β only up to scale. Manski (1975, 1985) gave conditions under which b_n consistently estimates β. The rate of convergence of b_n and its asymptotic distribution were derived by Cavanagh (1987) and Kim and Pollard (1990). They showed that the rate of convergence in probability of b_n to β is $n^{-1/3}$ and that $n^{1/3}(b_n - \beta)$ converges in distribution to the maximum of a complicated multidimensional stochastic process. The complexity of the limiting distribution of the maximum score estimator limits its usefulness for statistical inference. Delgado et al. (2001) proposed using subsampling methods to form confidence intervals for β.

The maximum score estimator has a slow rate of convergence and a complicated asymptotic distribution because it is obtained by maximizing a step function.

Table 21.2 Smoothed maximum score estimates of a work-trip mode-choice model

		Half-Width of nominal 90% Conf. interval based on	
Variable[a]	Estimated Coefficient	Asymp. normal Approximation	Bootstrap
INTRCPT	−1.5761	0.2812	0.7664
AUTOS	2.2418	0.2989	0.7488
DOVTT	0.0269	0.0124	0.0310
DIVTT	0.0143	0.0033	0.0087
DCOST	1.0[b]		

[a]Definitions of variables: INTRCPT: Intercept term equal to 1; AUTOS: Number of cars owned by traveler's household; DOVTT: Transit out-of-vehicle travel time minus automobile out-of-vehicle travel time (minutes); DIVTT: Transit in-vehicle travel time minus automobile in-vehicle travel time; DCOST: Transit fare minus automobile travel cost ($)
[b]Coefficient equal to 1 by scale normalization

Horowitz (1992) proposed replacing the indicator function in (21.21) by a smooth function. The resulting estimator of β is called the *smoothed maximum score* estimator. Specifically, let K be a smooth function, possibly but not necessarily a distribution function, that satisfies $K(-\infty) = 0$ and $K(\infty) = 1$. Let $\{h_n : n = 1, 2, \ldots\}$ be a sequence of strictly positive constants (bandwidths) that satisfies $h_n \to 0$ as $n \to \infty$. The smoothed maximum score estimator, b_{ns}, is

$$b_{ns} = \arg\max_{b \in B} \sum_{i=1}^{n} (2Y_i - 1) K(X_i' b / h_n) ,$$

where B is a compact parameter set that satisfies the scale normalization $|b_1| = 1$. Horowitz (1992) shows that under assumptions that are stronger than those of Manski (1975, 1985) but still quite weak, $n^r(b_{ns} - \beta)$ is asymptotically normal, where $2/5 \leq r < 1/2$ and the exact value of r depends on the smoothness of the distribution of $X'\beta$ and of $P(Y = 1|X = x)$. Moreover, the smoothed maximum score estimator has the fastest possible rate of convergence under its assumptions (Horowitz 1993b). Monte Carlo evidence suggests that the asymptotic normal approximation can be inaccurate with samples of practical size. However, Horowitz (2002) shows that the bootstrap, which is implemented by sampling the data randomly with replacement, provides asymptotic refinements for tests of hypotheses about β and produces low ERPs for these tests. Thus, the bootstrap provides a practical way to carry out inference with the smoothed maximum score estimator.

Horowitz (1993c) used the smoothed maximum score method to estimate the parameters of a model of the choice between automobile and transit for work trips in the Washington, D.C., area. The explanatory variables are defined in Table 21.2. Scale normalization is achieved by setting the coefficient of DCOST equal to 1. The data consist of 842 observations sampled randomly from the Washington, D.C., area transportation study. Each record contains information about a single trip to work,

including the chosen mode (automobile or transit) and the values of the explanatory variables. Column 2 of Table 21.2 shows the smoothed maximum score estimates of the model's parameters. Column 3 shows the half-widths of nominal 90% symmetrical confidence intervals based on the asymptotic normal approximation (half width equals 1.67 times the standard error of the estimate). Column 4 shows half-widths obtained from the bootstrap. The bootstrap confidence intervals are 2.5–3 times wider than the intervals based on the asymptotic normal approximation. The bootstrap confidence interval for the coefficient of DOVTT contains 0, but the confidence interval based on the asymptotic normal approximation does not. Therefore, the hypothesis that the coefficient of DOVTT is zero is not rejected at the 0.1 level based on the bootstrap but is rejected based on the asymptotic normal approximation.

Acknowledgements Research supported in part by NSF Grant SES-9910925.

References

Amemiya, T.: Advanced Econometrics. Harvard University Press, Cambridge (1985)
Amemiya, T., Powell, J.L.: A comparison of the Box-Cox maximum likelihood estimator and the Non-linear Two-Stage least squares estimator. J. Econometric. **17**, 351–381 (1981)
Cavanagh, C.L.: Limiting Behavior of Estimators Defined by Optimization, unpublished manuscript (1987)
Chamberlain, G.: Heterogeneity, Omitted Variable Bias, and Duration Dependence. In: Heckman, J.J., Singer, B. (eds.) Longitudinal Analysis of Labor Market Data, pp. 3–38. Cambridge University Press, Cambridge (1985)
Chen, S.: Rank estimataion of transformation models. Econometrica **70**, 1683–1697 (2002)
Cheng, S.C., Wei, L.J., Ying, Z.: Analysis of transformation models with censored data. Biometrika **82**, 835–845 (1995)
Cheng, S.C., Wei, L.J., Ying, Z.: Predicting survival probabilities with semiparametric transformation models. J. Am. Stat. Assoc. **92**, 227–235 (1997)
Cox, D.R.: Regression models and life tables. J. Roy. Stat. Soc. B **34**, 187–220 (1972)
Delgado, M.A., Rodríguez-Poo, J.M., Wolf, M.: Subsampling inference in cube root asymptotics with an application to manski's maximum score estimator. Econ. Lett. **73**, 241–250 (2001)
Elbers, C., Ridder, G.: True and spurious duration dependence: The identifiability of the proportional hazard model. Rev. Econ. Stud. **49**, 403–409 (1982)
Engle, R.F., Granger, C.W.J., Rice, J., Weiss, A.: Semiparametric estimates of the relationship beteween weather and electricity sales. J. Am. Stat. Assoc. **81**, 310–320 (1986)
Fan, J., Gijbels, I.: Local Polynomial Modelling and Its Applications. Chapman & Hall, London (1996)
Foster, A.M., Tian, L., Wei, L.J.: Estimation for the Box-Cox transformation model without assuming prametric error distribution. J. Am. Stat. Assoc. **96**, 1097–1101 (2001)
Gørgens, T., Horowitz, J.L.: Semiparametric estimation of a censored regression model with an unknown transformation of the dependent variable. J. Econometric. **90**, 155–191 (1999)
Härdle, W.: Applied Nonparametric Regression. Cambridge University Press, Cambridge (1990)
Hastie, T.J., Tibshirani, R.J.: Generalized Additive Models. Chapman and Hall, London (1990)
Heckman, J., Singer, B.: A method for minimizing the impact of distributional assumptions in econometric models for duration data. Econometrica **52**, 271–320 (1984a)
Heckman, J., Singer, B.: The identifiability of the proportional hazard model. Rev. Econ. Stud. **51**, 231–243 (1984a)

Honoré, B.E.: Identification results for duration models with multiple spells. Rev. Econ. Stud. **60**, 241–246 (1993)
Horowitz, J.L.: A smoothed maximum score estimator for the binary response model. Econometrica **60**, 505–531 (1992)
Horowitz, J.L.: Semiparametric and nonparametric estimation of quantal response models. In: Maddala, G.S., Rao, C.R., Vinod H.D. (eds.) Handbook of Statistics, Vol. 11, Elsevier, Amsterdam 45–72 (1993a)
Horowitz, J.L.: Optimal rates of convergence of parameter estimators in the binary response model with weak distributional assumptions. Econometric Theor. **9**, 1–18 (1993b)
Horowitz, J.L.: Semiparametric estimation of a work-trip mode choice model. J. Econometric. **58**, 49–70 (1993c)
Horowitz, J.L.: Semiparametric estimation of a regression model with an unknown transformation of the dependent variable. Econometrica **64**, 103–137 (1996)
Horowitz, J.L.: Semiparametric Methods in Econometrics. Springer, New York (1998)
Horowitz, J.L.: Semiparametric estimation of a proportional hazard model with unobserved heterogeneity. Econometrica **67**, 1001–1028 (1999)
Horowitz, J.L.: Nonparametric estimation of a generalized additive model with an unknown link function. Econometrica **69**, 499–513 (2001)
Horowitz, J.L.: Bootstrap critical values for tests based on the smoothed maximum score estimator. J. Econometric. **111**, 141–167 (2002)
Horowitz, J.L., Härdle, W.: Direct semiparametric estimation of Single-Index models with discrete covariates. J. Am. Stat. Assoc. **91**, 1632–1640 (1996)
Horowitz, J.L., Lee, S.: Semiparametric estimation of a panel data proportional hazards model with fixed effects. J. Econometric. **119**, 155–198 (2004)
Hristache, M., Juditsky, A., Polzehl, J., Spokoiny, V.: Structure adaptive approach for dimension reduction. Ann. Stat. **29**, 1537–1566 (2001a)
Hristache, M., Juditsky, A., Spokoiny, V.: Structure adaptive approach for dimension reduction. Ann. Stat. **29**, 1–32 (2001b)
Ichimura, H.: Semiparametric least squares (SLS) and weighted SLS estimation of Single-Index models. J. Econometric. **58**, 71–120 (1993)
Ichimura, H., Lee, L.-F.: Semiparametric least squares estimation of multiple index models: Single equation estimation. In: Barnett, W.A., Powell, J., Tauchen G. (eds.) Nonparametric and Semiparametric Methods in Econometrics and Statistics. Cambridge University Press, Cambridge 3–49 (1991)
Ishwaran, H.: Identifiability and rates of estimation for scale parameters in location mixture models. Ann. Stat. **24**, 1560–1571 (1996)
Kalbfleisch and Prentice: The Statistical Analysis of Failure Time Data, Wiley, New York (1980)
Kim, J., Pollard, D.: Cube root asymptotics. Ann. Stat. **15**, 541–551 (1990)
Klein, R.W., Spady, R.H.: An efficient semiparametric estimator for binary response models. Econometrica **61**, 387–421 (1993)
Lancaster, T.: Econometric methods for the duration of unemployment. Econometrica **47**, 939–956 (1979)
Lancaster, T.: The incidental parameter problem since 1948. J. Econometric. **95**, 391–413 (2000)
Linton, O.B.: Efficient estimation of additive nonparametric regression models. Biometrika **84**, 469–473 (1997)
Linton, O.B., Härdle, W.: Estimating additive regression models with known links. Biometrika **83**, 529–540 (1996)
Linton, O.B., Nielsen J.P.: A kernel method of estimating structured nonparametric regression based on marginal integration. Biometrika **82**, 93–100 (1995)
Mammen, E., Linton, O.B., Nielsen, J.P.: The existence and asymptotic properties of backfitting projection algorithm under weak conditions. Ann. Stat. **27**, 1443–1490 (1999)
Manski, C.F.: Maximum score estimation of the stochastic utility model of choice. J. Econometric. **3**, 205–228 (1975)

Manski, C.F.: Semiparametric analysis of discrete response: Asymptotic properties of the maximum score estimator. J. Econometric. **27**, 313–333 (1985)
Manski, C.F.: Identification of binary response models. J. Am. Stat. Assoc. **83**, 729–738 (1988)
Matzkin, R.L.: Restrictions of economic theory in nonparametric methods. In: Engle, R.F., McFadden, D.L. (eds.) Handbook of Econometrics. vol. 4, pp. 2523–2558. North-Holland, Amsterdam (1994)
Meyer, B.D.: Unemployment insurance and unemployment spells. Econometrica **58**, 757–782 (1990)
Murphy, S.A.: Consistency in a proportional hazards model incorporating a random effect. Ann. Stat. **22**, 712–731 (1994)
Murphy, S.A.: Asymptotic theory for the frailty model. Ann. Stat. **23**, 182–198 (1995)
Nielsen, G.G., Gill, R.D., Andersen, P.K., Søensen, T.I.A.: A counting process approach to maximum likelihood estimation in frailty models. Scan. J. Stat. **19**, 25–43 (1992)
Powell, J.L.: Estimation of semiparametric models. In: Engle, R.F., McFadden, D.L. (eds.) Handbook of Econometrics. Vol. 4, pp. 2444–2521. North-Holland, Amsterdam (1994)
Powell, J.L., Stock, J.H., Stoker, T.M.: Semiparametric estimation of index coefficients. Econometrica **51**, 1403–1430 (1989)
Ridder, G., Tunali, I.: Stratified partial likelihood estimation. J. Econometric. **92**, 193–232 (1999)
Robinson, P.M.: Root-N-consistent semiparametric regression. Econometrica **56**, 931–954 (1988)
Stock, J.H.: Nonparametric policy analysis. J. Am. Stat. Assoc. **84**, 567–575 (1989)
Stock, J.H.: Nonparametric policy analysis: An application to estimating hazardous waste cleanup benefits. In: Barnett, W.A., Powell, J., Tauchen, G. (eds.) Nonparametric and Semiparametric Methods in Econometrics and Statistics. pp. 77–98. Cambridge University Press, Cambridge (1991)
Tsiatis, A.A.: A large sample study of cox's regression model. Ann. Stat. **9**, 93–108 (1981)
Ye, J., Duan, N.: Nonparametric $n^{-1/2}$-consistent estimation for the general transformation model. Ann. Stat. **25**, 2682–2717 (1997)

Chapter 22
Dimension Reduction Methods

Masahiro Mizuta

22.1 Introduction

One characteristic of computational statistics is the processing of enormous amounts of data. It is now possible to analyze large amounts of high-dimensional data through the use of high-performance contemporary computers. In general, however, several problems occur when the number of dimensions becomes high. The first problem is an explosion in execution time. For example, the number of combinations of subsets taken from p variables is 2^p; when p exceeds 20, calculation becomes difficult pointing terms of computation time. When p exceeds 25, calculation becomes an impossible no matter what type of computer is used. This is a fundamental situation that arises in the selection of explanatory variables during regression analysis. The second problem is the sheer cost of surveys or experiments. When questionnaire surveys are conducted, burden is placed on the respondent because there are many questions. And since there are few inspection items to a patient, there are few the burdens on the body or on cost. The third problem is the essential restriction of methods. When the number of explanatory variables is greater than the data size, most methods are incapable of directly dealing with the data; microarray data are typical examples of this type of data.

For these reasons, methods for dimension reduction without loss of statistical information are important techniques for data analysis. In this chapter, we will explain linear and nonlinear methods for dimension reduction; linear methods reduce dimension through the use of linear combinations of variables, and nonlinear methods do so with nonlinear functions of variables. We will also discuss the reduction of explanatory variables in regression analysis. Explanatory variables can be reduced with several linear combinations of explanatory variables.

M. Mizuta (✉)
Information Initiative Center, Hokkaido University, Sapporo, Japan
e-mail: mizuta@iic.hokudai.ac.jp

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_22,

22.2 Linear Reduction of High-dimensional Data

The p-dimensional data can be reduced into q-dimensional data using q linear combinations of p variables. The linear combinations can be considered as linear projection. Most methods for reduction involve the discovery of linear combinations of variables under set criterion. Principal component analysis (PCA) and projection pursuit are typical methods of this type. These methods will be described in the following subsections.

22.2.1 Principal Component Analysis

Suppose that we have observations of p variables size n; $\{\boldsymbol{x}_i; i = 1, 2, \ldots, n\}$ (referred to as X hereafter). PCA is conducted for the purpose of constructing linear combinations of variables so that their variances are large under certain conditions. A linear combination of variables is denoted by $\{\boldsymbol{a}^\top \boldsymbol{x}_i; i = 1, 2, \ldots, n\}$ (simply, $\boldsymbol{a}^\top \boldsymbol{X}$), where $\boldsymbol{a} = (a_1, a_2, \ldots, a_p)^\top$.

Then, the sample variance of $\boldsymbol{a}^\top \boldsymbol{X}$ can be represented by

$$V\left(\boldsymbol{a}^\top \boldsymbol{x}\right) = \boldsymbol{a}^\top \hat{\Sigma} \boldsymbol{a} \ ,$$

where $\hat{\Sigma} = V(\boldsymbol{X})$. $\boldsymbol{a}^\top \hat{\Sigma} \boldsymbol{a}$ is regarded as a p variable function of $(a_1, a_2, \ldots, a_p)$: $\phi(a_1, a_2, \ldots, a_p) = \boldsymbol{a}^\top \hat{\Sigma} \boldsymbol{a}$. To consider the optimization problem for ϕ, $\boldsymbol{a}$ is constrained to $\boldsymbol{a}^\top \boldsymbol{a} = 1$. This problem is solved using Lagrange multipliers. The following Lagrange function is defined as

$$\begin{aligned} L\left(a_1, a_2, \ldots, a_p\right) &= \phi\left(a_1, a_2, \ldots, a_p\right) - \lambda_1 \left(\sum_{i=1}^{p} a_i^2 - 1 \right) \\ &= \boldsymbol{a}^\top \hat{\Sigma} \boldsymbol{a} - \lambda_1 \left(\boldsymbol{a}^\top \boldsymbol{a} - 1 \right) \ , \end{aligned}$$

where λ is the Lagrange multiplier. L is partially differentiated with respect to $\boldsymbol{a} = (a_1, a_2, \ldots, a_p)^\top$ and λ_1, and the derivatives are equated to zero. We therefore obtain the simultaneous equations:

$$\begin{cases} 2\hat{\Sigma} \boldsymbol{a} - 2\lambda_1 \boldsymbol{a} = 0 \\ \boldsymbol{a}^\top \boldsymbol{a} - 1 = 0 \ . \end{cases}$$

This is an eigenvector problem; the solution to this problem for $\boldsymbol{a} = (a_1, a_2, \ldots, a_p)^\top$ is a unit eigenvector of $\hat{\Sigma}$ corresponding to the largest eigenvalue. Let $\boldsymbol{a}$ be an eigenvector and let λ be an eigenvalue. We then have

$$\phi\left(a_1, a_2, \ldots, a_p\right) = V\left(\boldsymbol{a}^\top \boldsymbol{x}\right) = \boldsymbol{a}^\top \hat{\Sigma} \boldsymbol{a} = \lambda \boldsymbol{a}^\top \boldsymbol{a} = \lambda \ .$$

The eigenvector is denoted as $\boldsymbol{a}_1$. Then $\boldsymbol{a}_1^\top \boldsymbol{x}_i; i = 1, 2, \ldots, n$ are referred to as the first principal components. The first principal components are one-dimensional data that are the projection of the original data with the maximum variance. If all of the information for the data can be represented by the first principal components, further calculation is unnecessary. However, the first principal components usually exhibit the "size factor" only, whereas we would like to obtain another projection, namely the second principal components $\boldsymbol{a}_2^\top \boldsymbol{x}_i$.

The second principal components serve to explain the maximum variance under the constraint and the fact that they are independent of the first principal components. In other words, the second principal components $\boldsymbol{a}_2^\top \boldsymbol{X}$ take the maximum variance under the constraints $\boldsymbol{a}_1^\top \boldsymbol{a}_2 = 0$ and $\boldsymbol{a}_2^\top \boldsymbol{a}_2 = 1$. The second principal components can also be derived with Lagrange multipliers;

$$L(a_1, a_2, \ldots, a_p, \lambda, \lambda_2) = \boldsymbol{a}^\top \hat{\Sigma} \boldsymbol{a} - \lambda \boldsymbol{a}_1^\top \boldsymbol{a} - \lambda_2 (\boldsymbol{a}^\top \boldsymbol{a} - 1) \ .$$

L is partially differentiated with respect to $\boldsymbol{a} = (a_1, a_2, \ldots, a_p)^\top$, λ and λ_2, and the derivatives are equated to zero. The simultaneous equations below are obtained:

$$\begin{cases} 2\hat{\Sigma}\boldsymbol{a} - \lambda \boldsymbol{a}_1 - 2\lambda_2 \boldsymbol{a} = 0 \\ \boldsymbol{a}_1^\top \boldsymbol{a} = 0 \\ \boldsymbol{a}_2^\top \boldsymbol{a}_2 - 1 = 0 \ . \end{cases}$$

We can obtain $\lambda = 0$ and λ_2 is another eigenvalue (not equal to λ_1). Since the variance of $\boldsymbol{a}_2^\top X$ is λ_2, the $\boldsymbol{a}_2$ must be the second largest eigenvalue of $\hat{\Sigma}$. $\{\boldsymbol{a}_2^\top \boldsymbol{x}_i; i = 1, 2, \ldots, n\}$ are referred to as the second principal components. The third principal components, fourth principal components, ..., and the p-th principal components can then be derived in the same manner.

Proportion and Accumulated Proportion

The first principal components through the p-th principal components were defined in the discussions above. As previously mentioned, the variance of the k-th principal components is λ_k. The sum of variances of p variables is $\sum_{j=1}^{p} \hat{\sigma}_j = trace(\hat{\Sigma})$, where $\hat{\Sigma} = (\hat{\sigma}_{ij})$. It is well known that $trace(\hat{\Sigma}) = \sum_{j=1}^{p} \lambda_j$; the sum of the variances coincides with the sum of the eigenvalues. The proportion of the k-th principal components is defined as the proportion of the entire variance to the variance of the k-th principal components:

$$\frac{\lambda_k}{\sum_{j=1}^{p} \lambda_j} \ .$$

The first principal components through the k-th principal components are generally used consecutively. The total variance of these principal components is represented by the accumulated proportion:

$$\frac{\sum_{j=1}^{k} \lambda_j}{\sum_{j=1}^{p} \lambda_j}.$$

We have explained PCA as an eigenvalue problem of covariance matrix. However, the results of this method are affected by units of measurements or scale transformations of variables. Thus, another method is to employ a correlation matrix rather than a covariance matrix. This method is invariant under units of variables, but does not take the variances of the variables into account.

22.2.2 Projection Pursuit

PCA searches a lower dimensional space that captures the majority of the variation within the data and discovers linear structures in the data. This method, however, is ineffective in analyzing nonlinear structures, i.e. curves, surfaces or clusters. In 1974, Friedman and Tukey (1974) proposed projection pursuit to search for linear projection onto the lower dimensional space that robustly reveals structures in the data. After that, many researchers developed new methods for projection pursuit and evaluated them (e.g. Friedman 1987; Hall 1989; Huber 1985; Iwasaki 1991; Koyama et al. 1998; Nason 1995). The fundamental idea behind projection pursuit is to search linear projection of the data onto a lower dimensional space their distribution is "interesting"; "interesting" is defined as being "far from the normal distribution", i.e. the normal distribution is assumed to be the most uninteresting. The degree of "far from the normal distribution" is defined as being a projection index, the details of which will be described later.

Algorithm

The use of a projection index makes it possible to execute projection pursuit with the projection index. Here is the fundamental algorithm of k-dimensional projection pursuit.

1. Sphering $\boldsymbol{x}$: $\boldsymbol{z}_i = \hat{\Sigma}_{xx}^{-1/2}(\boldsymbol{x}_i - \hat{\boldsymbol{x}})$ $(i = 1, 2, \ldots, n)$, where $\hat{\Sigma}$ is the sample covariance matrix and $\hat{\boldsymbol{x}}$ is the sample mean of $\boldsymbol{x}$.
2. Initialize the project direction: $\alpha = (\boldsymbol{\alpha}_1, \boldsymbol{\alpha}_2, \ldots, \boldsymbol{\alpha}_k)$.
3. Search the direction α that maximizes the projection index.
4. Project the data onto the lower dimensional space and display or analyze them.
5. Change the initial direction and repeat Steps 3 and 4, if necessary.

Projection Indexes

The goal of projection pursuit is to find a projection that reveals *interesting* structures in the data. There are various standards for interestingness, and it is a very difficult task to define. Thus, the normal distribution is regarded as uninteresting, and uninterestingness is defined as a degree that is "far from the normal distribution."

Projection indexes are defined as of this degree. There are many definitions for projection indexes. Projection pursuit searches projections based on the projection index; methods of projection pursuit are defined by the projection indexes.

Here we will present several projection indexes. It is assumed that $\boldsymbol{Z} = (z_1, \ldots, z_n)$ is the result of sphering $\boldsymbol{X}$; the mean vector is a zero vector and the covariance matrix is an identity matrix.

Friedman's Index

Friedman (1987) proposed the following projection index:

$$I = \frac{1}{2}\sum_{j=1}^{J}(2j+1)\left[\frac{1}{n}\sum_{i=1}^{n} P_j\left(2\Phi\left(\boldsymbol{\alpha}^\top \boldsymbol{Z}_i\right)-1\right)\right]^2 ,$$

where $P_j(\cdot)$ are Legendre polynomials of order j and $\Phi(\cdot)$ is the cumulative distribution function of the normal distribution and J is a user-defined constant number, i.e. the degree of approximation.

In the case of two-dimensional projection pursuit, the index is represented by

$$\begin{aligned} I = &\sum_{j=1}^{J}(2j+1)E^2[P_j(R_1)]/4 \\ &+ \sum_{k=1}^{J}(2k+1)E^2\left[P_k(R_2)\right]/4 \\ &+ \sum_{j=1}^{J}\sum_{k=1}^{J-j}(2j+1)(2k+1)E^2\left[P_j(R_1)P_k(R_2)\right]/4 , \end{aligned}$$

where

$$\begin{aligned} X_1 = \boldsymbol{\alpha}_1^\top \boldsymbol{Z} , \quad X_2 = \boldsymbol{\alpha}_2^\top \boldsymbol{Z} \\ R_1 = 2\Phi\left(X_1\right) - 1 , \quad R_2 = 2\Phi\left(X_2\right) - 1 . \end{aligned}$$

Moment Index

The third and higher cumulants of the normal distribution vanish. The cumulants are sometimes used for the test of normality, i.e. they can be used for the projection index. Jones and Sibson (1987) proposed a one-dimensional projection index named the "moment index," with the third cumulant $k_3 = \mu_3$ and the fourth cumulant $k_4 = \mu_4 - 3$:

$$I = k_3^2 + \frac{1}{4}k_4^2 .$$

For two-dimensional projection pursuit, the moment index can be defined as

$$I = \left(k_{30}^2 + 3k_{21}^2 + 3k_{12}^2 + k_{03}^2\right) + \frac{1}{4}\left(k_{40}^2 + 4k_{31}^2 + 6k_{22}^2 + 4k_{13}^2 + k_{04}^2\right) .$$

Hall's Index.

Hall (1989) proposed the following projection index:

$$I = \left[\boldsymbol{\theta}_0(\boldsymbol{\alpha}) - 2^{-1/2}\pi^{-1/4}\right]^2 + \sum_{j=1}^{J} \boldsymbol{\theta}_j^2(\boldsymbol{\alpha}) ,$$

where

$$\boldsymbol{\theta}_j(\boldsymbol{\alpha}) = n^{-1} \sum_{i=1}^{n} P_j\left(\boldsymbol{\alpha}^\top \boldsymbol{Z}_i\right) \phi\left(\boldsymbol{\alpha}^\top \boldsymbol{Z}_i\right) ,$$

$$P_j(z) = \frac{\sqrt{2}}{\sqrt{j!}} \pi^{1/4} H_j\left(2^{1/2} z\right) ,$$

$\phi(z)$ is the normal density function and $H_j(z)$ are the Hermite polynomials of degree j. J is a user-defined constant number. Hall's index is much more robust for outliers than Freidman's index.

Relative Projection Pursuit

The main objective of ordinary projection pursuit is the discovery of non-normal structures in a dataset. Non-normality is evaluated using the degree of difference between the distribution of the projected dataset and the normal distribution.

There are times in which it is desired that special structures be discovered using criterion other than non-normal criterion. For example, if the purpose of analysis is to investigate a feature of a subset of the entire dataset, the projected direction should be searched so that the projected distribution of the subset is far from the distribution of the entire dataset. In sliced inverse regression (please refer to the

final subsection of this chapter), the dataset is divided into several subsets based on the values of the response variable, and the effective dimension-reduction direction is searched for using projection pursuit. In this application of projection pursuit, projections for which the distributions of the projected subsets are far from those of the entire dataset are required. Mizuta (2002) proposed the adoption of relative projection pursuit for these purposes. Relative projection pursuit finds *interesting* low-dimensional space that differs from the reference dataset predefined by the user.

22.3 Nonlinear Reduction of High-dimensional Data

In the previous section, we discussed linear methods i.e. methods for dimension reduction through the use of linear projections. We will now move on to nonlinear methods for dimension reduction. First, we will describe a generalized principal component analysis (GPCA) method that is a nonlinear extension of PCA. Algebraic curve fitting methods will then be mentioned for a further extension of GPCA. Finally, we will introduce principal curves i.e. the parametric curves that pass through the middle of the data.

22.3.1 Generalized Principal Component Analysis

As long as data have a near-linear structure, the singularities of the data can be pointed out using PCA. On the contrary, if data have a nonlinear structure, GPCA will not be adequate for drawing conclusions regarding the nature of the data. To overcome this difficulty, GPCA has been proposed by Gnanadesikan and Wilk (1969), whereby fitting functions to the data points can be discovered.

Suppose that we have observations of p variables $\boldsymbol{x} = (x_1, x_2, \ldots, x_p)$ on each of n individuals. Let $f_i(\boldsymbol{x})(i = 1, 2, \ldots, k)$ be k real-valued functions of the original variables.

The aim of GPCA is to discover a new set of variables (or functions of $\boldsymbol{x}$), as denoted by $z_1, z_2, \ldots, z_k$, which are mutually uncorrelated and whose variances decrease, from first to last. Each $z_j(j = 1, 2, \ldots, k)$ is considered to be a linear combination of $f_i(\boldsymbol{x})(i = 1, 2, \ldots, k)$, so that

$$z_j = \sum_{i=1}^{k} l_{ij} f_i(\boldsymbol{x}) = \boldsymbol{l}_j^\top \boldsymbol{f}(\boldsymbol{x}) ,$$

where $\boldsymbol{l}_j = (l_{1j}, l_{2j}, \ldots, l_{kj})^\top$ are k constant vectors such that $\boldsymbol{l}_j^\top \boldsymbol{l}_j = 1$, and $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x}))^\top$. The vectors $\boldsymbol{l}_1, \boldsymbol{l}_2, \ldots, \boldsymbol{l}_k$ are the eigenvectors of the covariance matrix of $(f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x}))$, as in PCA. The function z_k defined by the "smallest" eigenvalue is considered to be one of the fitting functions to the data.

PCA is a special case of GPCA: real-valued functions $f_i(\boldsymbol{x})$ are reduced to $x_i (i = 1, 2, \ldots, p)$.

Quadratic principal component analysis (QPCA) is specified by the following functions:

$$\begin{cases} f_i(\boldsymbol{x}) = x_i & (i = 1, 2, \ldots, p) \\ f_i(\boldsymbol{x}) = x_j x_m & \left(i = p+1, \ldots, \left(p^2 + 3p\right)/2\right) \end{cases},$$

where j, m is uniquely determined by

$$i = \{(2p - j + 3)\, j/2\} + m - 1 ,$$
$$1 \leq j \leq m \leq p ,$$

for $i\,(i = p + 1, \ldots, (p^2 + 3p)/2)$.

QPCA for two dimensional data is defined by

$$f_1(x, y) = x$$
$$f_2(x, y) = y$$
$$f_3(x, y) = x^2$$
$$f_4(x, y) = xy$$
$$f_5(x, y) = y^2 .$$

Most GPCA methods are not invariant under orthogonal transformations and/or the translations (parallel transformations) of a coordinate system, though PCA is invariant under them. For example, QPCA is not invariant under them. The expression "the method is invariant" in this subsection means that the results of the method are never changed in the original coordinate by coordinate transformation. In the following, the determination of the GPCA methods that are invariant under the orthogonal transformations of a coordinate system will be described in the case of two variables. Translations of a coordinate system are disregarded here because the data can be standardized to have a zero mean vector.

Hereafter, let us assume the following conditions:

A1 $f_1(\boldsymbol{x}), f_2(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x})$ are linearly independent as functions of $\boldsymbol{x}$.

A2 For any orthogonal matrix T, there is a matrix W such that $\boldsymbol{f}(T\boldsymbol{x}) \equiv W\boldsymbol{f}(\boldsymbol{x})$.

A3 $f_i(\boldsymbol{x})$ are continuous functions.

Conditions **A1** and **A3** may be proper for GPCA, and condition **A2** is necessary for discussing the influence of orthogonal coordinate transformations. PCA and QPCA clearly satisfy these conditions.

A GPCA method is referred to as "invariant" if its results in the original coordinate system are not changed by the orthogonal transformation of a coordinate

system. It can be mathematically described as follows. For any orthogonal coordinate transformation: $\boldsymbol{x}^* = T\boldsymbol{x}$,

$$\begin{aligned} z_j^* &= \boldsymbol{l}_j^{*\top} \boldsymbol{f}(\boldsymbol{x}^*) \\ &= \boldsymbol{l}_j^{*\top} \boldsymbol{f}(T\boldsymbol{x}) \quad (j = 1, 2, \ldots, k) \end{aligned}$$

denote the results of the method for transformed variables $\boldsymbol{x}^*$, where $\boldsymbol{l}_j^*$ are eigenvectors of $\mathrm{Cov}(\boldsymbol{f}(\boldsymbol{x}^*))$. The method is "invariant" if it holds that

$$\boldsymbol{l}_j^{\top} \boldsymbol{f}(\boldsymbol{x}) \equiv \pm \boldsymbol{l}_j^{*\top} \boldsymbol{f}(T\boldsymbol{x}) \quad (j = 1, 2, \ldots, k)$$

as vector-valued functions of $\boldsymbol{x}$ for any orthogonal matrix T. The plus or minus sign is indicated only for the orientations of the eigenvectors.

The GPCA method specified by $\boldsymbol{f}(\boldsymbol{x})$ is invariant under an orthogonal transformation, if and only if the matrix W is an orthogonal matrix for any orthogonal matrix T. The proof will be described below. If the method is invariant, W can be taken as

$$\left(\boldsymbol{l}_1^*, \boldsymbol{l}_2^*, \ldots, \boldsymbol{l}_k^*\right) \left(\boldsymbol{l}_1, \boldsymbol{l}_2, \ldots, \boldsymbol{l}_k\right)^{\top} ,$$

which is an orthogonal matrix. Conversely, if W is an orthogonal matrix, $W^{\top}\boldsymbol{l}_j^*$ are eigenvectors of $\mathrm{Cov}(\boldsymbol{f}(\boldsymbol{x}))$. Therefore the following is obtained:

$$\boldsymbol{l}_j^{\top} = \pm \boldsymbol{l}_j^{*\top} W \ .$$

Mizuta (1983) derived a theorem on invariant GPCA.

Theorem 1. *GPCA methods for two-dimensional data* (x, y) *under the conditions* ***A1***, ***A2*** *and* ***A3*** *that are invariant under rotations can be restricted to those specified by the following functions.*

(1) s pairs of functions:

$$\begin{cases} f_{2i-1}(x,y) = g_i\left(\sqrt{x^2+y^2}\right)\left(x^{N_i} - \binom{N_i}{2} y^2 x^{N_i-2} + \binom{N_i}{4} y^4 x^{N_i-4} - \ldots\right) \\ \qquad\qquad -h_i\left(\sqrt{x^2+y^2}\right)\left(N_i y x^{N_i-1} - \binom{N_i}{3} y^3 x^{N_i-3} + \binom{N_i}{5} y^5 x^{N_i-5} - \ldots\right) \\ f_{2i}(x,y) \quad = g_i\left(\sqrt{x^2+y^2}\right)\left(N_i y x^{N_i-1} - \binom{N_i}{3} y^3 x^{N_i-3} + \binom{N_i}{5} y^5 x^{N_i-5} - \ldots\right) \\ \qquad\qquad +h_i\left(\sqrt{x^2+y^2}\right)\left(x^{N_i} - \binom{N_i}{2} y^2 x^{N_i-2} + \binom{N_i}{4} y^4 x^{N_i-4} - \ldots\right) \end{cases}$$

$$(i = 1, 2, \ldots, s) \ ,$$

where g_i, h_i *are arbitrary continuous functions of* $\sqrt{x^2+y^2}$ *and* N_i *are arbitrary positive integers.*

(2) Continuous functions of $\sqrt{x^2+y^2}$.

The above theorem can be extended for use with GPCA methods for p-dimensional data because invariant GPCA for p-dimensional data methods are invariant under the rotations of any pair of two variables and the reverse is also true.

We will show some set of functions for invariant GPCA here.

(1) 3 dimensional and degree 1:

$$x, y, z \ .$$

(2) 3 dimensional and degree 2:

$$x^2, y^2, z^2, \sqrt{2}xy, \sqrt{2}yz, \sqrt{2}zx \ .$$

(3) 3 dimensional and degree 3:

$$x^3, y^3, z^3, \sqrt{3}x^2y, \sqrt{3}y^2z, \sqrt{3}z^2x, \sqrt{3}xy^2, \sqrt{3}yz^2, \sqrt{3}zx^2, \sqrt{6}xyz \ .$$

(4) 3 dimensional and degree q:

$$\sqrt{\frac{q!}{i!j!k!}} x^i y^j z^k$$

$$(i + j + k = q \ ; \quad 0 \leq i, j, k) \ .$$

(5) p dimensional and degree q:

$$\sqrt{\frac{q!}{\prod_{i=1}^{p} k_t!}} \prod_{t=1}^{p} (x_t)_t^k$$

$$\sum_{t=1}^{p} k_t = q \ ; \quad 0 \leq k_t \ .$$

22.3.2 Algebraic Curve and Surface Fitting

Next, we will discuss a method involving algebraic curve and surface fitting to multidimensional data.

The principal component line minimizes the sum of squared deviations in each of the variables. The PCA cannot find non-linear structures in the data. GPCA is used to discover an algebraic curve fitted to data; the function z_k defined by the "smallest" eigenvalue is considered to be one of the fitting functions to the data. However, it is difficult to interpret algebraic curves statistically derived form GPCA.

We will now describe methods for estimating the algebraic curve or surface that minimizes the sum of squares of perpendicular distances from multidimensional data.

Taubin (1991) developed an algorithm for discovering the algebraic curve for which the sum of *approximate* squares distances between data points and the curve is minimized. The approximate squares distance does not always agree with the *exact* squares distance. Mizuta (1995, 1996) presented an algorithm for evaluating the exact distance between the data point and the curve, and have presented a method for algebraic curve fitting with exact distances. In this subsection, we describe the method of algebraic surface fitting with exact distances. The method of the algebraic curve fitting is nearly identical to that of surface fitting, and is therefore omited here.

Algebraic Curve and Surface

A p-dimensional algebraic curve or surface is the set of zeros of k-polynomials $\boldsymbol{f}(\boldsymbol{x}) = (f_1(\boldsymbol{x}), \ldots, f_k(\boldsymbol{x}))$ on $\boldsymbol{R}^p$,

$$Z(\boldsymbol{f}) = \{\boldsymbol{x} : \boldsymbol{f}(\boldsymbol{x}) = 0\} \ .$$

In the case of $p = 2$ and $k = 1$, $Z(f)$ is a curve in the plane. For example, $Z(x^2 + 2y^2 - 1)$ is an ellipse and $Z(y^2 - x^2 + 1)$ is a hyperbola. In the case of $p = 3$ and $k = 2$, $Z(\boldsymbol{f})$ is a curve in the space.

In the case of $p = 3$ and $k = 1$, $Z(f)$ is a surface:

$$Z(f) = \{(x, y, z) : f(x, y, z) = 0\} \ .$$

Hereafter, we will primarily discuss this case.

Approximate Distance

The distance from a point $\boldsymbol{a}$ to the surface $Z(f)$ is usually defined by

$$\mathrm{dist}\,(\boldsymbol{a}, Z(f)) = \inf(\| \boldsymbol{a} - \boldsymbol{y} \| : \boldsymbol{y} \in Z(f)) \ .$$

It was said that the distance between a point and the algebraic curve or surface cannot be computed using direct methods. Thus, Taubin proposed *an approximate distance* from $\boldsymbol{a}$ to $Z(f)$ (Taubin 1991). The point $\hat{\boldsymbol{y}}$ that approximately minimizes the distance $\| \boldsymbol{y} - \boldsymbol{a} \|$, is given by

$$\hat{\boldsymbol{y}} = \boldsymbol{a} - \left(\nabla f(\boldsymbol{a})^\top\right)^+ f(\boldsymbol{a}) \ ,$$

where $(\nabla f(\boldsymbol{a})^\top)^+$ is the pseudoinverse of $\nabla f(\boldsymbol{a})^\top$. The distance from $\boldsymbol{a}$ to $Z(f)$ is approximated to

$$\mathrm{dist}\,(\boldsymbol{a}, Z(f))^2 \approx \frac{f(\boldsymbol{a})^2}{\| \nabla f(\boldsymbol{a}) \|^2} .$$

Taubin also presented an algorithm to find the algebraic curve for which the sum of *approximate* squares distances between data points and the curve is minimized.

Exact Distance

In the following, we present a method for calculating the distance between a point $\boldsymbol{a} = (\alpha, \beta, \gamma)$ and an algebraic surface $Z(f)$.

If (x, y, z) is the nearest point to the point $\boldsymbol{a} = (\alpha, \beta, \gamma)$ on $Z(f)$, (x, y, z) satisfies the following simultaneous equations:

$$\begin{cases} \phi_1(x, y, z) = 0 \\ \phi_2(x, y, z) = 0 \\ f(x, y, z) = 0 , \end{cases} \tag{22.1}$$

where $\phi_1(x, y, z) = (x - \alpha)(\partial f/\partial y) - (y - \beta)(\partial f/\partial x)$, and $\phi_2(x, y, z) = (z - \gamma)(\partial f/\partial y) - (y - \beta)(\partial f/\partial z)$.

Equations (22.1) can be solved using the Newton–Rapson method:

1. Set x_0, y_0 and z_0 (see below).
2. Solve the equations:

$$\begin{cases} h\dfrac{\partial \phi_1}{\partial x} + k\dfrac{\partial \phi_1}{\partial y} + l\dfrac{\partial \phi_1}{\partial z} = -\phi_1(x, y, z) \\ h\dfrac{\partial \phi_2}{\partial x} + k\dfrac{\partial \phi_2}{\partial y} + l\dfrac{\partial \phi_2}{\partial z} = -\phi_2(x, y, z) \\ h\dfrac{\partial f}{\partial x} + k\dfrac{\partial f}{\partial y} + l\dfrac{\partial f}{\partial z} = -f(x, y, z) . \end{cases} \tag{22.2}$$

3. Replace x, y:

$$\begin{cases} x_{i+1} = x_i + h \\ y_{i+1} = y_i + k \\ z_{i+1} = z_i + l . \end{cases}$$

4. Stop if $h^2 + k^2 + l^2$ is below a certain threshold. Otherwise, go to Step 2.

One of the important points to consider when applying the Newton–Rapson method is to compute an initial point. We have a good initial point: (α, β, γ).

When $x_0 = \alpha$, $y_0 = \beta$, $z_0 = \gamma$, (22.2) are

$$\begin{cases} h\dfrac{\partial \phi_1}{\partial x} + k\dfrac{\partial \phi_1}{\partial y} + l\dfrac{\partial \phi_1}{\partial z} = 0 \\ h\dfrac{\partial \phi_2}{\partial x} + k\dfrac{\partial \phi_2}{\partial y} + l\dfrac{\partial \phi_2}{\partial z} = 0 \\ h\dfrac{\partial f}{\partial x} + k\dfrac{\partial f}{\partial y} + l\dfrac{\partial f}{\partial z} = -f(x, y, z) \ . \end{cases}$$

It is very simple to show that the distance between (x_1, y_1, z_1) and (α, β, γ) agrees with Taubin's approximate distance.

Algebraic Surface Fitting

We have already described the method for calculating the distance between a point and a surface.

The problem of finding a fitting surface that minimizes the sum of the distances from data points can therefore be solved by using an optimization method without derivatives. However, for computing efficiency, the partial derivatives of the sum of squares of distances from data with the coefficients of an algebraic curve are derived.

In general, a polynomial f in a set is denoted by

$$f\left(b_1, \ldots, b_q; x, y, z\right) \ ,$$

where $b_1, \ldots, b_q$ are the parameters of the set.

Let $\boldsymbol{a}_i = (\alpha_i, \beta_i, \gamma_i)(i = 1, 2, \ldots, n)$ be n data points within the space. The point in $Z(f)$ that minimizes the distance from $(\alpha_i, \beta_i, \gamma_i)$ is denoted by $(x_i, y_i, z_i)(i = 1, 2, \ldots, n)$.

The sum of squares of distances is

$$R = \sum_{i=1}^{n} (\boldsymbol{x}_i - \boldsymbol{a}_i)^\top (\boldsymbol{x}_i - \boldsymbol{a}_i) \ .$$

R can be minimized with respect to the parameters of polynomial f with the *Levenberg–Marquardt Method*. This method requires partial derivatives of R with respect to b_j:

$$\frac{\partial R}{\partial b_j} = \sum_{i=1}^{n} \frac{\partial R_i}{\partial b_j} \ , \tag{22.3}$$

where

$$\frac{\partial R_i}{\partial b_j} = 2\left((x_i - \alpha_i)\frac{\partial x_i}{\partial b_j} + (y_i - \beta_i)\frac{\partial y_i}{\partial b_j} + (z_i - \gamma_i)\frac{\partial z_i}{\partial b_j} \right) . \tag{22.4}$$

The only matter left to discuss is a solution for $\partial x_i/\partial b_j, \partial y_i/\partial b_j$ and $\partial z_i/\partial b_j$. Hereafter, the subscript i is omitted. By the derivative of both sides of $f(b_1, \ldots, b_q, x, y, z) = 0$ with respect to b_j $(j = 1, \ldots, q)$, we obtain

$$\frac{\partial f}{\partial x}\frac{\partial x}{\partial b_j} + \frac{\partial f}{\partial y}\frac{\partial y}{\partial b_j} + \frac{\partial f}{\partial z}\frac{\partial z}{\partial b_j} + \frac{\mathrm{d}f}{\mathrm{d}b_j} = 0 \,, \tag{22.5}$$

where $\mathrm{d}f/\mathrm{d}b_j$ is the differential of f with b_j when x and y are fixed.

Since $\boldsymbol{x}_i$ is on the normal line from $\boldsymbol{a}_i$,

$$\left(\left.\frac{\partial f}{\partial x}\right|_{\boldsymbol{x}_i}, \left.\frac{\partial f}{\partial y}\right|_{\boldsymbol{x}_i}, \left.\frac{\partial f}{\partial z}\right|_{\boldsymbol{x}_i} \right)^{\top} (\boldsymbol{x}_i - \boldsymbol{a}_i) = 0 \,.$$

By the derivative of

$$(y-\beta)(z-\gamma)\left.\frac{\partial f}{\partial x}\right|_{\boldsymbol{x}} = t$$

$$(x-\alpha)(z-\gamma)\left.\frac{\partial f}{\partial y}\right|_{\boldsymbol{x}} = t$$

$$(x-\alpha)(y-\beta)\left.\frac{\partial f}{\partial z}\right|_{\boldsymbol{x}} = t$$

with respect to b_j, we obtain the linear combinations of $\partial x/\partial b_j, \partial y/\partial b_j$ and $\partial z/\partial b_j$:

$$c_{1m}\frac{\partial x}{\partial b_j} + c_{2m}\frac{\partial y}{\partial b_j} + c_{3m}\frac{\partial z}{\partial b_j} + c_{4m} = \frac{\partial t}{\partial b_j} \,, \tag{22.6}$$

where $c_{1m}, \ldots, c_{4m}$ are constants $(m = 1, \ldots, 3)$.

Equations (22.5) and (22.6) are simultaneous linear equations in four variables $\partial x/\partial b_j, \partial y/\partial b_j, \partial z/\partial b_j$ and $\partial t/\partial b_j$. We then obtain $\partial x/\partial b_j, \partial y/\partial b_j$ and $\partial z/\partial b_j$ at (x_i, y_i, z_i). By (22.4), we have the partial differentiation of R_i with respect to b_j.

Therefore, we can obtain the algebraic curve that minimizes the sum of squares of distances from data points with the Levenberg–Marquardt method.

Bounded and Stably Bounded Algebraic Curve and Surface

Although algebraic curves can fit the data very well, they usually contain points far remote from the given data set. In 1994, Keren et al. (1994) and Taubin et al. (1994) independently developed algorithms for a *bounded* (closed) algebraic curve with approximate squares distance. We will now introduce the definition and properties of a bounded algebraic curve.

$Z(f)$ is referred to as *bounded* if there exists a constant r such that $Z(f) \subset \{\boldsymbol{x} : \| \boldsymbol{x} \| < r\}$. For example, it is clear that $Z(x^2 + y^2 - 1)$ is bounded, but $Z(x^2 - y^2)$ is not bounded.

Keren et al. (1994) defined $Z(f)$ to be *stably bounded* if a small perturbation of the coefficients of the polynomial leaves its zero set bounded. An algebraic curve $Z((x-y)^4 + x^2 + y^2 - 1)$ is bounded but not stably bounded because $Z((x-y)^4 + x^2 + y^2 - 1 + \varepsilon x^3)$ is not bounded for any $\varepsilon \neq 0$.

Let $f_k(x, y)$ be the form of degree k of a polynomial $f(x, y)$: $f(x, y) = \sum_{k=0}^{d} f_k(x, y)$. The leading form of a polynomial $f(x, y)$ of degree d is defined by $f_d(x, y)$. For example, the leading form of $f(x, y) = x^2 + 2xy - y^2 + 5x - y + 3$ is $f_2(x, y) = x^2 + 2xy - y^2$.

Lemma 1. *For an even positive integer d, any leading form $f_d(x, y)$ can be represented by $\boldsymbol{x}^\top A \boldsymbol{x}$. Where A is a symmetric matrix and $\boldsymbol{x} = (x^{d/2}, x^{d/2-1}y, \ldots, xy^{d/2-1}, y^{d/2})^\top$.*

Theorem 2. *(Keren et al. 1994): The $Z(f)$ is stably bounded if and only if d is even and there exists a symmetric positive definite matrix A such that*

$$f_d(x, y) = \boldsymbol{x}^\top A \boldsymbol{x} \ ,$$

where $\boldsymbol{x} = (x^{d/2}, x^{d/2-1}y, \ldots, xy^{d/2-1}, y^{d/2})^\top$.

These definitions and theorem for algebraic curves are valid for algebraic surfaces. Hereafter, we will restrict our discussion to algebraic *surfaces*.

Parameterization

We parameterize the set of all polynomials of degree k and the set of polynomials that induce (stably) bounded algebraic surfaces. In general, a polynomial f of degree p with q parameters can be denoted by $f(b_1, \ldots, b_q; x, y)$, where $b_1, \ldots, b_q$ are the parameters of the polynomial.

For example, *all of the polynomials* of degree 2 can be represented by

$$f\ (b_1, b_2, \ldots, b_{10}; x, y, z) = B^\top X \ ,$$

where $X = (1, x, y, z, x^2, y^2, z^2, xy, yz, zx)^\top$, $B = (b_1, b_2, \ldots, b_{10})^\top$.

For *stably bounded algebraic curves* of degree 4,

$$f\ (b_1, \ldots, b_{41}; x, y, z) = \left(x^2, y^2, z^2, xy, yz, zx\right) A^2 \left(x^2, y^2, z^2, xy, yz, zx\right)^\top + (b_{22}, \ldots, b_{41}) \left(1, x, y, z, \ldots, z^3\right)^\top \ ,$$

where

$$A = \begin{pmatrix} b_1 & b_2 & b_3 & b_4 & b_5 & b_6 \\ b_2 & b_7 & b_8 & b_9 & b_{10} & b_{11} \\ b_3 & b_8 & b_{12} & b_{13} & b_{14} & b_{15} \\ b_4 & b_9 & b_{13} & b_{16} & b_{17} & b_{18} \\ b_5 & b_{10} & b_{14} & b_{17} & b_{19} & b_{20} \\ b_6 & b_{11} & b_{15} & b_{18} & b_{20} & b_{21} \end{pmatrix}.$$

Examples

Here we will show a numerical example of the algebraic surface and bounded algebraic surface fitting methods.

The data in this example is three-dimensional data of size 210. The 210 points nearly lie on a closed cylinder (Fig. 22.1). The result of GPCA is set for an initial surface and the method is used to search for a fitting algebraic surface of degree 4 (Figs. 22.2, 22.3 and 22.4). The value of R is 0.924.

Figure 22.5 presents the result of a *bounded* algebraic surface fitting the same data. The value of R is 1.239, and is greater than that of unbounded fitting. The bounded surface, however, directly reveals the outline of the data.

In this subsection, we have discussed algebraic surface fitting to multidimensional data. Two sets of algebraic surfaces were described: an unbounded algebraic surface and a bounded algebraic surface. This method can be extended for use with any other family of algebraic surfaces.

Taubin (1994) proposed the approximate distance of order k and presented algorithms for rasterizing algebraic curves. The proposed algorithm for exact distance can also be used for rasterizing algebraic curves and surfaces. Mizuta (1997) has successfully developed a program for rasterizing them with exact distances.

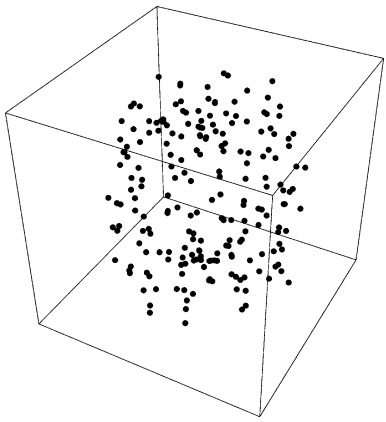


Fig. 22.1 Surface fitting for distributed cylinder data (Original Data Points)

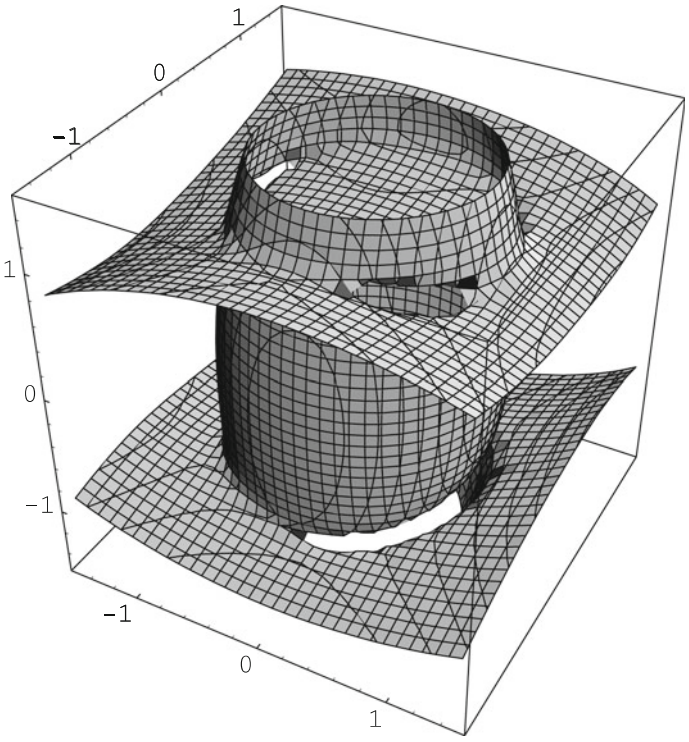


Fig. 22.2 Surface fitting for distributed cylinder data (Unbounded Fitting Surface)

22.3.3 Principal Curves

Curve fitting to data is an important method for data analysis. When we obtain a fitting curve for data, the dimension of the data is nonlinearly reduced to one dimension. Hastie and Stuetzle (1989) proposed the concept of a principal curve and developed a concrete algorithm to find the principal curve, which is represented by a parametric curve. We can therefore obtain a new nonlinear coordinate for the data using the principal curve.

Definition of Principal Curve

First, we will define principal curves for a p-dimensional distribution function $h(\boldsymbol{x})(\boldsymbol{x} \in R^p)$, rather than a dataset.

The expectation of X with density function h in R^p is denoted by $E_h(X)$. The parametric curve within the p-dimensional space is represented by $\boldsymbol{f}(\lambda)$, where λ is the parameter.

For each point $\boldsymbol{x}$ in R^p, the parameter λ of the nearest point on the curve $\boldsymbol{f}(\lambda)$ is denoted by $\lambda_{\boldsymbol{f}}(\boldsymbol{x})$, which is referred to as the *projection index*. The projection

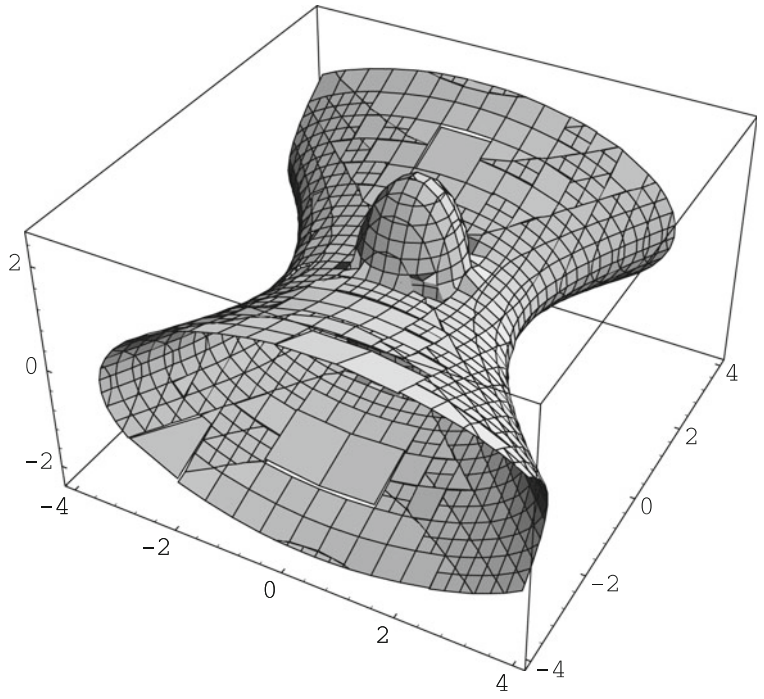


Fig. 22.3 Surface fitting for distributed cylinder data (Global View of 2)

index, which is different from projection index in projection pursuit, is defined as follows:

$$\lambda_f(\boldsymbol{x}) = \sup_{\lambda} \left\{ \lambda | \parallel \boldsymbol{x} - \boldsymbol{f}(\lambda) \parallel = \inf_{\mu} \parallel \boldsymbol{x} - \boldsymbol{f}(\mu) \parallel \right\} .$$

The curve $\boldsymbol{f}(\lambda)$ is referred to as the principal curve of density function h, if

$$E_h \left(\boldsymbol{x} \parallel \lambda_f(\boldsymbol{x}) = \lambda \right) = \boldsymbol{f}(\lambda) \quad \text{(for a.e. } \lambda)$$

is satisfied. After all, for any point $\boldsymbol{f}(\lambda)$ on the curve, the average of the conditional distribution of $\boldsymbol{x}$ given $\lambda_f(\boldsymbol{x}) = \lambda$ is consistent with $\boldsymbol{f}(\lambda)$ with the exception of a set of measure 0.

The principal curves of a given distribution are not always unique. For example, two principal components of the two-dimensional normal distribution are principal curves.

The algorithm for finding the principal curves of a distribution is:

1. *Initialization.* Put

$$\boldsymbol{f}^{(0)}(\lambda) = \bar{\boldsymbol{x}} + \boldsymbol{a}\lambda ,$$

where $\boldsymbol{a}$ is the first principal component of the distribution defined by the density function h and $\bar{\boldsymbol{x}}$ is the average of $\boldsymbol{x}$.

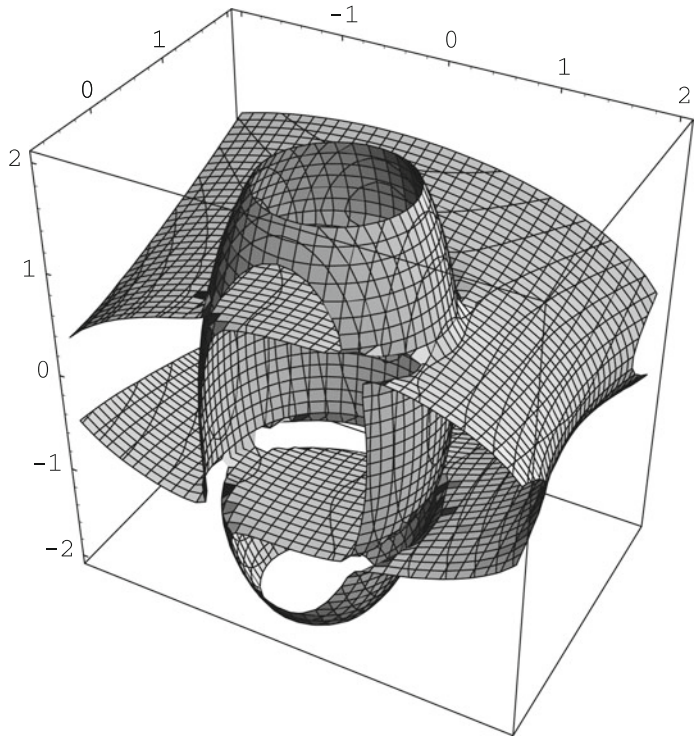


Fig. 22.4 Surface fitting for distributed cylinder data (Cutting View of 2)

2. *Expectation Step* (update of $\boldsymbol{f}(\lambda)$).

$$\boldsymbol{f}^{(j)}(\lambda) = E\left(\boldsymbol{x}|\lambda_{\boldsymbol{f}^{(j-1)}}(\boldsymbol{x}) = \lambda\right) \quad \forall\lambda$$

3. *Projection Step* (update of λ).

$$\lambda^{(j)}(\boldsymbol{x}) = \lambda_{\boldsymbol{f}^{(j)}}(\boldsymbol{x}) \quad \forall \boldsymbol{x} \in R^p$$

And transform the $\lambda^{(j)}$ to be arc length.

4. *Evaluation.* Calculate

$$D^2\left(h, \boldsymbol{f}^{(j)}\right) = E_{\lambda^{(j)}} E\left\{\| \boldsymbol{x} - \boldsymbol{f}\left(\lambda^{(j)}(\boldsymbol{x})\right) \|^2 \, |\lambda^{(j)}(\boldsymbol{x})\right\} .$$

If the value

$$\frac{\left|D^2\left(h, \boldsymbol{f}^{(j-1)}\right) - D^2\left(h, \boldsymbol{f}^{(j)}\right)\right|}{D^2\left(h, \boldsymbol{f}^{(j-1)}\right)}$$

is smaller than ε, then stop, otherwise $j = j + 1$ and go to Step 1.

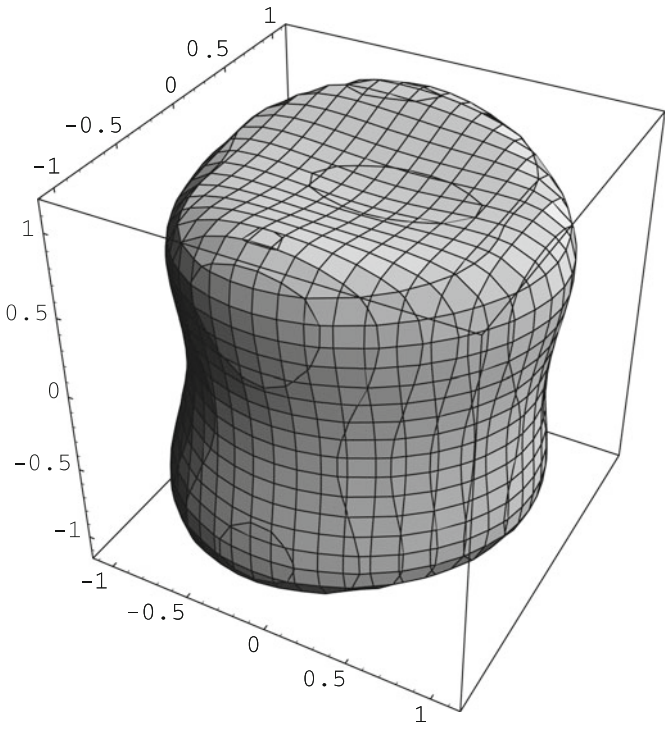


Fig. 22.5 Surface fitting for distributed cylinder data (Bounded Fitting Surface)

In the Expectation Step, calculate the expectation with respect to the distribution h of the set of $\boldsymbol{x}$ satisfying $\lambda_{\boldsymbol{f}^{(j-1)}}(\boldsymbol{x}) = \lambda$ and substitute $\boldsymbol{f}^{(j)}(\lambda)$ for it. In the Projection Step, project data points in R^p to the curve $\boldsymbol{f}^{(j)}(\lambda)$ and assign $\lambda^{(j)}(\boldsymbol{x})$.

For actual data analysis, only a set of data points is given and the distribution is unknown. Hastie and Stuetzle (1989) also proposed an algorithm with which to derive the principal curve for given p-dimensional data of size n: $x_{ik}(i = 1, 2, \ldots, N; k = 1, 2, \ldots, p)$. In this case, the principal curves are represented by lines determined by N points $(\lambda_i, \boldsymbol{f}_i)$.

1. *Initialization*.
$$\boldsymbol{f}^{(0)}(\lambda) = \bar{\boldsymbol{x}} + \boldsymbol{u}\lambda \ ,$$
where $\boldsymbol{u}$ is the first principal component of the data and $\bar{\boldsymbol{x}}$ is the average of $\boldsymbol{x}$.
2. *Expectation Step*. Smooth x_{ik} $(i = 1, 2, \ldots, N)$ with respect to λ for each k independently and calculate $\boldsymbol{f}^{(j)}(\lambda)$.
3. *Projection Step*. Search for the nearest point on the curve (line curve) of each data point and assign it to their value of λ.
4. *Evaluation*. If a terminal condition is satisfied, the algorithm is stopped. If not, $j = j + 1$ and go to Step 2.

22.4 Linear Reduction of Explanatory Variables

Thus far, we have described dimension reduction methods for multidimensional data, where there are no distinctions among variables. However, there are times when we must analyze multidimensional data in which a variable is a response variable and others are explanatory variables. Regression analysis is usually used for the data. Dimension reduction methods of explanatory variables are introduced below.

Sliced Inverse Regression

Regression analysis is one of the fundamental methods used for data analysis. A response variable y is estimated by a function of explanatory variables $\boldsymbol{x}$, a p-dimensional vector. An immediate goal of ordinary regression analysis is to find the function of $\boldsymbol{x}$. When there are many explanatory variables in the data set, it is difficult to stably calculate the regression coefficients. An approach to reducing the number of explanatory variables is explanatory variable selection, and there are many studies on variable selection. Another approach is to project the explanatory variables on a lower dimensional space that nearly estimates the response variable.

Sliced Inverse Regression (SIR), which was proposed by Li (1991), is a method that can be employed to reduce explanatory variables with linear projection. SIR finds linear combinations of explanatory variables that are a reduction for non-linear regression. The original SIR algorithm, however, cannot derive suitable results for some artificial data with trivial structures. Li also developed another algorithm, SIR2, which uses the conditional estimation $E[\mathrm{cov}(\boldsymbol{x}|y)]$. However, SIR2 is also incapable of finding trivial structures for another type of data.

We hope that projection pursuit can be used for finding linear combinations of explanatory variables. A new SIR method with projection pursuit (SIRpp) is described here. We also present a numerical example of the proposed method.

Sliced Inverse Regression Model

SIR is based on the model (SIR model):

$$y = f\left(\boldsymbol{\beta}_1^\top \boldsymbol{x}, \boldsymbol{\beta}_2^\top \boldsymbol{x}, \ldots, \boldsymbol{\beta}_K^\top \boldsymbol{x}\right) + \varepsilon\,, \tag{22.7}$$

where $\boldsymbol{x}$ is the vector of p explanatory variables, $\boldsymbol{\beta}_k$ are unknown vectors, ε is independent of $\boldsymbol{x}$, and f is an arbitrary unknown function on $\boldsymbol{R}^K$.

The purpose of SIR is to estimate the vectors $\boldsymbol{\beta}_k$ for which this model holds. If we obtain $\boldsymbol{\beta}_k$, we can reduce the dimension of $\boldsymbol{x}$ to K. Hereafter, we shall refer to any linear combination of $\boldsymbol{\beta}_k$ as the effective dimensional reduction (e.d.r.) direction.

Li (1991) proposed an algorithm for finding e.d.r. directions, and it was named SIR. However, we refer to the algorithm as SIR1 to distinguish it from the SIR model.

The main idea of SIR1 is to use $E[\boldsymbol{x}|y]$. $E[\boldsymbol{x}|y]$ is contained in the space spanned by e.d.r. directions, but there is no guarantee that $E[\boldsymbol{x}|y]$ will span the space. For example, in Li, if $(X_1, X_2) \sim N(0, I_2), Y = X_1^2$ then $E[X_1|y] = E[X_2|y] = 0$.

SIR Model and Non-Normality

Hereafter, it is assumed that the distribution of $\boldsymbol{x}$ is standard normal distribution: $\boldsymbol{x} \sim N(0, I_p)$. If not, standardize $\boldsymbol{x}$ by affine transformation. In addition, $\boldsymbol{\beta}_i^\top \boldsymbol{\beta}_j = \delta_{ij}, (i, j = 1, 2, \ldots, K)$ is presumed without loss of generality. We can choose $\boldsymbol{\beta}_i (i = K+1, \ldots, p)$ such that $\{\boldsymbol{\beta}_i\}$ $(i = 1, \ldots, p)$ is a basis for $\boldsymbol{R}^p$.

Since the distribution of $\boldsymbol{x}$ is $N(0, I_p)$, the distribution of $(\boldsymbol{\beta}_1^\top \boldsymbol{x}, \ldots, \boldsymbol{\beta}_p^\top \boldsymbol{x})$ is also $N(0, I_p)$. The density function of $(\boldsymbol{\beta}_1^\top \boldsymbol{x}, \ldots, \boldsymbol{\beta}_p^\top \boldsymbol{x}, y)$ is

$$h\left(\boldsymbol{\beta}_1^\top \boldsymbol{x}, \ldots, \boldsymbol{\beta}_p^\top \boldsymbol{x}, y\right) = \phi\left(\boldsymbol{\beta}_1^\top \boldsymbol{x}\right) \ldots \phi\left(\boldsymbol{\beta}_p^\top \boldsymbol{x}\right) \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{\left(y - f\left(\boldsymbol{\beta}_1^\top \boldsymbol{x}, \ldots, \boldsymbol{\beta}_K^\top \boldsymbol{x}\right)\right)^2}{2\sigma^2}\right),$$

where $\phi(\boldsymbol{x}) = 1/\sqrt{2\pi} \exp(-x^2/2)$ and we assume $\varepsilon \sim N(0, \sigma^2)$.

The conditional density function is

$$h\left(\boldsymbol{\beta}_1^\top \boldsymbol{x}, \ldots, \boldsymbol{\beta}_p^\top \boldsymbol{x} \mid y\right) = \phi\left(\boldsymbol{\beta}_{K+1}^\top \boldsymbol{x}\right) \ldots \phi\left(\boldsymbol{\beta}_p^\top \boldsymbol{x}\right) g\left(\boldsymbol{\beta}_1^\top \boldsymbol{x}, \ldots, \boldsymbol{\beta}_K^\top \boldsymbol{x}\right),$$

where $g()$ is a function of $\boldsymbol{\beta}_1^\top \boldsymbol{x}, \ldots, \boldsymbol{\beta}_K^\top \boldsymbol{x}$, which is not generally the normal density function.

Thus, $h(\boldsymbol{\beta}_1^\top \boldsymbol{x}, \ldots, \boldsymbol{\beta}_p^\top \boldsymbol{x} \mid y)$ is separated into the normal distribution part $\phi(\boldsymbol{\beta}_{K+1}^\top \boldsymbol{x}) \ldots \phi(\boldsymbol{\beta}_p^\top \boldsymbol{x})$ and the non-normal distribution part $g()$.

Projection Pursuit is an excellent method for finding non-normal parts, so we adopt it for SIR.

SIRpp Algorithm

Here we show the algorithm for the SIR model with projection pursuit (SIRpp). The algorithm for the data $(y_i, \boldsymbol{x}_i)$ $(i = 1, 2, \ldots, n)$ is as follows:

1. Standardize $\boldsymbol{x}$: $\tilde{\boldsymbol{x}_i} = \hat{\Sigma}_{\boldsymbol{xx}}^{-1/2}(\boldsymbol{x}_i - \bar{\boldsymbol{x}})(i = 1, 2, \ldots, n)$, where $\hat{\Sigma}_{\boldsymbol{xx}}$ is the sample covariance matrix and $\bar{\boldsymbol{x}}$ is the sample mean of $\boldsymbol{x}$.
2. Divide the range of y into H slices, $I_1, \ldots, I_H$.

3. Conduct a projection pursuit in K dimensional space for each slice. The following H projections are obtained: $(\boldsymbol{\alpha}_1^{(h)}, \ldots, \boldsymbol{\alpha}_K^{(h)})$, $(h = 1, \ldots, H)$.
4. Let the K largest eigenvectors of $\hat{V}$ be $\hat{\boldsymbol{\eta}}_k (k = 1, \ldots, K)$. Output $\hat{\boldsymbol{\beta}}_k = \hat{\boldsymbol{\eta}}_k \boldsymbol{\Sigma}_{\boldsymbol{xx}}^{-1/2} (k = 1, 2, \ldots, K)$ for the estimation of e.d.r. directions, where $\hat{V} = \sum_{h=1}^{H} w(h) \sum_{k=1}^{K} \boldsymbol{\alpha}_k^{(h)\top} \boldsymbol{\alpha}_k^{(h)}$.

Numerical Examples

Two models of the multicomponent are used:

$$y = x_1(x_1 + x_2 + 1) + \sigma \cdot \varepsilon \,, \tag{22.8}$$

$$y = \sin(x_1) + \cos(x_2) + \sigma \cdot \varepsilon \tag{22.9}$$

to generate $n = 400$ data, where $\sigma = 0.5$ (Fig. 22.6, Fig. 22.7). We first generate x_1, x_2, ε with $N(0, 1)$ and calculate response variable y using (22.8) or (22.9). Eight variables $x_3, \ldots, x_{10}$ generated by $N(0, 1)$ are added to the explanatory variables. The ideal e.d.r. directions are contained within the space spanned by two vectors $(1, 0, \ldots, 0)$ and $(0, 1, \ldots, 0)$.

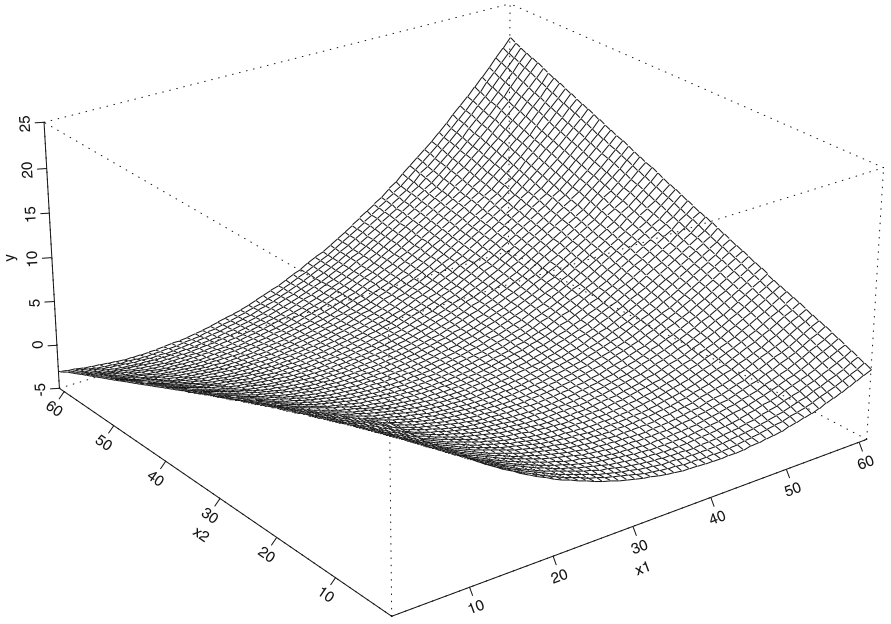


Fig. 22.6 Function of the example 1. Asymmetric function $y = x_1(x_1 + x_2 + 1)$

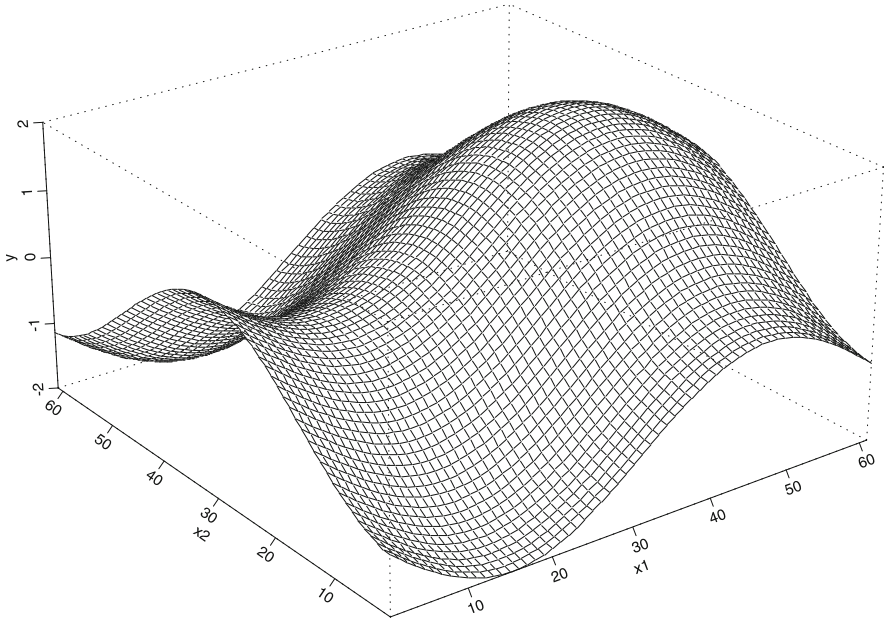


Fig. 22.7 Function of the example 2. Function of asymmetric with respect to the x_1 axis and symmetric with respect to x_2 axis. $y = \sin(x_1) + \cos(x_2)$

The squared multiple correlation coefficient between the projected variable $\boldsymbol{b}^\top \boldsymbol{x}$ and the space B spanned by ideal e.d.r. directions:

$$R^2(\boldsymbol{b}) = \max_{\boldsymbol{\beta} \in B} \frac{\left(\boldsymbol{b}^\top \sum_{xx} \boldsymbol{\beta}\right)^2}{\boldsymbol{b}^\top \sum_{xx} \boldsymbol{b} \cdot \boldsymbol{\beta}^\top \sum_{xx} \boldsymbol{\beta}} \tag{22.10}$$

is adopted as the criterion for evaluating the effectiveness of estimated e.d.r. directions.

Table 22.1 and Table 22.2 show the mean and the standard deviation (in parentheses) of $R^2(\hat{\boldsymbol{\beta}}_1)$ and $R^2(\hat{\boldsymbol{\beta}}_2)$ of four SIR algorithms for $H = 5, 10$, and 20, after 100 replicates. SIR2 cannot reduce the explanatory variables from the first example. The result of the second example is very interesting. SIR1 finds the asymmetric e.d.r. direction, but, does not find the symmetric e.d.r. direction. Conversely, SIR2 finds only the symmetric e.d.r. direction. SIRpp can detect both of the e.d.r. directions.

The SIRpp algorithm performs well in finding the e.d.r. directions; however, the algorithm requires more computing power. This is one part of projection pursuit for which the algorithm is time consuming.

Table 22.1 Results for SIR1, SIR2, and SIRpp (Example 1)

	SIR1		SIR2		SIRpp	
H	$R^2\left(\hat{\beta}_1\right)$	$R^2\left(\hat{\beta}_2\right)$	$R^2\left(\hat{\beta}_1\right)$	$R^2\left(\hat{\beta}_2\right)$	$R^2\left(\hat{\beta}_1\right)$	$R^2\left(\hat{\beta}_2\right)$
5	0.92	0.77	0.96	0.20	0.97	0.78
	(0.04)	(0.11)	(0.03)	(0.21)	(0.02)	(0.15)
10	0.93	0.81	0.92	0.10	0.95	0.79
	(0.03)	(0.09)	(0.09)	(0.12)	(0.04)	(0.13)
20	0.92	0.76	0.83	0.11	0.95	0.75
	(0.04)	(0.18)	(0.19)	(0.13)	(0.07)	(0.18)

Table 22.2 Results of SIR1, SIR2, and SIRpp (Example 2)

	SIR1		SIR2		SIRpp	
H	$R^2\left(\hat{\beta}_1\right)$	$R^2\left(\hat{\beta}_2\right)$	$R^2\left(\hat{\beta}_1\right)$	$R^2\left(\hat{\beta}_2\right)$	$R^2\left(\hat{\beta}_1\right)$	$R^2\left(\hat{\beta}_2\right)$
5	0.97	0.12	0.92	0.01	0.92	0.88
	(0.02)	(0.14)	(0.04)	(0.10)	(0.05)	(0.11)
10	0.97	0.12	0.90	0.05	0.88	0.84
	(0.02)	(0.15)	(0.06)	(0.07)	(0.08)	(0.13)
20	0.97	0.12	0.85	0.05	0.84	0.73
	(0.02)	(0.14)	(0.09)	(0.06)	(0.10)	(0.22)

22.5 Concluding Remarks

In this chapter, we discussed dimension reduction methods for data analysis. First, PCA methods were explained for the linear method. Then, projection pursuit methods were described. For nonlinear methods, GPCA algebraic curve fitting methods and principal curves were introduced. Finally, we explained sliced inverse regression for the reduction of the dimension of explanatory variable space.

These methods are not only useful for data analysis, but also effective for preprocessing when carrying out another data analysis. In particular, they are indispensable for the analysis of enormous amounts of and complex data, e.g. microarray data, log data on the Internet, etc. Research in this field will continue to evolve in the future.

References

Friedman, J.: Exploratory projection pursuit. J. Am. Stat. Assoc. **82**, 249–266 (1987)

Friedman, J., Tukey, J.: A projection pursuit algorithm for exploratory data analysis. IEEE Trans. Comput. **c-23**(9), 881–890 (1974)

Gnanadesikan, R., Wilk, M.: Data analytic methods. In: Krishnaiah, P. (eds.) Multivariate Analysis II, pp. 593–638. Academic Press, New York (Orlando.FL/London) (1969)

Hall, P.: On polynomial-based projection indices for exploratory projection pursuit. Ann. Stat. **17**, 589–605 (1989)
Hastie, T., Stuetzle, W.: Principal curves. J. Am. Stat. Assoc. **84**, 502–516 (1989)
Huber, P.: Projection pursuit (with discussion) Ann. Stat. **13**, 435–475 (1985)
Iwasaki, M.: Projection pursuit: the idea and practice (in japanese). Bull. Comput. Stat. Jpn. **4**(2), 41–56 (1991)
Jones, M.C., Sibson, R.: What is projection pursuit? (with discussion). J. Roy. Stat. Soc. A **150**, 1–36 (1987)
Keren, D., Cooper, D., Subrahmonia, J.: Describing complicated objects by implicit polynomials. IEEE Trans. Patt. Anal. Mach. Intell. **16**(1), 38–53 (1994)
Koyama, K., Morita, A., Mizuta, M., Sato, Y.: Projection pursuit into three dimensional space (in japanese). Jpns. J. Behaviormetrics **25**(1), 1–9 (1998)
Li, K.: Sliced inverse regression for dimension reduction. J. Am. Stat. Assoc. 86:316–342 (1991)
Mizuta, M.: Generalized principal components analysis invariant under rotations of a coordinate system. J. Jpn. Stat. Soc. **14**, 1–9 (1983)
Mizuta, M.: A derivation of the algebraic curve for two-dimensional data using the least-squares distance. In: Escoufier, Y., Hayashi, C., Fichet, B., Ohsumi, N., Diday, E., Baba, Y., Lebart, L. (eds.) Data Science and Its Application, pp. 167–176. Academic Press, Tokyo (1995)
Mizuta, M.: Algebraic curve fitting for multidimensional data with exact squares distance. In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, pp. 516–521 (1996)
Mizuta, M.: Rasterizing algebraic curves and surfaces in the space with exact distances. In: Progress in Connectionist-Based Information Systems – Proceedings of the 1997 International Conference on Neural Information Processing and Intelligent Information Systems, pp. 551–554 (1997)
Mizuta, M.: Relative projection pursuit. In: Sokolowski, A., Jajuga, K., (eds.) Data Analysis, Classification, and Related Methods, p. 131. Cracow University of Economics, Poland (2002)
Nason, G.: Three-dimensional projection pursuit (with discussion). Appl. Stat. **44**(4), 411–430 (1995)
Taubin, G.: Estimation of planar curves, surfaces, and nonplanar space curves defined by implicit equations with applications to edge and range image segmentation. IEEE Trans. Patt. Anal. Mach. Intell. **13**(11), 1115–1138 (1991)
Taubin, G.: Distance approximations for rasterizing implicit curves. ACM Trans. Graph. **13**, 3–42 (1994)
Taubin, G., Cukierman, F., Sullivan, S., Ponce, J., Kriegman, D.: Parameterized families of polynomials for bounded algebraic curve and surface fitting. IEEE Trans. Patt. Anal. Mach. Intell. **16**(3), 287–303 (1994)

Chapter 23
(Non) Linear Regression Modeling

Pavel Čížek

We will study causal relationships of a known form between random variables. Given a model, we distinguish one or more dependent (endogenous) variables $\boldsymbol{Y} = (Y_1, \ldots, Y_l), l \in \mathbb{N}$, which are explained by a model, and independent (exogenous, explanatory) variables $\boldsymbol{X} = (X_1, \ldots, X_p), p \in \mathbb{N}$, which explain or predict the dependent variables by means of the model. Such relationships and models are commonly referred to as regression models.

A regression model describes the relationship between the dependent and independent variables. In this chapter, we restrict our attention to models with a form known up to a finite number of unspecified parameters. The model can be either linear in parameters,

$$\boldsymbol{Y} = \boldsymbol{X}^\top \boldsymbol{\beta_0} + \varepsilon,$$

or nonlinear,

$$\boldsymbol{Y} = h(\boldsymbol{X}, \boldsymbol{\beta_0}) + \varepsilon,$$

where $\boldsymbol{\beta}$ represents a vector or a matrix of unknown parameters, ε is the error term (fluctuations caused by unobservable quantities), and h is a known regression function. The unknown parameters $\boldsymbol{\beta}$ are to be estimated from observed realizations $\{y_{1i}, \ldots, y_{li}\}_{i=1}^n$ and $\{x_{1i}, \ldots, x_{pi}\}_{i=1}^n$ of random variables $\boldsymbol{Y}$ and $\boldsymbol{X}$.

Here we discuss both kinds of models, primarily from the least-squares estimation point of view, in Sects. 23.1 and 23.2, respectively. Both sections present the main facts concerning the fitting of these models and relevant inference, whereby their focus is above all on the estimation of these regression models under near and exact multicollinearity and closely related model selection.

P. Čížek (✉)
Department of Econometrics & Operations Research, Tilburg University, Tilburg, The Netherlands
e-mail: P.Cizek@uvt.nl

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_23,

23.1 Linear Regression Modeling

Let us first study the linear regression model $Y = \boldsymbol{X}^\top \boldsymbol{\beta_0} + \varepsilon$ assuming $E(\varepsilon|\boldsymbol{X}) = 0$. Unless said otherwise, we consider here only one dependent variable Y. The unknown vector $\boldsymbol{\beta^0} = (\beta_1^0, \ldots, \beta_p^0)$ is to be estimated given observations $\boldsymbol{y} = (y_1, \ldots, y_n) \in \mathbb{R}^n$ and $\{\boldsymbol{x_i}\}_{i=1}^n = \{(x_{1i}, \ldots, x_{pi})\}_{i=1}^n$ of random variables Y and X; let us denote $\boldsymbol{X} = (\boldsymbol{x_1}, \ldots, \boldsymbol{x_n})^\top \in \mathbb{R}^{n \times p}$ and let $\boldsymbol{x_{\cdot k}}$ be the kth column of $\boldsymbol{X}$. Thus, the linear regression model can be written in terms of observations as

$$\boldsymbol{y} = \boldsymbol{X\beta_0} + \boldsymbol{\varepsilon} = \sum_{k=1}^{p} \boldsymbol{x_{\cdot k}} \beta_k^0 + \boldsymbol{\varepsilon}, \tag{23.1}$$

where $\boldsymbol{\varepsilon} = (\varepsilon_1, \ldots, \varepsilon_n) \in \mathbb{R}^n$.

Section 23.1.1 summarizes how to estimate the model (23.1) by the method of least squares. Later, we specify what ill-conditioning and multicollinearity are in Sect. 23.1.2 and discuss methods dealing with it in Sects. 23.1.3–23.1.10.

23.1.1 Fitting of Linear Regression

Let us first review the least squares estimation and its main properties to facilitate easier understanding of the fitting procedures discussed further. For a detailed overview of linear regression modeling see Rao and Toutenberg (1999).

The *least squares* (LS) approach to the estimation of (23.1) searches an estimate $\hat{\boldsymbol{\beta}}$ of unknown parameters $\boldsymbol{\beta_0}$ by minimizing the sum of squared differences between the observed values y_i and the predicted ones $\hat{y}_i(\hat{\boldsymbol{\beta}}) = \boldsymbol{x_i}^\top \hat{\boldsymbol{\beta}}$.

Definition 1. The least squares estimate of linear regression model (23.1) is defined by

$$\hat{\boldsymbol{\beta}}^{LS} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{i=1}^{n} \{y_i - \hat{y}_i(\boldsymbol{\beta})\}^2 = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{i=1}^{n} (y_i - \boldsymbol{x_i}^\top \boldsymbol{\beta})^2. \tag{23.2}$$

This differentiable problem can be expressed as the minimization of

$$(\boldsymbol{y} - \boldsymbol{X\beta})^\top (\boldsymbol{y} - \boldsymbol{X\beta}) = \boldsymbol{y}^\top \boldsymbol{y} - 2\boldsymbol{\beta}^\top \boldsymbol{X}^\top \boldsymbol{y} + \boldsymbol{\beta}^\top \boldsymbol{X}^\top \boldsymbol{X\beta}$$

with respect to $\boldsymbol{\beta}$ and the corresponding first-order conditions are

$$-\boldsymbol{X}^\top \boldsymbol{y} + \boldsymbol{X}^\top \boldsymbol{X\beta} = 0 \quad \Longrightarrow \quad \boldsymbol{X}^\top \boldsymbol{X\beta} = \boldsymbol{X}^\top \boldsymbol{y}. \tag{23.3}$$

They are commonly referred to as the normal equations and identify the global minimum of (23.2) as long as the second order conditions $\boldsymbol{X}^\top \boldsymbol{X} > 0$ hold; that is, the matrix $\boldsymbol{X}^\top \boldsymbol{X}$ is supposed to be positive definite, or equivalently, non-singular.

(This mirrors an often used assumption specified in terms of the underlying random vector X if regressors are stochastic: $E(\boldsymbol{X}\boldsymbol{X}^\top) > 0$.) Provided that $\boldsymbol{X}^\top\boldsymbol{X} > 0$ and $E(\boldsymbol{\varepsilon}|\boldsymbol{X}) = 0$, the LS estimator is unbiased and can be found as a solution of (23.3)

$$\hat{\boldsymbol{\beta}}^{LS} = (\boldsymbol{X}^\top\boldsymbol{X})^{-1}\boldsymbol{X}^\top\boldsymbol{y}. \tag{23.4}$$

Additionally, it is the best linear unbiased estimator of (23.1), see Amemiya (1985).

Theorem 1. *(Gauss-Markov) Assume that* $E(\boldsymbol{\varepsilon}|\boldsymbol{X}) = 0$, $E(\boldsymbol{\varepsilon}^2|\boldsymbol{X}) = \sigma^2\boldsymbol{I_n}$, *and* $\boldsymbol{X}^\top\boldsymbol{X}$ *is non-singular. Let* $\hat{\boldsymbol{\beta}} = \boldsymbol{C}^\top\boldsymbol{y}$, *where* $\boldsymbol{C}$ *is a* $t \times p$ *matrix orthogonal to* $\boldsymbol{X}$, $\boldsymbol{C}^\top\boldsymbol{X} = \boldsymbol{I}$. *Then* $Var(\hat{\boldsymbol{\beta}}) - Var(\hat{\boldsymbol{\beta}}^{LS}) > 0$ *is a positive definite matrix for any* $\hat{\boldsymbol{\beta}} \neq \hat{\boldsymbol{\beta}}^{LS}$.

Finally, the LS estimate actually coincides with the maximum likelihood estimate provided that random errors ε are normally distributed (in addition to the assumptions of Theorem 1) and shares then the asymptotic properties of the maximum likelihood estimation (see Amemiya 1985).

Computing LS Estimates

The LS estimate $\hat{\boldsymbol{\beta}}^{LS}$ can be and often is found by directly solving the system of linear equations (23.3) or evaluating formula (23.4), which involves a matrix inversion. Both direct and iterative methods for solving systems of linear equations are presented in Chap. II.4. Although this straightforward computation may work well for many regression problems, it often leads to an unnecessary loss of precision, see Miller (2002). Additionally, it is not very suitable if the matrix $\boldsymbol{X}^\top\boldsymbol{X}$ is ill-conditioned (a regression problem is called ill-conditioned if a small change in data causes large changes in estimates) or nearly singular (multicollinearity) because it is not numerically stable. Being concerned mainly about statistical consequences of multicollinearity, the numerical issues regarding the identification and treatment of ill-conditioned regression models are beyond the scope of this contribution. Let us refer an interested reader to Barlow (1993), Bjorck (1996), Miller (2002), Thisted (1988), and Wang et al. (1990).

Let us now briefly review a class of numerically more stable algorithms for the LS minimization. They are based on orthogonal transformations. Assuming a matrix $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ is an orthonormal matrix, $\boldsymbol{Q}^\top\boldsymbol{Q} = \boldsymbol{Q}\boldsymbol{Q}^\top = \boldsymbol{I_n}$,

$$(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^\top(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}) = (\boldsymbol{Q}\boldsymbol{y} - \boldsymbol{Q}\boldsymbol{X}\boldsymbol{\beta})^\top(\boldsymbol{Q}\boldsymbol{y} - \boldsymbol{Q}\boldsymbol{X}\boldsymbol{\beta}).$$

Thus, multiplying a regression model by an orthonormal matrix does not change it from the LS point of view. Since every matrix $\boldsymbol{X}$ can be decomposed into the product $\boldsymbol{Q}_x\boldsymbol{R}_x$ (the QR decomposition), where $\boldsymbol{Q}_x$ is an orthonormal matrix and $\boldsymbol{R}_x$ is an upper triangular matrix, pre-multiplying (23.1) by $\boldsymbol{Q}_x^\top$ produces

$$\boldsymbol{Q}_x^\top\boldsymbol{y} = \boldsymbol{R_x}\boldsymbol{\beta} + \boldsymbol{Q}_x^\top\boldsymbol{\varepsilon}, \tag{23.5}$$

where $\boldsymbol{R_x} = (\boldsymbol{R_1}, \boldsymbol{R_2})^\top$, $\boldsymbol{R_1} \in \mathbb{R}^{p \times p}$ is an upper triangular matrix, and $\boldsymbol{R_2} \in \mathbb{R}^{(n-p) \times p}$ is a zero matrix. Hence, the sum of squares to minimize can be written as

$$(\boldsymbol{Q}_x^\top \boldsymbol{y} - \boldsymbol{R_x}\boldsymbol{\beta})^\top (\boldsymbol{Q}_x^\top \boldsymbol{y} - \boldsymbol{R_x}\boldsymbol{\beta}) = (\boldsymbol{y_1} - \boldsymbol{R_1}\boldsymbol{\beta})^\top (\boldsymbol{y_1} - \boldsymbol{R_1}\boldsymbol{\beta}) + \boldsymbol{y_2}^\top \boldsymbol{y_2},$$

where $\boldsymbol{y_1} \in \mathbb{R}^p$ and $\boldsymbol{y_2} \in \mathbb{R}^{n-p}$ form $\boldsymbol{Q}_x^\top \boldsymbol{y} = (\boldsymbol{y_1}^\top, \boldsymbol{y_2}^\top)^\top$. The LS estimate is then obtained from the upper triangular system $\boldsymbol{R_1}\boldsymbol{\beta} = \boldsymbol{y_1}$, which is trivial to solve by backward substitution. There are many algorithms for constructing a suitable QR decomposition for finding LS estimates, such as the Householder or Givens transformations; see Chap. II.4, Bjorck (1996), and Gentle (1998) for more details.

LS Inference

Linear regression modeling does not naturally consist only of obtaining a point estimate $\hat{\boldsymbol{\beta}}^{\boldsymbol{LS}}$. One needs to measure the variance of the estimates in order to construct confidence intervals or test hypotheses. Additionally, one should assess the quality of the regression fit. Most such measures are based on regression residuals $\boldsymbol{e} = \boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}$. We briefly review the most important regression statistics, and next, indicate how it is possible to compute them if the LS regression is estimated by means of some orthogonalization procedure described in the previous paragraph.

The most important measures used in statistics to assess model fit and inference are the total sum of squares $TSS = (\boldsymbol{y} - \bar{y})^\top (\boldsymbol{y} - \bar{\boldsymbol{y}}) = \sum_{i=1}^n (y_i - \bar{y})^2$, where $\bar{y} = \sum_{i=1}^n y_i / n$, the residual sum of squares $RSS = \boldsymbol{e}^\top \boldsymbol{e} = \sum_{i=1}^n e_i^2$, and the complementary regression sum of squares $RegSS = (\boldsymbol{y} - \hat{\boldsymbol{y}})^\top (\boldsymbol{y} - \hat{\boldsymbol{y}}) = \sum_{i=1}^n (y_i - \hat{y}_i)^2 = TSS - RSS$. Using these quantities, the regression fit can be evaluated; for example, the coefficient of determination $R^2 = 1 - RSS/TSS$ as well as many information criteria (modified $\bar{R}^2$, Mallows and Akaike criteria, etc.; see Sect. 23.1.3). Additionally, they can be used to compute the variance of the estimates in simple cases. The variance of the LS estimates can be estimated by

$$Var(\hat{\boldsymbol{\beta}}^{\boldsymbol{LS}}) = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{S}^{-1} \boldsymbol{X} (\boldsymbol{X}^\top \boldsymbol{X})^{-1}, \tag{23.6}$$

where $\boldsymbol{S}$ represents an estimate of the covariance matrix $Var(\boldsymbol{\varepsilon}) = \boldsymbol{\Sigma}$. Provided that the model is homoscedastic, $\boldsymbol{\Sigma} = \sigma^2 \boldsymbol{I_n}$, the residual variance σ^2 can be estimated as an average of squared residuals $s^2 = \boldsymbol{e}^\top \boldsymbol{e} / n$. Apart from the residual variance, one needs also an inverse of $(\boldsymbol{X}^\top \boldsymbol{X})^{-1}$, which will often be a by-product of solving normal equations.

Let us now describe how one computes these quantities if a numerically stable procedure based on the orthonormalization of normal equations is used. Let us assume we already constructed a QR decomposition of $\boldsymbol{X} = \boldsymbol{Q_x}\boldsymbol{R_x}$. Thus, $\boldsymbol{Q_x}\boldsymbol{Q_x}^\top = \boldsymbol{I}$ and $\boldsymbol{Q_x}^\top \boldsymbol{X} = \boldsymbol{R_x}$. RSS can be computed as

$$RSS = \boldsymbol{e}^\top \boldsymbol{e} = (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})^\top (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}) = (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})^\top \boldsymbol{Q}_x \boldsymbol{Q}_x^\top (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})$$
$$= (\boldsymbol{Q}_x^\top \boldsymbol{y} - \boldsymbol{R}_x \boldsymbol{X}\hat{\boldsymbol{\beta}})^\top (\boldsymbol{Q}_x^\top \boldsymbol{y} - \boldsymbol{R}_x \boldsymbol{X}\hat{\boldsymbol{\beta}}).$$

Consequently, RSS is invariant with respect to orthonormal transformations (23.5) of the regression model (23.1). The same conclusion applies also to TSS and $RegSS$, and consequently, to the variance estimation. Thus, it is possible to use the data in (23.5), transformed to achieve better numerical stability, for computing regression statistics of the original model (23.1).

23.1.2 Multicollinearity

Let us assume that the design matrix $\boldsymbol{X}$ fixed. We talk about *multicollinearity* when there is a linear dependence among the variables in regression, that is, the columns of $\boldsymbol{X}$.

Definition 2. In model (23.1), the exact multicollinearity exists if there are real constants $a_1, \ldots, a_p$ such that $\sum_{k=1}^{p} |a_k| > 0$ and $\sum_{k=1}^{p} a_k \boldsymbol{x}_{\boldsymbol{\cdot k}} = 0$.

The exact multicollinearity (also referred to as reduced-rank data) is relatively rare in linear regression models unless the number of explanatory variables is very large or even larger than the number of observations, $p \geq n$. This happens often in agriculture, chemometrics, sociology, and so on. For example, Miller (2002) uses data on the absorbances of infra-red rays at many different wavelength by chopped meat, whereby the aim is to determine the moisture, fat, and protein content of the meat as a function of these absorbances. The study employs measurements at 100 wavelengths from 850 nm to 1,050 nm, which gives rise to many possibly correlated variables.

When the number p of variables is small compared to the sample size n, near multicollinearity is more likely to occur: there are some real constants $a_1, \ldots, a_p$ such that $\sum_{k=1}^{p} |a_k| > 0$ and $\sum_{k=1}^{p} a_k \boldsymbol{x}_{\boldsymbol{\cdot k}} \approx 0$, where $\approx$ denotes approximate equality. The multicollinearity in data does not have to arise only as a result of highly correlated variables (i.e., be systematic because, for example, more measurements of the same characteristic are taken by different sensors or methods and are used as separate variables), but it could also result from the lack of information and variability in data (i.e., be erratic and purely numerical in its nature). Different causes of multicollinearity and their consequences are extensively discussed by Spanos and McGuirk (2002).

Whereas the exact multicollinearity implies that $\boldsymbol{X}^\top \boldsymbol{X}$ is singular and the LS estimator is not identified, the near multicollinearity permits a non-singular matrix $\boldsymbol{X}^\top \boldsymbol{X}$. The eigenvalues $\lambda_1 \leq \ldots \leq \lambda_p$ of matrix $\boldsymbol{X}^\top \boldsymbol{X}$ can give some indication concerning multicollinearity: if the smallest eigenvalue λ_1 equals zero, the matrix is singular and data are exactly multicollinear; if λ_1 is close to zero, near multicollinearity is present in data. Since measures based on eigenvalues depend on

the parametrization of the model, they are not necessarily optimal and it is often easier to detect multicollinearity by looking at LS estimates and their behavior as discussed in the next paragraph. See Bjorck (1996) and Leamer (1983) for more details on detection and treatment of ill-conditioned problems. (Nearly singular matrices are dealt with also in numerical mathematics. To measure near singularity, numerical mathematics uses conditioning numbers $d_k = \sqrt{\lambda_k / \lambda_1}$, which converge to infinity for singular matrices, that is, as $\lambda_1 \to 0$. Matrices with very large conditioning numbers are called ill-conditioned.)

The multicollinearity has important implications for LS. In the case of the exact multicollinearity, matrix $\boldsymbol{X}^\top \boldsymbol{X}$ does not have a full rank. Hence, the solution of the normal equations is not unique, the LS estimate $\hat{\boldsymbol{\beta}}^{\boldsymbol{LS}}$ is not identified, and one has to introduce additional restrictions to identify the LS estimate. On the other hand, even though near multicollinearity does not prevent the identification of LS, it negatively influences estimation results. Since both the estimate $\hat{\boldsymbol{\beta}}^{\boldsymbol{LS}}$ and its variance are proportional to the inverse of $\boldsymbol{X}^\top \boldsymbol{X}$, which is nearly singular under multicollinearity, near multicollinearity inflates $\hat{\boldsymbol{\beta}}^{\boldsymbol{LS}}$, which may become unrealistically large, and variance $Var(\hat{\boldsymbol{\beta}}^{\boldsymbol{LS}})$. Consequently, the corresponding t-statistics are typically very low. Moreover, due to the large values of $(\boldsymbol{X}^\top \boldsymbol{X})^{-1}$, the least squares estimate $\hat{\boldsymbol{\beta}}^{\boldsymbol{LS}} = (\boldsymbol{X}^\top \boldsymbol{X})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$ reacts very sensitively to small changes in data. See Hocking (1996) and Montgomery et al. (2001) for a more detailed treatment and real-data examples of the effects of multicollinearity.

There are several strategies to limit the adverse consequences of multicollinearity provided that one cannot improve the design of a model or experiment or get better data. First, one can impose an additional structure on the model. This strategy cannot be discussed in details since it is model specific, and in principle, it requires only to test a hypothesis concerning additional restrictions. Second, it is possible to reduce the dimension of the space spanned by $\boldsymbol{X}$, for example, by excluding some variables from the regression (Sects. 23.1.3 and 23.1.4). Third, one can also leave the class of unbiased estimators and try to find a biased estimator with a smaller variance and mean squared error. Assuming we want to judge the performance of an estimator $\hat{\boldsymbol{\beta}}$ by its mean squared error (MSE), the motivation follows from

$$\begin{aligned} MSE(\hat{\boldsymbol{\beta}}) &= E[(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta^0})(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta^0})^\top] \\ &= E[\{\hat{\boldsymbol{\beta}} - E(\hat{\boldsymbol{\beta}})\}\{\hat{\boldsymbol{\beta}} - E(\hat{\boldsymbol{\beta}})\}^\top] + [E\{E(\hat{\boldsymbol{\beta}}) - \boldsymbol{\beta^0}\}][E\{E(\hat{\boldsymbol{\beta}}) - \boldsymbol{\beta^0}\}]^\top \\ &= Var(\hat{\boldsymbol{\beta}}) + Bias(\hat{\boldsymbol{\beta}}) Bias(\hat{\boldsymbol{\beta}})^\top. \end{aligned}$$

Thus, it is possible that introducing a bias into estimation in such a way that the variance of estimates is significantly reduced can improve the estimator's MSE. There are many biased alternatives to the LS estimation as discussed in Sects. 23.1.5–23.1.10 and some of them even combine biased estimation with variable selection. In all cases, we present methods usable both in the case of near and exact multicollinearity.

23.1.3 Variable Selection

The presence of multicollinearity may indicate that some explanatory variables are linear combinations of the other ones (note that this is more often a "feature" of data rather than of the model). Consequently, they do not improve explanatory power of a model and could be dropped from the model provided there is some justification for dropping them also on the model level rather than just dropping them to fix data problems. As a result of removing some variables, matrix $\boldsymbol{X}^\top \boldsymbol{X}$ would not be (nearly) singular anymore.

Eliminating variables from a model is a special case of model selection procedures, which are discussed in details in Chap. III.1. Here we first discuss methods specific for the variable selection within a single regression model, mainly variants of stepwise regression. Later, we deal with more general model selection methods, such as cross validation, that are useful both in the context of variable selection and of biased estimation discussed in Sects. 23.1.5–23.1.10. An overview and comparison of many classical variable selection is given, for example, in Miller (1984, 2002) and Montgomery et al. (2001). For discussion of computational issues related to model selection, see Kennedy and Gentle (1980) and Miller (2002). Finally, note that proper inference after variable selection is generally difficult as even resampling methods (see Chap. III.2) are not applicable and outside of the scope of this chapter. A general method for approximately unbiased inference after variable selection was proposed by Shen et al. (2004).

Backward Elimination

A simple and often used method to eliminate non-significant variables from regression is *backward elimination*, a special case of stepwise regression. Backward elimination starts from the full model $\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{\varepsilon}$ and identifies a variable $\boldsymbol{x}_{\cdot k}$ such that:

1. Its omission results in the smallest increase of RSS; or
2. It has the smallest t-statistics $t_k = b_k^{LS} / \sqrt{s_k^2/(n-p)}$, where s_k^2 is an estimate of b_k^{LS} variance, or any other test statistics of $H_0 : \beta_{0k} = 0$; or
3. Its removal causes the smallest change of a prediction or information criterion characterizing the fit or prediction power of the model. Well-known examples of information criteria are the modified coefficient of determination $\bar{R}^2 = 1 - (n+p)\boldsymbol{e}^\top \boldsymbol{e}/n(n-p)$, Akaike information criterion (Akaike 1974), $AIC = \log(\boldsymbol{e}^\top \boldsymbol{e}/n) + 2p/n$, and Schwarz information criterion (Schwarz 1978), $SIC = \log(\boldsymbol{e}^\top \boldsymbol{e}/n) + p \ln n/n$, where n and p represents sample size and the number of regressors, respectively.

Next, the variable $\boldsymbol{x}_{\cdot k}$ is excluded from regression by setting $b_k = 0$ if (1) one did not reach a pre-specified number of variables yet or (2) the test statistics or change of the information criterion lies below some selected significance level.

Before discussing properties of backward elimination, let us make several notes on information criteria used for the elimination and their optimality. There is a wide range of selection criteria, including classical AIC, SIC, FPE_λ by Shibata (1984), cross validation by Stone (1974), and so on. Despite one can consider the same measure of the optimality of variable selection, such as the sum of squared prediction errors (Shibata 1981), one can often see contradictory results concerning the selection criteria (cf. Li 1997, and Shao 1993; or Shibata (1981); and Rao and Wu 1989). This is caused by different underlying assumptions about the true model (Shao 1997). Some criteria, such as AIC and cross validation, are optimal if one assumes that there is no finite-dimensional true model (i.e., the number of variables increases with the sample size); see Shibata (1981) and Li (1987). On the other hand, some criteria, such as SIC, are consistent if one assumes that there is a true model with a finite number of variables; see Rao and Wu (1989) and Shao (1997). A combination of two concepts can be achieved by using an information criterion that penalizes the number of selected variables in a data-dependent way (Shen and Ye 2002). Finally, note that even though some criteria, being optimal in the same sense, are asymptotically equivalent, their finite sample properties can differ substantially. See Chap. III.1 for more details.

Let us now return back to backward elimination, which can be also viewed as a pre-test estimator (Judge and Bock 1983). Although it is often used in practice, it involves a largely arbitrary choice of the significance level. In addition, it has rather poor statistical properties caused primarily by discontinuity of the selection decision, see Magnus (1999). Attempts to improve the properties of backward selection by bootstrap do not seem to be successful either (Austin 2008). Moreover, even if a stepwise procedure is employed, one should take care of reporting correct variances and confidence intervals valid for the whole decision sequence. Inference for the finally selected model as if it were the only model considered leads to significant biases, see Danilov and Magnus (2004), Weiss (1995), and Zhang (1992). Backward elimination also does not perform well in the presence of multicollinearity and it cannot be used if $p > n$. Finally, let us note that a nearly optimal and admissible alternative is proposed in Magnus (2002).

Forward Selection

Backward elimination cannot be applied if there are more variables than observations, and additionally, it may be very computationally expensive if there are many variables. A classical alternative is *forward selection*, where one starts from an intercept-only model and adds one after another variables that provide the largest decrease of RSS. Adding stops when the F-statistics

$$R = \frac{RSS_p - RSS_{p+1}}{RSS_{p+1}}(n - p - 2)$$

lies below a pre-specified critical "F-to-enter" value. The forward selection can be combined with the backward selection (e.g., after adding a variable, one performs one step of backward elimination), which is known as a stepwise regression Efroymson (1960). Its computational complexity is discussed in Miller (2002).

Note that most disadvantages of backward elimination apply to forward selection as well. In particular, correct variances and confidence intervals should be reported, see Miller (2002) for their approximations. Moreover, forward selection can be overly aggressive in selection in the respect that if a variable $\boldsymbol{x}$ is already included in a model, forward selection primarily adds variables orthogonal to $\boldsymbol{x}$, thus ignoring possibly useful variables that are correlated with $\boldsymbol{x}$. To improve upon this, Efron et al. (2004) proposed the least angle regression, considering correlations of to-be-added variables jointly with respect to all variables already included in the model (see Sect. 23.1.9).

All-Subsets Regression

Neither forward selection, nor backward elimination guarantee the optimality of the selected submodel, even when both methods lead to the same results. This can happen especially when a pair of variables has jointly a high predictive power; for example, if the dependent variable $\boldsymbol{y}$ depends on the difference of two variables $\boldsymbol{x_1} - \boldsymbol{x_2}$. An alternative approach, which is aiming at optimality of the selected subset of variables – *all-subsets regression* – is based on forming a model for each subset of explanatory variables. Each model is estimated and a selected prediction or information criterion, which quantifies the unexplained variation of the dependent variable and the parsimony of the model, is evaluated. Finally, the model attaining the best value of a criterion is selected and variables missing in this model are omitted.

This approach deserves several comments. First, one can use many other criteria instead of AIC or SIC. These could be based on the test statistics of a joint hypothesis that a group of variables has zero coefficients, extensions or modifications of AIC or SIC, general Bayesian predictive criteria, criteria using non-sample information, model selection based on estimated parameter values at each subsample and so on. See the next subsection, Bedrick and Tsai (1994), Hughes and Maxwell (2003), Jian and Liu (2004), Ibrahim and Ming-Hui (1997), Shao (1997), Shi and Tsai (1998), Zheng and Loh (1995), for instance, and Chap. III.1 for a more detailed overview.

Second, the evaluation and estimation of all submodels of a given regression model can be very computationally intensive, especially if the number of variables is large. This motivated tree-like algorithms searching through all submodels, but once they reject a submodel, they automatically reject all models containing only a subset of variables of the rejected submodel, see Edwards and Havranek (1987). These so-called branch-and-bound techniques are discussed in Miller (2002), for instance.

An alternative computational approach, which is increasingly used in applications where the number of explanatory variables is very large, is based on the *genetic programming* (genetic algorithm, GA) approach, see Wasserman and Sudjianto (1994). Similarly to branch-and-bound methods, GAs perform an non-exhaustive search through the space of all submodels. The procedure works as follows. First, each submodel which is represented by a "chromosome" – a $p \times 1$ vector $\boldsymbol{m}_j = \{I_1^j, \ldots, I_p^j\} \in \{0,1\}^p$ of indicators, where I_k^j indicates whether the kth variable is included in the submodel defined by $\boldsymbol{m}_j$. Next, to find the best submodel, one starts with an (initially randomly selected) population $\mathcal{P} = \{\boldsymbol{m}_j\}_{j=1}^J$ of submodels that are compared with each other in terms of information or prediction criteria. Further, this population $\mathcal{P}$ is iteratively modified: in each step, pairs of submodels $\boldsymbol{m}_j, \boldsymbol{m}_{j'} \in \mathcal{P}$ combine their characteristics (chromosomes) to create their offsprings $\boldsymbol{m}_j^*$. This process can have many different forms such as $\boldsymbol{m}_j^* = (\boldsymbol{m}_j + \boldsymbol{m}_{j'} + \boldsymbol{r}_m) \bmod 1$ or $\boldsymbol{m}_j^* = (1, \ldots, 1, 0, \ldots, 0)^\top \boldsymbol{m}_j + (0, \ldots, 0, 1, \ldots, 1)^\top \boldsymbol{m}_{j'} + \boldsymbol{r}_m$, where $\boldsymbol{r}_m$ is a possibly non-zero random mutation. Whenever an offspring $\boldsymbol{m}_j^*$ performs better than its "parent" models $\boldsymbol{m}_j$, $\boldsymbol{m}_j^*$ replaces $\boldsymbol{m}_j$ in population $\mathcal{P}$. Performing this action for all $j = 1, \ldots, J$ creates a new population. By repeating this population renewal, GAs search through the space of all available submodels and keep only the best ones in the population $\mathcal{P}$. Thus, GAs provide a rather effective way of obtaining the best submodel, especially when the number of explanatory variables is very high, since the search is not exhaustive. See Chap. II.6 and Chambers (1998) for a more detailed introduction to genetic programming.

Finally, an estimation strategy related to all-subset regression is the so-called *model averaging*. It also aims to estimate all submodels of a given regression model, but defines the final estimates as weighted averages across parameter estimates in all submodels, where weights are probabilities or measures of fit assigned to each submodel (see Chap. III.11). The all-subset regression represents thus a special case of model averaging, assigning weight one to the best submodel and zero weights to all other submodels. Being more general and frequently used in the context of Bayesian estimation, model averaging obviously presents the same computational difficulties as the all-subsets regression and leads to the use of simulation techniques (Chap. II.3). A recent exception among the model-averaging methods is the weighted-average least squares estimator (Magnus et al. 2010) with the computational complexity increasing only linearly in the number of variables rather exponentially despite averaging across all submodels.

Example 1. We compare several mentioned variable selection methods using a classical data set on air pollution used originally by McDonald and Schwing (1973), who modeled mortality depending on 15 explanatory variables ranging from climate and air pollution to socioeconomic characteristics and who additionally demonstrated instabilities of LS estimates using this data set. We refer to the explanatory variables of data Pollution simply by numbers 1 to 15.

We applied the forward, backward, and all-subset selection procedures to this data set. The results reported in Table 23.1 demonstrate that although all three

Table 23.1 Variables selected from pollution data by different selection procedures. RSS is reported in brackets

Number of variables	Forward selection	Backward elimination	All-subset selection
1	9	9	9
	(133,695)	(133,695)	9 (133,695)
2	6, 9	6, 9	6, 9
	(99,841)	(99,841)	(99,841)
3	2, 6, 9	2, 6, 9	2, 6, 9
	(82,389)	(82,389)	(82,389)
4	2, 6, 9, 14	2, 5, 6, 9	1, 2, 9, 14
	(72,250)	(74,666)	(69,154)
5	1, 2, 6, 9, 14	2, 6, 9, 12, 13	1, 2, 6, 9, 14
	(64,634)	(69,135)	(64,634)

methods could lead to the same subset of variables (e.g., if we search a model consisting of two or three variables), this is not the case in general. For example, searching for a subset of four variables, the variables selected by backward and forward selection differ, and in both cases, the selected model is suboptimal (compared to all-subsets regression) in the sense of the unexplained variance measured by RSS.

Cross Validation

Cross validation (CV) is a general model-selection principle, proposed already in Stone (1974), which chooses a specific model in a similar way as the prediction criteria. CV compares models, which can include all variables or exclude some, based on their out-of-sample performance, which is measured typically by MSE. To achieve this, a sample is split to two disjunct parts: one part is used for estimation and the other part serves for checking the fit of the estimated model on "new" data (i.e., data which were not used for estimation) by comparing the observed and predicted values.

Probably the most popular variant is the leave-one-out cross-validation (LOU CV), which can be used not only for model selection, but also for choosing nuisance parameters (e.g., in nonparametric regression; see Härdle 1992). Assume we have a set of models $\boldsymbol{y} = h_k(\boldsymbol{X}, \boldsymbol{\beta}) + \boldsymbol{\varepsilon}$ defined by regression functions $h_k, k = 1, \ldots, M$, that determine variables included or excluded from regression. For model given by h_k, LOU CV evaluates

$$CV_k = \sum_{i=1}^{n} (y_i - \hat{y}_{i,-i})^2 , \tag{23.7}$$

where $\hat{y}_{i,-i}$ is the prediction at $\boldsymbol{x_i}$ based on the model $\boldsymbol{y_{-i}} = h_k(\boldsymbol{X_{-i}}, \boldsymbol{\beta}) + \boldsymbol{\varepsilon_{-i}}$ and $\boldsymbol{y_{-i}}, \boldsymbol{X_{-i}}, \boldsymbol{\varepsilon_{-i}}$ are the vectors and matrices $\boldsymbol{y}, \boldsymbol{X}, \boldsymbol{\varepsilon}$ without their ith elements and rows, respectively. Thus, all but the ith observation are used for estimation and the ith observation is used to check the out-of-sample prediction. Having evaluated

CV_k for each model, $k = 1, \ldots, M$, we select the model commanding the minimum $\min_{k=1,\ldots,M} CV_k$.

Unfortunately, LOU CV is not consistent as far as the linear model selection is concerned. To make CV a consistent model selection method, it is necessary to omit n_v observations from the sample used for estimation, where $\lim_{n\to\infty} n_v/n = 1$. This fundamental result derived in Shao (1993) places a heavy computational burden on the CV model selection. Since our main use of CV in this chapter concerns nuisance parameter selection, we do not discuss this type of CV any further. See Miller (2002) and Chap. III.1 for further details.

23.1.4 Principle Components Regression

In some situations, it is not feasible to use variable selection to reduce the number of explanatory variables or it is not desirable to do so. The first case can occur if the number of explanatory variables is large compared to the number of observations. The latter case is typical in situations when we observe many characteristics of the same type, for example, temperature or electro-impulse measurements from different sensors on a human body. They could be possibly correlated with each other and there is no a priori reason why measurements at some points of a skull, for instance, should be significant while other ones would not be important at all. Since such data typically exhibit (exact) multicollinearity and we do not want to exclude some or even majority of variables, we have to reduce the dimension of the data in another way.

A general method that can be used both under near and exact multicollinearity is based on the *principle components analysis* (PCA), see Chap. III.6. Its aim is to reduce the dimension of explanatory variables by finding a small number of linear combinations of explanatory variables $\boldsymbol{X}$ that capture most of the variation in $\boldsymbol{X}$ and to use these linear combinations as new explanatory variables instead the original ones. Suppose that $\boldsymbol{G}$ is an orthonormal matrix that diagonalizes matrix $\boldsymbol{X}^\top\boldsymbol{X}$: $\boldsymbol{G}^\top\boldsymbol{G} = \boldsymbol{I}$, $\boldsymbol{X}^\top\boldsymbol{X} = \boldsymbol{G}\boldsymbol{\Lambda}\boldsymbol{G}^\top$, and $\boldsymbol{G}^\top\boldsymbol{X}^\top\boldsymbol{X}\boldsymbol{G} = \boldsymbol{\Lambda}$, where $\boldsymbol{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_p)$ is a diagonal matrix of eigenvalues of $\boldsymbol{X}^\top\boldsymbol{X}$.

Definition 3. Assume without loss of generality that $\lambda_1 \geq \ldots \geq \lambda_p$ and $\boldsymbol{g_1}, \ldots, \boldsymbol{g_p}$ are the corresponding eigenvectors (columns of matrix $\boldsymbol{G}$). Vector $\boldsymbol{z_i} = \boldsymbol{X}\boldsymbol{g_i}$ for $i = 1, \ldots, p$ such that $\lambda_i > 0$ is called the ith principle component (PC) of $\boldsymbol{X}$ and $\boldsymbol{g_i}$ represents the corresponding loadings.

PCA tries to approximate the original matrix $\boldsymbol{X}$ by projecting it into the lower-dimensional space spanned by the first k eigenvectors $\boldsymbol{g_1}, \ldots, \boldsymbol{g_k}$. It can be shown that these projections capture most of the variability in $\boldsymbol{X}$ among all linear combinations of columns of $\boldsymbol{X}$, see Härdle and Simar (2003).

Theorem 2. *There is no standardized linear combination* $\boldsymbol{X}\boldsymbol{a}$, *where* $\|a\| = 1$, *that has strictly larger variance than* $\boldsymbol{z_1} = \boldsymbol{X}\boldsymbol{g_1}$: $Var(\boldsymbol{X}\boldsymbol{a}) \leq Var(\boldsymbol{z_1}) = \lambda_1$. *Additionally, the variance of the linear combination* $\boldsymbol{z} = \boldsymbol{X}\boldsymbol{a}$, $\|a\| = 1$, *that is*

uncorrelated with the first k principle components $\boldsymbol{z_1}, \ldots, \boldsymbol{z_k}$ *is maximized by the* $(k+1)$*-st principle component* $\boldsymbol{z} = \boldsymbol{z_{k+1}}$ *and* $\boldsymbol{a} = \boldsymbol{g_{k+1}}$, $k = 1, \ldots, p-1$.

Consequently, one chooses a number k of PCs that capture a sufficient amount of data variability. This can be done by looking at the ratio $L_k = \sum_{i=1}^{k} \lambda_i / \sum_{i=1}^{p} \lambda_i$, which quantifies the fraction of the variance captured by the first k PCs compared to the total variance of $\boldsymbol{X}$.

In the regression context, the chosen PCs are used as new explanatory variables, and consequently, PCs with small eigenvalues can be important too. Therefore, one can alternatively choose the PCs that exhibit highest correlations with the dependent variable $\boldsymbol{y}$ because the aim is to use the selected PCs for regressing the dependent variable $\boldsymbol{y}$ on them, see Jollife (1982). Moreover, for selecting "explanatory" PCs, it is also possible to use any variable selection method discussed in Sect. 23.1.3. Recently, Hwang and Nettleton (2003) proposed a new data-driven PC selection for PCR obtained by minimizing MSE.

Next, let us assume we selected a small number k of PCs $\boldsymbol{Z_k} = (\boldsymbol{z_1}, \ldots, \boldsymbol{z_k})^\top$ by some rule such that matrix $\boldsymbol{Z_k}^\top \boldsymbol{Z_k}$ has a full rank, $k \leq p$. Then the *principle components regression* (PCR) is performed by regressing the dependent variable $\boldsymbol{y}$ on the selected PCs $\boldsymbol{Z_k}$, which have a (much) smaller dimension than original data $\boldsymbol{X}$, and consequently, multicollinearity is diminished or eliminated, see Gunst and Mason (1980). We estimate this new model by LS,

$$\boldsymbol{y} = \boldsymbol{Z_k}\boldsymbol{\gamma} + \eta = \boldsymbol{X}\boldsymbol{G_k}\boldsymbol{\gamma} + \eta,$$

where $\boldsymbol{G_k} = (\boldsymbol{g_1}, \ldots, \boldsymbol{g_k})^\top$. Comparing it with the original model (23.1) shows that $\boldsymbol{\beta} = \boldsymbol{G_k}\boldsymbol{\gamma}$. It is important to realize that in PCR we first fix $\boldsymbol{G_k}$ by means of PCA and then estimate $\boldsymbol{\gamma}$.

Finally, concerning different PC selection criteria, Barros and Rutledge (1998) demonstrate the superiority of the correlation-based PCR (CPCR) and convergence of many model-selection procedures toward the CPCR results. See also Depczynski et al. (2000) for a similar comparison of CPRC and PCR based on GA variable selection and Heij et al. (2007) for comparison with the closely related principal covariate regression.

Example 2. Let us use data Pollution to demonstrate several important issues concerning PCR. First, we identify PCs of the data. The fraction of variance explained by the first k PCs as a function of k is depicted on Fig. 23.1 (dashed line). On the one side, almost all of the $\boldsymbol{X}$ variance is captured by the first PC. On the other side, the percentage of the $\boldsymbol{y}$ variance explained by the first k PCs (solid line) grows and reaches its maximum relatively slowly. Thus, the inclusion of about 7 PCs seems to be necessary when using this strategy.

On the other hand, using some variable selection method or checking the correlation of PCs with the dependent variable $\boldsymbol{y}$ reveals that PCs 1, 3, 4, 5, 7 exhibit highest correlations with $\boldsymbol{y}$ (higher than 0.25), and naturally, a model using these 5 PCs has more explanatory power ($\bar{R}^2 = 0.70$) than for example the first 6 PCs

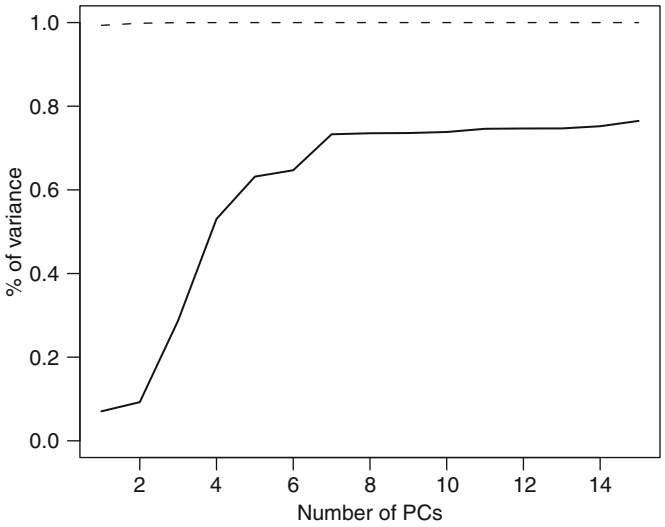


Fig. 23.1 Fraction of the explained variance of $\boldsymbol{X}$ (dashed line) and $\boldsymbol{y}$ (solid line) by the first k PCs

together ($\bar{R}^2 = 0.65$). Thus, considering not only PCs that capture most of the $\boldsymbol{X}$ variability, but also those having large correlations with the dependent variable enables building more parsimonious models.

23.1.5 Shrinkage Estimators

We argued in Sect. 23.1.2 that an alternative way of dealing with unpleasant consequences of multicollinearity lies in biased estimation: we can sacrifice a small bias for a significant reduction in variance of an estimator so that its MSE decreases. Since it holds for an estimator b and a real constant $c \in \mathbb{R}$ that $Var(c\hat{\boldsymbol{\beta}}) = c^2 Var(\hat{\boldsymbol{\beta}})$, a bias of the estimator $\hat{\boldsymbol{\beta}}$ towards zero, $|c| < 1$, naturally leads to a reduction in variance. This observation motivates a whole class of biased estimators – *shrinkage estimators* – that are biased towards zero in all or just some of their components. In other words, they "shrink" the Euclidean norm of estimates compared to that of the corresponding unbiased estimate. This is perhaps easiest to observe on the example of the Stein-rule estimator, which can be expressed in linear regression model (23.1) as

$$\hat{\boldsymbol{\beta}}^{SR} = \left(1 - \frac{k\boldsymbol{e}^\top \boldsymbol{e}}{n\hat{\boldsymbol{\beta}}^{LS\top}\boldsymbol{X}^\top \boldsymbol{X}\hat{\boldsymbol{\beta}}^{LS}}\right)\hat{\boldsymbol{\beta}}^{LS}, \tag{23.8}$$

where $k > 0$ is an arbitrary scalar constant and $\boldsymbol{e}^\top \boldsymbol{e}/n$ represents an estimate of the residual variance (Gruber 1998). Apparently, the Stein-rule estimator just multiplies the LS estimator by a constant smaller than one. See Gruber (1998) and Judge and Bock (1983) for an overview of this and many other biased estimators.

In the following subsections, we discuss various shrinkage estimators that perform well under multicollinearity and that can possibly act as variable selection tools as well: the ridge regression estimator and its modifications (Sect. 23.1.6), the continuum regression (Sect. 23.1.7), the Lasso estimator and its variants (Sect. 23.1.8), the least angle regression (Sect. 23.1.9), and partial least squares (Sect. 23.1.10). Let us note that there are also other shrinkage estimators, which either do not perform well under various forms of multicollinearity (e.g., Stein-rule estimator) or are discussed in other parts of this chapter (e.g., pre-test and PCR estimators in Sects. 23.1.3 and 23.1.4, respectively).

23.1.6 Ridge Regression

Probably the best known shrinkage estimator is the *ridge estimator* proposed and studied by Hoerl and Kennard (1970). Having a non-orthogonal or even nearly singular matrix $\boldsymbol{X}^\top \boldsymbol{X}$, one can add a positive constant $k > 0$ to its diagonal to improve conditioning.

Definition 4. Ridge regression (RR) estimator is defined for model (23.1) by

$$\hat{\boldsymbol{\beta}}^{\boldsymbol{RR}} = (\boldsymbol{X}^\top \boldsymbol{X} + k\boldsymbol{I})^{-1} \boldsymbol{X}^\top \boldsymbol{y} \tag{23.9}$$

for some ridge parameter $k > 0$.

"Increasing" the diagonal of $\boldsymbol{X}^\top \boldsymbol{X}$ before inversion shrinks $\hat{\boldsymbol{\beta}}^{\boldsymbol{RR}}$ compared to $\hat{\boldsymbol{\beta}}^{\boldsymbol{LS}}$ and introduces a bias. Additionally, Hoerl and Kennard (1970) also showed that the derivative of $MSE(\hat{\boldsymbol{\beta}}^{\boldsymbol{RR}})$ with respect to k is negative at $k = 0$. This indicates that the bias

$$Bias(\hat{\boldsymbol{\beta}}^{\boldsymbol{RR}}) = -k(\boldsymbol{X}^\top \boldsymbol{X} + k\boldsymbol{I})^{-1} \boldsymbol{\beta}$$

can be smaller than the decrease in variance (here for a homoscedastic linear model with error variance σ^2)

$$Var(\hat{\boldsymbol{\beta}}^{\boldsymbol{RR}}) - Var(\hat{\boldsymbol{\beta}}^{\boldsymbol{LS}}) = \sigma^2(\boldsymbol{X}^\top \boldsymbol{X} + k\boldsymbol{I})^{-1} \boldsymbol{X}^\top \boldsymbol{X} (\boldsymbol{X}^\top \boldsymbol{X} + k\boldsymbol{I})^{-1} - \sigma^2 (\boldsymbol{X}^\top \boldsymbol{X})^{-1}$$

caused by shrinking at least for some values of k. The intervals for k where RR dominates LS are derived, for example, in Chawla (1990), Gruber (1998), and Rao and Toutenberg (1999). Moreover, the improvement in $MSE(\hat{\boldsymbol{\beta}}^{\boldsymbol{RR}})$ with respect to $MSE(\hat{\boldsymbol{\beta}}^{\boldsymbol{LS}})$ is significant under multicollinearity while being negligible for nearly orthogonal systems. A classical result for model (23.1) under $\boldsymbol{\varepsilon} \sim \mathrm{N}(0, \sigma^2 \boldsymbol{I_n})$ states that $MSE(\hat{\boldsymbol{\beta}}^{\boldsymbol{RR}}) - MSE(\hat{\boldsymbol{\beta}}^{\boldsymbol{LS}}) < 0$ is negative definite if $k < k_{max} = 2\sigma^2/\boldsymbol{\beta}^\top \boldsymbol{\beta}$,

see Vinod and Ullah (1981), where an operational estimate of k_{max} is discussed too. Notice however that the conditions for the dominance of the RR and other some other shrinkage estimators over LS can look quite differently in the case of non-normal errors (Ullah et al. 1983).

In applications, an important question remains: how to choose the ridge parameter k? In the original paper by Hoerl and Kennard (1970), the use of the ridge trace, a plot of the components of the estimated $\hat{\boldsymbol{\beta}}^{\boldsymbol{RR}}$ against k, was advocated. If data exhibit multicollinearity, one usually observes a region of instability for k close to zero and then stable estimates for large values of ridge parameter k. One should choose the smallest k lying in the region of stable estimates. Alternatively, one could search for k minimizing $MSE(\hat{\boldsymbol{\beta}}^{\boldsymbol{RR}})$; see the subsection on the generalized RR for more details. Furthermore, many other methods for model selection could be employed too; for example, LOU CV (Sect. 23.1.3) performed on a grid of k values is often used in this context.

Statistics important for inference based on RR estimates are discussed in Hoerl and Kennard (1970) and Vinod and Ullah (1981) both for the case of a fixed k as well as in the case of some data-driven choices. Moreover, the latter work also describes algorithms for a fast and efficient RR computation.

To conclude, let us note that the RR estimator $\hat{\boldsymbol{\beta}}^{\boldsymbol{RR}}$ in model (23.1) can be also defined as a solution of a restricted minimization problem

$$\hat{\boldsymbol{\beta}}^{\boldsymbol{RR}} = \underset{\hat{\boldsymbol{\beta}}:\|\hat{\boldsymbol{\beta}}\|_2^2 \leq r^2}{\operatorname{argmin}} \ (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})^\top (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}), \tag{23.10}$$

or equivalently as

$$\hat{\boldsymbol{\beta}}^{\boldsymbol{RR}} = \underset{\hat{\boldsymbol{\beta}}}{\operatorname{argmin}} \ (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}})^\top (\boldsymbol{y} - \boldsymbol{X}\hat{\boldsymbol{\beta}}) + k\|\hat{\boldsymbol{\beta}}\|_2^2, \tag{23.11}$$

where r represents a tuning parameter corresponding to k (Swamy et al. 1978). This formulation was used by Ngo et al. (2003), for instance. Moreover, (23.10) reveals one controversial issue in RR: rescaling of the original data to make $\boldsymbol{X}^\top \boldsymbol{X}$ a correlation matrix. Although there are no requirements of this kind necessary for theoretical results, standardization is often recommended to make influence of the constraint $\|\hat{\boldsymbol{\beta}}\|_2^2 \leq r^2$ same for all variables. There are also studies showing adverse effects of this standardization on estimation, see Vinod and Ullah (1981) for a discussion. A possible solution is generalized RR, which assigns to each variable its own ridge parameter (see the next paragraph).

Generalized Ridge Regression

The RR estimator can be generalized in the sense that each diagonal element of $\boldsymbol{X}^\top \boldsymbol{X}$ is modified separately. To achieve that let us recall that this matrix can be

diagonalized: $\boldsymbol{X}^\top \boldsymbol{X} = \boldsymbol{G}^\top \boldsymbol{\Lambda} \boldsymbol{G}$, where $\boldsymbol{G}$ is an orthonormal matrix and $\boldsymbol{\Lambda}$ is a diagonal matrix containing eigenvalues $\lambda_1, \ldots, \lambda_p$.

Definition 5. The generalized ridge regression (GRR) estimator is defined for model (23.1) by

$$\hat{\boldsymbol{\beta}}^{\boldsymbol{GRR}} = (\boldsymbol{X}^\top \boldsymbol{X} + \boldsymbol{G}\boldsymbol{K}\boldsymbol{G}^\top)^{-1} \boldsymbol{X}^\top \boldsymbol{y} \tag{23.12}$$

for a diagonal matrix $\boldsymbol{K} = \operatorname{diag}(k_1, \ldots, k_p)$ of ridge parameters.

The main advantage of this generalization being ridge coefficients specific to each variable, it is important to know how to choose the matrix $\boldsymbol{K}$. In Hoerl and Kennard (1970) the following result is derived.

Theorem 3. *Assume that $\boldsymbol{X}$ in model (23.1) has a full rank, $\boldsymbol{\varepsilon} \sim N(0, \sigma^2 \boldsymbol{I_n})$, and $n > p$. Further, let $\boldsymbol{X} = \boldsymbol{H}\boldsymbol{\Lambda}^{1/2}\boldsymbol{G}^\top$ be the singular value decomposition of $\boldsymbol{X}$ and $\boldsymbol{\gamma} = \boldsymbol{G}^\top \boldsymbol{\beta_0}$. The MSE-minimizing choice of $\boldsymbol{K}$ in (23.12) is $\boldsymbol{K} = \sigma^2 \operatorname{diag}(\gamma_1^{-2}, \ldots, \gamma_p^{-2})$.*

An operational version (feasible GRR) is based on an unbiased estimate $\hat{\gamma}_i = \boldsymbol{G}^\top \hat{\boldsymbol{\beta}}^{\boldsymbol{LS}}$ and $s^2 = (\boldsymbol{y} - \boldsymbol{H}\hat{\gamma})^\top (\boldsymbol{y} - \boldsymbol{H}\hat{\gamma})$. See Hoerl and Kennard (1970) and Vinod and Ullah (1981), where you also find the bias and MSE of this operational GRR estimator, and Wang and Chow (1990) for further extensions of this approach. Let us note that the feasible GRR (FGRR) estimator does not have to possess the MSE-optimality property of GRR because the optimal choice of $\boldsymbol{K}$ is replaced by an estimate. Nevertheless, the optimality property of FGRR is preserved if $\lambda_i \gamma_i^2 \leq 2\sigma^2$, where λ_i is the (i, i)th-element of $\boldsymbol{\Lambda}$ (Farebrother 1976).

Additionally, given an estimate of MSE-minimizing $\hat{\boldsymbol{K}} = \operatorname{diag}(\hat{k}_1, \ldots, \hat{k}_p)$, many authors proposed to choose the ridge parameter k in ordinary RR as a harmonic mean of $\hat{k}_i, i = 1, \ldots, p$; see Hoerl et al. (1975), for instance.

Almost Unbiased Ridge Regression

Motivated by results on GRR, Kadiyala (1984) proposed to correct GRR for its bias using the first-order bias approximation. This yields almost unbiased GRR (AUGRR) estimator

$$\hat{\boldsymbol{\beta}}^{\boldsymbol{AUGRR}} = (\boldsymbol{X}^\top \boldsymbol{X} + \boldsymbol{G}\boldsymbol{K}\boldsymbol{G}^\top)^{-1} (\boldsymbol{X}^\top \boldsymbol{y} + \boldsymbol{K}\boldsymbol{G}^\top \boldsymbol{\beta_0}).$$

The true parameter value $\boldsymbol{\beta_0}$ being unknown, Ohtani (1986) defined a feasible AUFGRR estimator by replacing the unknown $\boldsymbol{\beta_0}$ by $\hat{\boldsymbol{\beta}}^{\boldsymbol{FGRR}}$ and $\boldsymbol{K}$ by the employed ridge matrix. Additionally, a comparison of the FGRR and feasible AUGRR estimators with respect to MSE proved that FGRR has a smaller MSE than AUGRR in a wide range of parameter space. Similar observation was also done under a more general loss function in Wan (2002). Furthermore, Akdeniz et al. (2004) derived exact formulas for the moments of the feasible AUGRR estimator.

Further Extensions

RR can be applied also under exact multicollinearity, which arises for example in data with more variables than observations. Although the theory and application of RR is the same as in the case of full-rank data, the computational burden of $O(np^2 + p^3)$ operations becomes too high for $p > n$. A faster algorithm with computational complexity only $O(np^2)$ was found by Hawkins and Yin (2002).

Moreover, there are many other extensions of the RR principle that go beyond the extent of this chapter. To mention at least some of them, let us refer a reader to works comparing or combining various ridge and shrinkage approaches (Kibria 1996; Lipovetski 2010; Shiaishi and Konno 1995; Singh et al. 1994) and to monograph by Gruber (1998). Further, even though RR is presented here as an alternative tool to variable selection for dealing with multicollinearity, variable selection can be performed also in the context of RR (Yanagihara and Satoh 2010).

Example 3. Using data Pollution once again, we estimated RR for the ridge parameter $k \in (0, 10)$ and plotted the estimated coefficients $\hat{\boldsymbol{\beta}}^{RR}$ as functions of k (ridge trace plot), see Fig. 23.2. For the sake of simplicity, we restricted ourselves only to variables that were selected by some variable selection procedure in Table 23.1 (1, 2, 6, 9, 12, 13, 14). The plot shows the effect of the ridge parameter k on slope estimates ($k = 0$ corresponds to LS). Apparently, slopes of some variables are affected very little (e.g., variable 1), some significantly (e.g., the magnitude of variable 14 increases more than twice), and some variables shrink

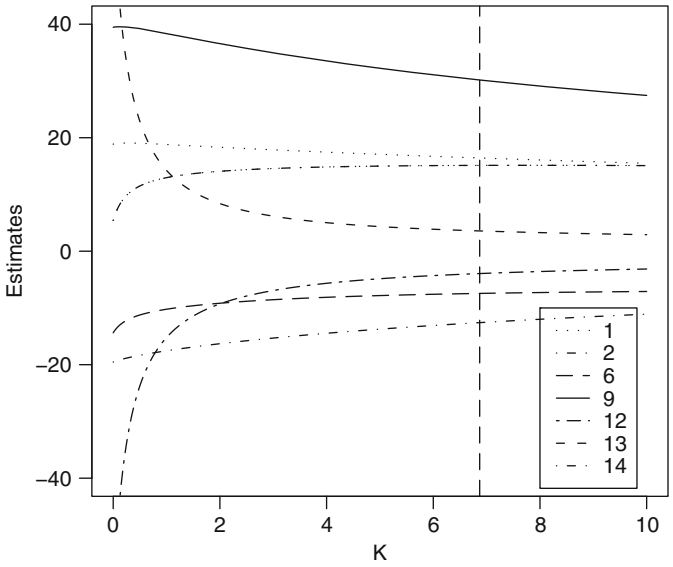


Fig. 23.2 Ridge trace plot for variables 1, 2, 6, 9, 12, 13, 14 of data Pollution. The vertical line represents the CV-choice of k

extremely (e.g., variables 12 and 13). In all cases, the biggest change occurs between $k = 0$ and $k = 2$ and estimates gradually stabilize for $k > 2$. The vertical dashed line in Fig. 23.2 represents the CV estimate of k ($k_{CV} = 6.87$).

23.1.7 Continuum Regression

RR discussed in Sect. 23.1.6 is very closely connected with the *continuum regression* proposed by Brooks and Stone (1990) as a unifying approach to the LS, PCR, and partial least squares (see Sect. 23.1.10) estimation.

Definition 6. A continuum regression (CR) estimator $\hat{\boldsymbol{\beta}}^{CR}(\alpha)$ of model (23.1) is a coefficient vector maximizing function

$$T_\alpha(\boldsymbol{c}) = (\boldsymbol{c}^\top \boldsymbol{s})^2 (\boldsymbol{c}^\top \boldsymbol{S}\boldsymbol{c})^{\alpha-1} = (\boldsymbol{c}^\top \boldsymbol{X}^\top \boldsymbol{y})^2 (\boldsymbol{c}^\top \boldsymbol{X}^\top \boldsymbol{X}\boldsymbol{c})^{\alpha-1}, \tag{23.13}$$

for a given value of parameter $\alpha \geq 0$ and a given length $\|c\|$, where $\boldsymbol{S} = \boldsymbol{X}^\top \boldsymbol{X}$ and $\boldsymbol{s} = \boldsymbol{X}^\top \boldsymbol{y}$.

This definition yields estimates proportional to LS for $\alpha = 0$, to PCR for $\alpha \to \infty$, and to yet-to-be-discussed partial least squares for $\alpha = 1$. Apart from this, the advantage of CR is that one can adaptively select among the methods by searching an optimal α. To determine α, Brooks and Stone (1990) used CV.

The relationship between RR and CR was indicated already in Sundberg (1993), but the most important result came after uncovering possible discontinuities of CR estimates as a function of data and α by Bjorkstrom and Sundberg (1996). In an attempt to remedy the discontinuity of the original CR, Bjorkstrom and Sundberg (1999) not only proposed to maximize

$$T_\delta(\boldsymbol{c}) = (\boldsymbol{c}^\top \boldsymbol{s})^2 (\boldsymbol{c}^\top \boldsymbol{S}\boldsymbol{c})^{-1} |\boldsymbol{c}^\top \boldsymbol{S}\boldsymbol{c} + \delta|^{-1},$$

for $\delta \geq 0$ instead of $T_\alpha(\boldsymbol{c})$ from Def. 6 (δ can be chosen by CV), but also proved the following proposition.

Theorem 4. *If a regressor $\boldsymbol{b}_f$ is defined according to*

$$\boldsymbol{b}_f = \underset{\|\boldsymbol{c}\|=1}{\operatorname{argmax}} f\{K^2(\boldsymbol{c}), V(\boldsymbol{c})\},$$

where $K(\boldsymbol{c}) = \boldsymbol{y}^\top \boldsymbol{X}\boldsymbol{c}$, $V(\boldsymbol{c}) = \|\boldsymbol{X}\boldsymbol{c}\|^2$, $f(K^2, V)$ is increasing in K^2 for constant V, and increasing in V for constant K^2, and finally, if $\boldsymbol{X}^\top \boldsymbol{y}$ is not orthogonal to all eigenvectors corresponding to the largest eigenvalue λ_{max} of $\boldsymbol{X}^\top \boldsymbol{X}$, then there exists a number $k \in (-\infty, \lambda_{max}) \cup [0, +\infty]$ such that $\boldsymbol{b}_f$ is proportional to $(\boldsymbol{X}^\top \boldsymbol{X} + k\boldsymbol{I})^{-1} \boldsymbol{X}^\top \boldsymbol{y}$, including the limiting cases $k \to 0, k \to \pm\infty$, and $k \to -\lambda_{max}$.

Thus, the RR estimator fundamentally underlies many methods dealing with multicollinear and reduced rank data such as mentioned PCR and partial least squares. Notice however that negative values of the ridge coefficient k have to be admitted here.

Finally, let us note that CR can be extended to multiple-response-variables models (Brooks and Stone 1994).

23.1.8 Lasso

The ridge regression discussed in Sect. 23.1.6 motivates another shrinkage method: *Lasso* (least absolute shrinkage and selection operator) by Tibshirani (1996). Formulation (23.10) states that RR can be viewed as a minimization with respect to an upper bound on the L_2 norm of estimate $\|\hat{\boldsymbol{\beta}}\|_2$. A natural extension is to consider constraints on the L_q norm $\|\hat{\boldsymbol{\beta}}\|_q$, $q > 0$, or even more general penalization functions (cf. Fan and Li 2001). Specifically, Tibshirani (1996) studied case of $q = 1$, that is L_1 norm.

Definition 7. The Lasso estimator for the regression model (23.1) is defined by

$$\hat{\boldsymbol{\beta}}^{L} = \underset{\|\boldsymbol{\beta}\|_1 \leq r}{\operatorname{argmin}} \ (\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta})^{\top}(\boldsymbol{y} - \boldsymbol{X}\boldsymbol{\beta}), \tag{23.14}$$

where $r \geq 0$ is a tuning parameter.

Lasso is a shrinkage estimator that has one specific feature compared to the ordinary RR. Because of the geometry of L_1-norm restriction, Lasso shrinks the effect of some variables and eliminates influence of the others, that is, sets their coefficients to zero. Thus, it combines regression shrinkage with variable selection, and as Tibshirani (1996) demonstrated also by means of simulation, it compares favorably to all-subsets regression. (It is even possible to induce an order in which Lasso introduces the variables in the model as shown by Tibshirani et al. 2005.) In this context, it is interesting that Lasso could be formulated as a special case of the least angle regression by Efron et al. (2004), see Sect. 23.1.9. Finally, let us note that to achieve the same kind of shrinking and variable-selection effects for all variables, they should be standardized before used in Lasso; see Miller (2002) for details.

As far as the inference for the Lasso estimator is concerned, Knight and Fu (2000) studied its asymptotic distribution using L_q-norm condition $\|\boldsymbol{\beta}\|_q \leq r$ with $q \leq 1$, including behavior under nearly-singular designs. These results were recently complemented by the study of the finite-sample and asymptotic distributions of Lasso-type estimators by Potscher and Leeb (2009), who show that Lasso tuned to work as a consistent model-selection procedure cannot (uniformly) reach the usual $\sqrt{n}$ rate of convergence. These results are important for the extensions of Lasso aiming at joint estimation and model selection (e.g., Wang et al. 2007; Zou 2006).

Now, it remains to find out how Lasso estimates can be computed. Equation (23.14) indicates that one has to solve a restricted quadratic optimization problem. Setting $\beta_j^+ = \max\{\beta_j, 0\}$ and $\beta_j^- = -\min\{\beta_j, 0\}$, the restriction $\|\boldsymbol{\beta}\| \leq r$ can be written as $2p + 1$ constraints: $\beta_j^+ \geq 0, \beta_j^- \geq 0$, and $\sum_{j=1}^{p}(\beta_j^+ - \beta_j^-) \leq r$. Thus, convergence is assured in $2p + 1$ steps. Additionally, the unknown tuning parameter r is to be selected by means of CV. Further, although solving (23.14) is straightforward in usual regression problems, it can become very demanding for reduced-rank data, $p > n$. Osborne et al. (1999) treated lasso as a convex programming problem, and by formulating its dual problem, developed an efficient algorithm usable even for $p > n$.

Example 4. Let us use data Pollution once more to exemplify the use of Lasso. To summarize the Lasso results, we use the same plot as Tibshirani (1996) and Efron et al. (2004) used, see Fig. 23.3. It contains standardized slope estimates as a function of the constraint $\|b\| \leq r$, which is represented by an index $r/\max\|\hat{\boldsymbol{\beta}}\| = \|\hat{\boldsymbol{\beta}}^{\boldsymbol{L}}\|/\|\hat{\boldsymbol{\beta}}^{\boldsymbol{LS}}\|$ (the LS estimate $\hat{\boldsymbol{\beta}}^{\boldsymbol{LS}}$ corresponds to $\hat{\boldsymbol{\beta}}^{\boldsymbol{L}}$ under $r = \infty$, and thus, renders the maximum of $\|\hat{\boldsymbol{\beta}}^{\boldsymbol{L}}\|$). Moreover, to keep the graph simple, we plotted again only variables that were selected by variable selection procedures in Table 23.1 (1, 2, 6, 9, 12, 13, 14).

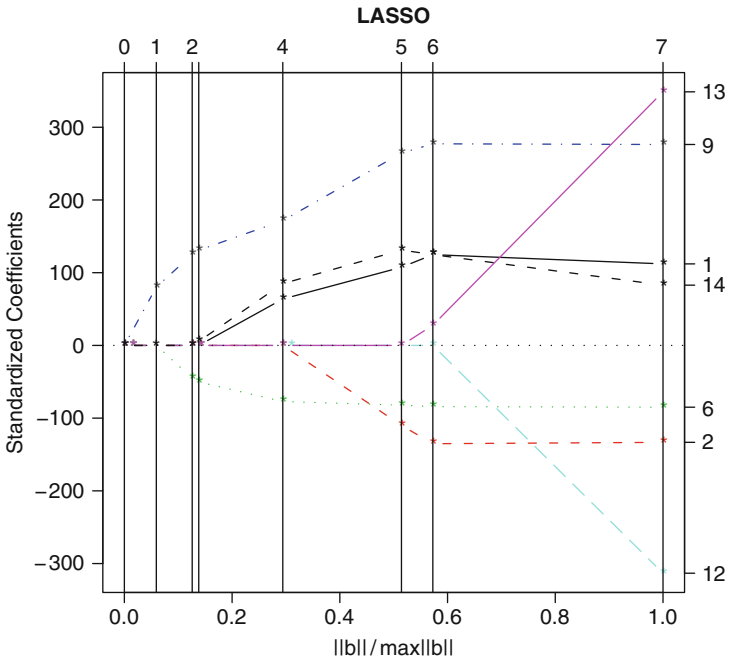


Fig. 23.3 Slope coefficients for variables 1, 2, 6, 9, 12, 13, 14 of data Pollution estimated by Lasso at different constraint levels, $r/\max\|\hat{\boldsymbol{\beta}}\|$. The right axis assigns to each line the number of variable it represents and the top axis indicates the number of variables included in the regression

In Fig. 23.3, we can observe which variables are included in the regression (have a nonzero coefficient) as tuning parameter r increases. Clearly, the order in which the first of these variables become significant – 9, 6, 14, 1, 2 – closely resembles the results of variable selection procedures in Table 23.1. Thus, Lasso combines shrinkage estimation and variable selection: at a given constraint level r, it shrinks coefficients of some variables and removes the others by setting their coefficients equal to zero.

23.1.9 Least Angle Regression

In Sect. 23.1.3, the forward selection was described as overly greedy and consequently ignoring possibly useful variables. A simple modification of the forward selection, where one always adds to the regression function only an "ε-fraction" of the selected variable for some small $\varepsilon > 0$, leads to the *forward stagewise regression*, and after finding the optimal size of the step ε, to the *least angle regression* (LARS) proposed by Efron et al. (2004). Thus, the below described LARS algorithm represents a generalization of the forward selection procedure. At the same time, Efron et al. (2004) also showed that LARS provides a fast way to compute all Lasso estimates, that is, estimates for all values of r in (23.14) can be computed just in p steps, where p is the number of explanatory variables.

To describe the LARS algorithm, let $\hat{\boldsymbol{\mu}}_k$ denote the regression function, I_k the set of variables selected in the kth step of the algorithm, and $\boldsymbol{X_k} = \{s_j \boldsymbol{x_{\cdot j}}\}_{j \in I_k}$, where s_j is a sign defined below. Starting from an "empty" regression function $\hat{\boldsymbol{\mu}}_0$ for $k = 0$, LARS adds in each step an additional variable. For a given $k = 0, \ldots, p-1$, the selection is done based on the vector of the correlations $\hat{\boldsymbol{c}} = \boldsymbol{X}^\top(\boldsymbol{y} - \hat{\boldsymbol{\mu}}_k)$ of variables $\boldsymbol{X}$ with the current regression residuals. For these correlations, let the signs $s_j = \operatorname{sign}(\hat{\boldsymbol{c}}_j)$ and the set $I_k = \{j : |\hat{\boldsymbol{c}}_j| = \hat{\boldsymbol{C}}\}$, where $\hat{\boldsymbol{C}} = \max_j |\hat{\boldsymbol{c}}_j|$, contain the variables with the highest current correlations. All these most correlated variables are now added to the regression function: $\hat{\boldsymbol{\mu}}_{k+1} = \hat{\boldsymbol{\mu}}_k + \gamma \boldsymbol{v_k}$, where the combination of variables is determined is such a way that $\boldsymbol{v_k}$ and $\boldsymbol{X_j}$ form the same angle for all $j \in I_k$:

$$\boldsymbol{v_k} = \boldsymbol{X_k}[\boldsymbol{A_k}(\boldsymbol{X_k}^\top \boldsymbol{X_k})^{-1} \boldsymbol{I}_{|I_k|}],$$

where $\boldsymbol{A_k} = \{\boldsymbol{I}_{|I_k|}^\top (\boldsymbol{X_k}^\top \boldsymbol{X_k})^{-1} \boldsymbol{I}_{|I_k|}\}^{-1/2}$. The step γ is chosen so that I_{k+1} computed for $\hat{\boldsymbol{\mu}}_{k+1}$ in the next step will contain more variables than I_k:

$$\gamma = \min{}^{+}_{j \in \{1,\ldots,n\} \setminus I_k} \left\{ \frac{\hat{\boldsymbol{C}} - \hat{\boldsymbol{c}}_j}{\boldsymbol{A_k} - \boldsymbol{a_j}}, \frac{\hat{\boldsymbol{C}} + \hat{\boldsymbol{c}}_j}{\boldsymbol{A_k} + \boldsymbol{a_j}} \right\},$$

where $\boldsymbol{a_k} = \boldsymbol{X}^\top \boldsymbol{v_k}$ and $\min^+$ means that the minimum is taken only over positive arguments.

The described form of LARS leads to a similar estimation procedure and similar outputs as Lasso (see Fig. 23.3), but does not produce exactly the same solutions as

Lasso. To achieve the equivalence of the two methods, the size of γ has to be shorter in some steps; see Efron et al. (2004). Similarly, the LARS algorithm produces the stagewise-regression result only if the changes of the regression coefficients are restricted to have the same signs as the current correlations of the explanatory variables and regression residuals at each step of the algorithm.

23.1.10 Partial Least Squares

A general modeling approach to most of the methods covered so far was CR in Sect. 23.1.7, whereby it has two "extremes": LS for $\alpha = 0$ and PCR for $\alpha \to \infty$. The *partial least squares* (PLS) regression lies in between – it is a special case of (23.13) for $\alpha = 1$, see Brooks and Stone (1990). Originally proposed by Wold (1966), it was presented as an algorithm that searches for linear combinations of explanatory variables best explaining the dependent variable. Similarly to PCR, PLS also aims especially at situations when the number of explanatory variables is large compared to the number of observations. Here we present the PLS idea and algorithm themselves as well as the latest results on variable selection and inference in PLS.

Having many explanatory variables $\boldsymbol{X}$, the aim of the PLS method is to find a small number of linear combinations $\boldsymbol{T_1} = \boldsymbol{X c_1}, \ldots, \boldsymbol{T_q} = \boldsymbol{X c_q}$ of these variables, thought about as latent variables, explaining observed responses

$$\hat{\boldsymbol{y}} = \hat{\beta}_0 + \sum_{j=1}^{q} \boldsymbol{T}_j \hat{\beta}_j \tag{23.15}$$

(see Garthwaite 1994; Helland 2001). Thus, similarly to PCR, PLS reduces the dimension of data, but the criterion for searching linear combinations is different. Most importantly, it does not depend only on $\boldsymbol{X}$ values, but on $\boldsymbol{y}$ too.

Let us now present the PLS algorithm itself, which defines yet another shrinkage estimator as shown by Coutis (1996) and Jong (1995). (See Rao and Toutenberg 1999, for more details, Garthwaite 1994, for an alternative formulation, and Zhang et al. 2004, for weighted versions of PLS.) The indices $\boldsymbol{T_1}, \ldots, \boldsymbol{T_q}$ are constructed one after another. Estimating the intercept by $b_0 = \bar{\boldsymbol{y}}$, let us start with centered variables $\boldsymbol{z_0} = \boldsymbol{y} - \bar{\boldsymbol{y}}$ and $\boldsymbol{U_0} = \boldsymbol{X} - \bar{\boldsymbol{X}}$ and set $k = 1$.

1. Define the index $\boldsymbol{T_k} = \boldsymbol{U_{k-1}}(\boldsymbol{U_{k-1}^\top z_{k-1}})$. This linear combination is given by the covariance of the unexplained part of the response variable $\boldsymbol{z_{k-1}}$ and the unused part of explanatory variables $\boldsymbol{U_{k-1}}$.
2. Regress the current explanatory matrix $\boldsymbol{U_{k-1}}$ on index $\boldsymbol{T_k}$

$$\boldsymbol{w_k} = (\boldsymbol{T_k^\top T_k})^{-1} \boldsymbol{T_k^\top U_{k-1}}$$

and the yet-unexplained part of response z_{k-1} on index T_k

$$\hat{\beta}_k = (T_k^\top T_k)^{-1} T_k^\top z_{k-1},$$

thus obtaining the kth regression coefficient.

3. Compute residuals, that is the remaining parts of explanatory and response variables: $U_k = U_{k-1} - T_k w_k$ and $z_k = z_{k-1} - T_k b_k$. This implies that the indices T_k and T_l are not correlated for $k < l$.
4. Iterate by setting $k = k + 1$ or stop if $k = q$ is large enough.

This algorithm provides us with indices T_k, which define the analogs of principle components in PCR, and the corresponding regression coefficients b_k in (23.15). The main open question is how to choose the number of components q. The original method proposed by Wold (1978) is based on cross validation. Provided that CV_k from (23.7) represents the CV index of PLS estimate with k factors, an additional index T_{k+1} is added if Wold's R criterion $R = CV_{k+1}/CV_k$ is smaller than 1. This selects the first local minimum of the CV index, which is superior to finding the global minimum of CV_k as shown in Osten (1988). Alternatively, one can stop already when Wold's R exceeds 0.90 or 0.95 bound (modified Wold's R criteria) or to use other variable selection criteria such as AIC. A recent simulation study by Li et al. (2002) showed that modified Wold's R is preferable to Wold's R and AIC. Furthermore, similarly to PCR, there are attempts to use GA for the component selection, see Leardi and Gonzales (1998), for instance.

Next, one of the first results on the asymptotic behavior of PLS stems from Helland and Almoy (1994). The covariance matrix, confidence and prediction intervals based on PLS estimates were first studied by Denham (1997), but a more compact expression was presented in Phatak et al. (2002). There are also attempts to find a sample-specific prediction error of PLS, which were compared by Faber et al. (2003). More importantly, while PLS is widely applied in data with large numbers of covariates, Butler and Denham (2000) showed theoretically and practically that PLS can severaly break down for data exhibiting particular covariance structures and has to be thus used with caution, especially in the presence of multicollinearity.

Finally, note that there are many extensions of the presented algorithm, which is usually denoted PLS1. First of all, there are extensions (PLS2, SIMPLS, etc.) of PLS1 to models with multiple dependent variables, see Jong (1993) and Frank et al. (1993) for instance, which choose linear combinations (latent variables) not only within explanatory variables, but does the same also in the space spanned by dependent variables. A recent survey of these and other so-called two-block methods is given in Wegelin (2000). PLS was also adapted for on-line process modeling, see Qin (1997) for a recursive PLS algorithm. Additionally, in an attempt to simplify the interpretation of PLS results, Trygg and Wold (2002) proposed orthogonalized PLS. Finally, PLS can be adapted for variable selection and seem to outperform similar modifications of Lasso and PCR, especially under multicollinearity (Chong and Jun 2005).

Example 5. Let us use again data Pollution, although it is not a typical application of PLS. As explained in Sects. 23.1.7 and 23.1.10, PLS and PCR are both based on the same principle (searching for linear combinations of original variables), but use different objective functions. To demonstrate, we estimated PLS for 1 to 15 latent variables and plotted the fraction of the $\boldsymbol{X}$ and $\boldsymbol{y}$ variance explained by the PLS latent variables in the same way as in Fig. 23.1. Both curves are in Fig. 23.4. Almost all of the variability in $\boldsymbol{X}$ is captured by the first latent variable, although this percentage is smaller than in the case of PCR. On the other hand, the percentage of the variance of $\boldsymbol{y}$ explained by the first k latent variables increases faster than in the case of PCR, see Fig. 23.4 (solid vs. dotted line).

23.1.11 Comparison of the Methods

Methods discussed in Sects. 23.1.3–23.1.10 are aiming at the estimation of (nearly) singular problems and they are often very closely related, see Sect. 23.1.7. Here we provide several references to studies comparing the discussed methods.

First, an extensive simulation study comparing variable selection, PCR, RR, and PLS regression methods is presented in Frank et al. (1993). Although the results are conditional on the simulation design used in the study, they indicate that PCR, RR, and PLS are, in the case of ill-conditioned problems, highly preferable to

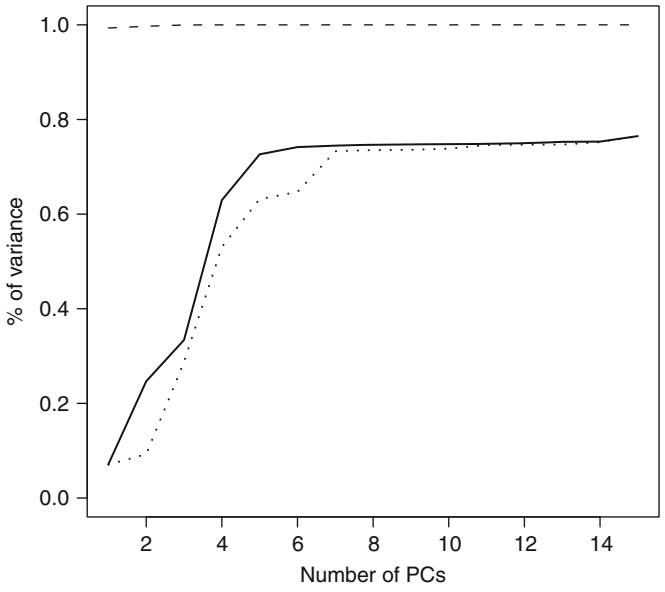


Fig. 23.4 Fraction of the explained variance of $\boldsymbol{X}$ (*dashed line*) and $\boldsymbol{y}$ (*solid line*) by the first k latent variables in PLS regression and by first k PCs (*dotted lines*)

variable selection. The differences between the best methods, RR and PLS, are rather small and the same holds for comparison of PLS and PCR, which seems to be slightly worse than RR. An empirical comparison of PCR and PLS was also done by Wentzell and Montoto (2003) with the same result. Next, the fact that neither PCR, nor PLS asymptotically dominates the other method was proved in Helland and Almoy (1994) and further discussed in Helland (2001). A similar asymptotic result was also given by Stoica and Soderstrom (1998). Finally, the fact that RR should not perform worse than PCR and PLS is supported by Theorem 4 in Sect. 23.1.7.

23.2 Nonlinear Regression Modeling

In this section, we study the nonlinear regression model

$$y_i = h(\boldsymbol{x_i}, \boldsymbol{\beta_0}) + \varepsilon_i, \tag{23.16}$$

$i = 1, \ldots, n$, where $h : R^p \times \mathbb{R}^k \to \mathbb{R}$ is a known regression function and $\boldsymbol{\beta_0}$ is a vector of k unknown parameters. Let us note that the methods discussed in this section are primarily meant for truly nonlinear models rather than intrinsically linear models. A regression model is called intrinsically linear if it can be unambiguously transformed to a model linear in parameters. For example, the regression model $y = \beta_1 x/(\beta_2 + x)$ can be expressed as $1/y = 1/\beta_1 + \beta_2/\beta_1 x$, which is linear in parameters $\boldsymbol{\theta}_1 = 1/\beta_1$ and $\boldsymbol{\theta}_2 = \beta_2/\beta_1$. Transforming a model to its linear form can often provide better inference, such as confidence regions, although one has to be aware of the effects of the transformation on the error-term distribution.

We first discuss the fitting and inference in the nonlinear regression (Sects. 23.2.1 and 23.2.2), whereby we again concentrate on the least square estimation. For an extensive discussion of theory and practice of nonlinear least squares regression see monographs Amemiya (1983), Bates and Watts (1988), and Seber and Wild (2003). Second, similarly to the linear modeling section, methods for ill-conditioned nonlinear systems are briefly reviewed in Sect. 23.2.3.

23.2.1 Fitting of Nonlinear Regression

In this section, we concentrate on estimating the vector $\boldsymbol{\beta_0}$ of unknown parameters in (23.16) by *nonlinear least squares*.

Definition 8. The nonlinear least squares (NLS) estimator for the regression model (23.16) is defined by

$$\hat{\boldsymbol{\beta}}^{NLS} = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{i=1}^{n} \{y_i - \hat{y}_i(\boldsymbol{\beta})\}^2 = \underset{\boldsymbol{\beta} \in \mathbb{R}^p}{\operatorname{argmin}} \sum_{i=1}^{n} \{y_i - h(\boldsymbol{x_i}, \boldsymbol{\beta})\}^2. \tag{23.17}$$

Contrary to the linear model fitting, we cannot express analytically the solution of this optimization problem for a general function h. On the other hand, we can try to approximate the nonlinear objective function using the Taylor expansion because the existence of the first two derivatives of h is an often used condition for the asymptotic normality of NLS, and thus, could be readily assumed. Denoting $\boldsymbol{h}(\hat{\boldsymbol{\beta}}) = \{h(\boldsymbol{x_i}, \hat{\boldsymbol{\beta}})\}_{i=1}^n$ and $S_n(\hat{\boldsymbol{\beta}}) = \sum_{i=1}^n [y_i - h(\boldsymbol{x_i}, \hat{\boldsymbol{\beta}})]^2$, we can state the following theorem from Amemiya (1985).

Theorem 5. *Let ε_i in (23.16) are independent and identically distributed with $E(\boldsymbol{\varepsilon}|\boldsymbol{X}) = 0$ and $Var(\boldsymbol{\varepsilon}|\boldsymbol{X}) = \sigma^2 \boldsymbol{I_n}$ and let B be an open neighborhood of $\boldsymbol{\beta_0}$. Further, assume that $h(\boldsymbol{x}, \boldsymbol{\beta})$ is continuous on B uniformly with respect to $\boldsymbol{x}$ and twice continuously differentiable in B and that:*

1. $\lim_{n\to\infty} S_n(\boldsymbol{\beta}) \neq 0$ *for* $\boldsymbol{\beta} \neq \boldsymbol{\beta_0}$;
2. $[\partial \boldsymbol{h}(\boldsymbol{\beta})/\partial \boldsymbol{\beta}^\top]^\top [\partial \boldsymbol{h}(\boldsymbol{\beta})/\partial \boldsymbol{\beta}^\top]/n$ *converges uniformly in B to a finite matrix $\boldsymbol{A}(\boldsymbol{\beta})$, such that $\boldsymbol{A}(\boldsymbol{\beta_0})$ is nonsingular;*
3. $\boldsymbol{h}(\boldsymbol{\beta^1})^\top [\partial^2 \boldsymbol{h}(\boldsymbol{\beta^2})/\partial\beta_j \partial\beta_k]/n$ *converges uniformly for $\boldsymbol{\beta^1}, \boldsymbol{\beta^2} \in B$ to a finite matrix for all $j, k = 1, \ldots, k$.*

Then the NLS estimator $\hat{\boldsymbol{\beta}}^{\boldsymbol{NLS}}$ is consistent and asymptotically normal

$$\sqrt{n}\left(\hat{\boldsymbol{\beta}}^{\boldsymbol{NLS}} - \boldsymbol{\beta_0}\right) \to N(0, \sigma^2 \boldsymbol{A}(\boldsymbol{\beta_0})^{-1}). \tag{23.18}$$

Hence, although there is no general explicit solution to (23.17), we can assume without loss of much generality that the objective function $S_n(\hat{\boldsymbol{\beta}})$ is twice differentiable in order to devise a numerical optimization algorithm. The second-order Taylor expansion provides then a quadratic approximation of the minimized function, which can be used for obtaining an approximate minimum of the function, see Amemiya (1983). As a result, one should search in the direction of the steepest descent of a function, which is given by its gradient, to get a better approximation of the minimum. We discuss here the incarnations of these methods specifically for the case of the quadratic loss function in (23.17).

Newton's Method

The classical method based on the gradient approach is Newton's method, see Kennedy and Gentle (1980) and Amemiya (1983) for detailed discussion. Starting from an initial point $\hat{\boldsymbol{\beta}}^{\boldsymbol{1}}$, a better approximation is found by taking

$$\begin{aligned} \hat{\boldsymbol{\beta}}^{\boldsymbol{k+1}} &= \hat{\boldsymbol{\beta}}^{\boldsymbol{k}} - \boldsymbol{H}^{-1}(\boldsymbol{r}^2, \hat{\boldsymbol{\beta}}^{\boldsymbol{k}}) \boldsymbol{J}(\boldsymbol{r}, \hat{\boldsymbol{\beta}}^{\boldsymbol{k}}) = \\ &= \hat{\boldsymbol{\beta}}^{\boldsymbol{k}} - \left[\boldsymbol{J}(\boldsymbol{h}, \hat{\boldsymbol{\beta}}^{\boldsymbol{k}})^\top \boldsymbol{J}(\boldsymbol{h}, \hat{\boldsymbol{\beta}}^{\boldsymbol{k}}) + \sum_{l=1}^{n} r_i(\hat{\boldsymbol{\beta}}) \boldsymbol{H}(h_i, \hat{\boldsymbol{\beta}}^{\boldsymbol{k}}) \right]^{-1} \boldsymbol{J}(\boldsymbol{h}, \hat{\boldsymbol{\beta}}^{\boldsymbol{k}})^\top \boldsymbol{r}(\hat{\boldsymbol{\beta}}^{\boldsymbol{k}}), \end{aligned} \tag{23.19}$$

where $\boldsymbol{r}(\boldsymbol{\beta}) = \{[y_i - h(x_i, \boldsymbol{\beta})]\}_{i=1}^n$ represents the vector of residuals, $\boldsymbol{J}(\boldsymbol{f}, \boldsymbol{\beta}) = \partial \boldsymbol{f}(\boldsymbol{\beta})/\partial \boldsymbol{\beta}^\top$ is the Jacobian matrix of a vector function $\boldsymbol{f}(\boldsymbol{\beta})$, and $\boldsymbol{H}(\boldsymbol{f}, \boldsymbol{\beta}) = \partial^2\{\sum_{i=1}^n \boldsymbol{f}_i(\boldsymbol{\beta})\}/\partial\boldsymbol{\beta}\partial\boldsymbol{\beta}^\top$ is the Hessian matrix of the sum of $\boldsymbol{f}(\boldsymbol{\beta})$.

To find $\hat{\boldsymbol{\beta}}^{\boldsymbol{NLS}}$, (23.19) is iterated until convergence is achieved. This is often verified by checking whether the relative change from $\hat{\boldsymbol{\beta}}^{\boldsymbol{k}}$ to $\hat{\boldsymbol{\beta}}^{\boldsymbol{k+1}}$ is sufficiently small. Unfortunately, this criterion can indicate a lack of progress rather than convergence. Instead, Bates and Watts (1988) proposed to check convergence by looking at some measure of orthogonality of residuals $\boldsymbol{r}(\hat{\boldsymbol{\beta}}^{\boldsymbol{k}})$ towards the regression surface given by $\boldsymbol{h}(\hat{\boldsymbol{\beta}}^{\boldsymbol{k}})$, since the identification assumption of model (23.16) is $E(\boldsymbol{r}(\boldsymbol{\beta_0})|\boldsymbol{X}) = 0$. See Bjorck (1996), Kennedy and Gentle (1980), and Thisted (1988) for more details and further modifications.

To evaluate iteration (23.19), it is necessary to invert the Hessian matrix $\boldsymbol{H}(\boldsymbol{r}^2, \boldsymbol{\beta})$. From the computational point of view, all issues discussed in Sect. 23.1 apply here too and one should use a numerically stable procedure, such as QR or SVD decompositions, to perform the inversion. Moreover, to guarantee that (23.19) leads to a better approximation of the minimum, that is $\boldsymbol{r}(\hat{\boldsymbol{\beta}}^{\boldsymbol{k+1}})^\top \boldsymbol{r}(\hat{\boldsymbol{\beta}}^{\boldsymbol{k+1}}) \leq \boldsymbol{r}(\hat{\boldsymbol{\beta}}^{\boldsymbol{k}})^\top \boldsymbol{r}(\hat{\boldsymbol{\beta}}^{\boldsymbol{k}})$, the Hessian matrix $\boldsymbol{H}(\boldsymbol{r}^2, \hat{\boldsymbol{\beta}}^{\boldsymbol{k}})$ needs to be positive definite, which in general holds only in a neighborhood of $\boldsymbol{\beta_0}$ (see the Levenberg-Marquardt method for a remedy). Even if it is so, the step in the gradient direction should not be too long, otherwise we "overshoot." Modified Newton's method addresses this by using some fraction α_{k+1} of iteration step $\hat{\boldsymbol{\beta}}^{\boldsymbol{k+1}} = \hat{\boldsymbol{\beta}}^{\boldsymbol{k}} - \alpha_{k+1}\boldsymbol{H}^{-1}(\boldsymbol{r}^2, \hat{\boldsymbol{\beta}}^{\boldsymbol{k}})\boldsymbol{J}(\boldsymbol{r}, \hat{\boldsymbol{\beta}}^{\boldsymbol{k}})$. See Berndt et al. (1974), Fletcher and Powell (1963), and Kennedy and Gentle (1980) for some choices of α_{k+1}.

Gauss-Newton Method

The Gauss-Newton method is designed specifically for NLS by replacing the regression function $h(\boldsymbol{x_i}, \hat{\boldsymbol{\beta}})$ in (23.17) by its first-order Taylor expansion. The resulting iteration step is

$$\hat{\boldsymbol{\beta}}^{\boldsymbol{k+1}} = \hat{\boldsymbol{\beta}}^{\boldsymbol{k}} - \left\{\boldsymbol{J}(\boldsymbol{h}, \hat{\boldsymbol{\beta}}^{\boldsymbol{k}})^\top \boldsymbol{J}(\boldsymbol{h}, \hat{\boldsymbol{\beta}}^{\boldsymbol{k}})\right\}^{-1} \boldsymbol{J}(\boldsymbol{h}, \hat{\boldsymbol{\beta}}^{\boldsymbol{k}})^\top \boldsymbol{r}(\hat{\boldsymbol{\beta}}^{\boldsymbol{k}}). \tag{23.20}$$

Being rather similar to Newton's method, it does not require the Hessian matrix $\boldsymbol{H}(\boldsymbol{r}^2, \hat{\boldsymbol{\beta}}^{\boldsymbol{k}})$, which is "approximated" by $\boldsymbol{J}(\boldsymbol{h}, \hat{\boldsymbol{\beta}}^{\boldsymbol{k}})^\top \boldsymbol{J}(\boldsymbol{h}, \hat{\boldsymbol{\beta}}^{\boldsymbol{k}})$ (both matrices are equal in probability for $n \to \infty$ under assumptions of Theorem 5, see Amemiya 1985). Because it only approximates the true Hessian matrix, this method belongs to the class of quasi-Newton methods. The issues discussed in the case of Newton's method apply also to the Gauss-Newton method.

Levenberg-Marquardt Method

Depending on data and the current approximation $\hat{\boldsymbol{\beta}}^{\boldsymbol{k}}$ of $\hat{\boldsymbol{\beta}}^{\boldsymbol{NLS}}$, the Hessian matrix $\boldsymbol{H}(\hat{\boldsymbol{\beta}}^{\boldsymbol{k}})$ or its approximations such as $\boldsymbol{J}(\boldsymbol{h}, \hat{\boldsymbol{\beta}}^{\boldsymbol{k}})^\top \boldsymbol{J}(\boldsymbol{h}, \hat{\boldsymbol{\beta}}^{\boldsymbol{k}})$ can be badly conditioned

or not positive definite, which could even result in divergence of Newton's method (or a very slow convergence in the case of modified Newton's method). The Levenberg-Marquardt method addresses the ill-conditioning by choosing the search direction $\boldsymbol{d}_k = \hat{\boldsymbol{\beta}}^{k+1} - \hat{\boldsymbol{\beta}}^k$ as a solution of

$$\left\{\boldsymbol{J}(\boldsymbol{h}, \hat{\boldsymbol{\beta}}^k)^\top \boldsymbol{J}(\boldsymbol{h}, \hat{\boldsymbol{\beta}}^k) + \tau \boldsymbol{I}_p\right\} \boldsymbol{d}_k = -\boldsymbol{J}(\boldsymbol{h}, \hat{\boldsymbol{\beta}}^k)^\top \boldsymbol{r}(\hat{\boldsymbol{\beta}}^k) \tag{23.21}$$

(see Marquardt 1963). This approach is an analogy of RR used in linear regression (Sect. 23.1.6). Similarly to RR, the Levenberg-Marquardt method improves conditioning of the Hessian matrix and it limits the length of the innovation vector $\boldsymbol{d}_k$ compared to the (Gauss-)Newton method. See Kennedy and Gentle (1980) and Bjorck (1996) for a detailed discussion of this algorithm. There are also algorithms combining both Newton's and the Levenberg-Marquardt approaches by using at each step the method that generates a larger reduction in objective function.

Although Newton's method and its modifications are most frequently used in applications, the fact that they find local minima gives rise to various improvements and alternative methods. They range from simple starting the minimization algorithm from several (randomly chosen) initial points to general global-search optimization methods such as genetic algorithms mentioned in Sect. 23.1.3 and discussed in more details in Chaps. II.5 and II.6.

23.2.2 Statistical Inference

Similarly to linear modeling, the inference in nonlinear regression models is mainly based, besides the estimate $\hat{\boldsymbol{\beta}}^{NLS}$ itself, on two quantities: the residual sum of squares $RSS = \boldsymbol{r}(\hat{\boldsymbol{\beta}}^{NLS})^\top \boldsymbol{r}(\hat{\boldsymbol{\beta}}^{NLS})$ and the (asymptotic) variance of the estimate $Var(\hat{\boldsymbol{\beta}}^{NLS}) = \sigma^2 \boldsymbol{A}(\boldsymbol{\beta}_0)^{-1}$, see (23.18). Here we discuss how to compute these quantities for $\hat{\boldsymbol{\beta}}^{NLS}$ and its functions.

RSS will be typically a by-product of a numerical computation procedure, since it constitutes the minimized function. RSS also provides an estimate of σ^2: $s^2 = RSS/(n-k)$. The same also holds for the matrix $\boldsymbol{A}(\boldsymbol{\beta}_0)$, which can be consistently estimated by $\boldsymbol{A}(\hat{\boldsymbol{\beta}}^{NLS}) = \boldsymbol{J}(\boldsymbol{h}, \hat{\boldsymbol{\beta}}^k)^\top \boldsymbol{J}(\boldsymbol{h}, \hat{\boldsymbol{\beta}}^k)$, that is, by the asymptotic representation of the Hessian matrix $\boldsymbol{H}(\boldsymbol{r}^2, \hat{\boldsymbol{\beta}}^k)$. This matrix or its approximations are computed at every step of (quasi-)Newton methods for NLS, and thus, it will be readily available after the estimation.

Furthermore, the inference in nonlinear regression models may often involve a nonlinear (vector) function of the estimate $\boldsymbol{f}(\hat{\boldsymbol{\beta}}^{NLS})$; for example, when we test a hypothesis (see Amemiya 1983, for a discussion of NLS hypothesis testing). Contrary to linear functions of estimates, where $Var(\boldsymbol{A}\hat{\boldsymbol{\beta}}^{NLS} + \boldsymbol{a}) = \boldsymbol{A}^\top Var(\hat{\boldsymbol{\beta}}^{NLS})\boldsymbol{A}$, there is no exact expression for $Var[\boldsymbol{f}(\hat{\boldsymbol{\beta}}^{NLS})]$ in a general case. Thus, we usually assume the first-order differentiability of $\boldsymbol{f}(\cdot)$ and use the Taylor expansion to approximate this variance. Since

$$\boldsymbol{f}(\hat{\boldsymbol{\beta}}) = \boldsymbol{f}(\boldsymbol{\beta_0}) + \frac{\partial \boldsymbol{f}(\boldsymbol{\beta_0})}{\partial \boldsymbol{\beta}^\top}(\hat{\boldsymbol{\beta}} - \boldsymbol{\beta_0}) + o(\|\hat{\boldsymbol{\beta}} - \boldsymbol{\beta_0}\|),$$

it follows that the variance can be approximated by

$$Var[\boldsymbol{f}(\hat{\boldsymbol{\beta}}^{NLS})] \doteq \frac{\partial \boldsymbol{f}(\hat{\boldsymbol{\beta}}^{NLS})}{\partial \boldsymbol{\beta}^\top} Var(\hat{\boldsymbol{\beta}}^{NLS}) \frac{\partial \boldsymbol{f}(\hat{\boldsymbol{\beta}}^{NLS})}{\partial \boldsymbol{\beta}}.$$

Hence, having an estimate of $Var(\hat{\boldsymbol{\beta}}^{NLS})$, the Jacobian matrix $\partial f / \partial \beta^\top$ of function $\boldsymbol{f}$ evaluated at $\hat{\boldsymbol{\beta}}^{NLS}$ provides the first-order approximation of the variance of $\boldsymbol{f}(\hat{\boldsymbol{\beta}}^{NLS})$.

23.2.3 Ill-Conditioned Nonlinear System

Similarly to linear modeling, the nonlinear models can also be ill-conditioned when the Hessian matrix $H(\boldsymbol{r}^2, \hat{\boldsymbol{\beta}})$ is nearly singular or does not even have a full rank, see Sect. 23.1.2. This can be caused either by the nonlinear regression function h itself or by too many explanatory variables relative to sample size n. Here we mention extensions of methods dealing with ill-conditioned problems in the case of linear models (discussed in Sects. 23.1.5–23.1.10) to nonlinear modeling: variable selection, ridge regression, Stein-rule estimator, Lasso, and partial least squares.

First, a straightforward tool to eliminate the ill-conditioning of a linear estimation problem is selecting a subset of variables sufficient for predicting the dependent variable as discussed in Sect. 23.1.3. However in nonlinear models, selecting relevant variables is complicated by the need to correctly specify a model first (an incorrect model specification leads to inconsistent estimates and statistics used to select variables). Therefore, the model-free variable selection is particularly useful in nonlinear regression. The model-free variable selection relies on the principle of a sufficient dimension reduction, which in general tries to replace a high-dimensional vector of explanatory variables by a small number of their projections (e.g., similarly to PLS) without assuming a parametric form of the regression function (i.e., the relationships between variables have to be estimated nonparametrically, see Chap. III.5). One way to use such a nonparametric fit is to construct t- or χ^2-statistics for testing significance of each variable (Li et al. 2005) and then perform backward elimination or forward selection in the same way as discussed in Sect. 23.1.3. Another possibility is to extend the nonparametric estimation of the dimension-reduction space by additional constraints analously to shrinkage estimators such as RR and Lasso (Bondell and Li 2009).

Now, considering the shrinkage methods, one of the early nonlinear RR estimators was proposed by Dagenais (1983), who simply added a diagonal matrix to $\boldsymbol{H}(\boldsymbol{r}^2, \boldsymbol{\beta})$ in equation (23.19). Since the nonlinear modeling is done by minimizing

of an objective function, a more straightforward way is to use the alternative formulation (23.11) of RR and to minimize

$$\sum_{i=1}^{n}\{y_i - h(\boldsymbol{x}_i^\top, \boldsymbol{\beta})\}^2 + k\sum_{j=1}^{p}\beta_j^2 = \boldsymbol{r}(\boldsymbol{\beta})^\top \boldsymbol{r}(\boldsymbol{\beta}) + k\|\boldsymbol{\beta}\|_2^2, \tag{23.22}$$

where k represents the ridge coefficient. See Ngo et al. (2003) for an application of this approach. Next, equally straightforward is an application of Stein-rule estimator (23.8) in nonlinear regression, see Kim and Hill (1995) for a recent study of the positive-part Stein-rule estimator within the Box-Cox model. The same could possibly apply to Lasso-type estimators discussed in Sect. 23.1.8 as well: the Euclidian norm $\|\boldsymbol{\beta}\|_2^2$ in (23.22) would just have to be replaced by another L_q norm. Nevertheless, the only application of Lasso in nonlinear models concerns situations, where a transformation to a model with a linear structure is possible; for example, in generalized linear models (Park and Hastie 2007) or in models approximating a regression function by a linear combination of basis functions (Tateishi et al. 2010).

Finally, there is a range of modifications of PLS designed for nonlinear regression modeling, which either try to make the relationship between dependent and explanatory variables linear in unknown parameters or deploy an intrinsically nonlinear model. First, the methods using linearization are typically based on approximating a nonlinear relationship by higher-order polynomials (see quadratic PLS by Wold et al. 1989, and INLR approach by Berglund and Wold 1997) or a piecewise constant approximation (GIFI approach, see Berglund et al. 2001). Wold et al. (2001) present an overview of these methods. Second, several recent works introduced intrinsic nonlinearity into PLS modeling. Among most important contributions, there are Qin and McAvoy (1992) and Malthouse et al. (1997) modeling the nonlinear relationship using a forward-feed neural network, Wold (1992) and Durand and Sabatier (1997) transforming predictors by spline functions, and Bang et al. (2003) using fuzzy-clustering regression approach.

References

Akaike, H.: A new look at the statistical model identification. IEEE Trans. Automat. Contr. **19**, 716–723 (1974)

Akdeniz, F., Yüksel, G., Wan, A.T.K.: The moments of the operational almost unbiased ridge regression estimator. Appl. Math. Comput. **153**, 673–684 (2004)

Amemiya, T.: Non-linear regression models. In: Griliches, Z., Intriligator, M.D. (eds.) Handbook of Econometrics, vol. 1, North-Holland Publishing Company, Amsterdam (1983)

Amemiya, T.: Advanced Econometrics. Harvard University Press, Cambridge, USA (1985)

Austin, P.C.: Bootstrap model selection had similar performance for selecting authentic and noise variables compared to backward variable elimination: A simulation study. J. Clin. Epidemiol. **61**, 1009–1017 (2008)

Bang, Y.H., Yoo, C.K., Lee, I.-B.: Nonlinear PLS modeling with fuzzy inference system. Chemometr. Intell. Lab. Syst. **64**, 137–155 (2003)

Barlow, J.L.: Numerical aspects of solving linear least squares problems. In: Rao, C.R. (eds.) Handbook of Statistics, Volume 9. Elsevier, Amsterdam, London, New York, Tokyo (1993)
Barros, A.S., Rutledge, D.N.: Genetic algorithm applied to the selection of principal components. Chemometr. Intell. Lab. Syst. **40**, 65–81 (1998)
Bates, D.M., Watts, D.G.: Nonlinear Regression Analysis and Its Applications. Wiley, New York, USA (1988)
Bedrick, E.J., Tsai, C.-L.: Model selection for multivariate regression in small samples. Biometrics **50**, 226–231 (1994)
Berglund, A., Wold, S.: INLR, implicit nonlinear latent variable regression. J. Chemometr. **11**, 141–156 (1997)
Berglund, A., Kettaneh, W.S., Bendwell, N., Cameron, D.R.: The GIFI approach to non-linear PLS modelling. J. Chemometr. **15**, 321–336 (2001)
Berndt, E.R., Hall, B.H., Hall, R.E., Hausman, J.A.: Estimation and inference in nonlinear structural models. Ann. Econometr. Soc. Meas. **3**, 653–666 (1974)
Björck, A.: Numerical Methods for Least Squares Problems. SIAM Press, Philadelphia, USA (1996)
Björkström, A., Sundberg, R.: Continuum regression is not always continuous. J. Roy. Stat. Soc. B **58**, 703–710 (1996)
Björkström, A., Sundberg, R.: A generalized view on continuum regression. Scand. J. Stat. **26**, 17–30 (1999)
Bondell, H.D., Li, L.: Shrinkage inverse regression estimation for model-free variable selection. J. Roy. Stat. Soc. B **71**, 287–299 (2009)
Brooks, R., Stone, M.: Continuum regression: cross-validated sequentially constructed prediction embracing ordinary least squares, partial least squares and principal component regression. J. Roy. Stat. Soc. B **52**, 237–269 (1990)
Brooks, R., Stone, M.: Joint continuum regression for multiple predicants. J. Am. Stat. Assoc. **89**, 1374–1377 (1994)
Butler, N.A., Denham, M.C.: The peculiar shrinkage properties of partial least squares regression. J. Roy. Stat. Soc. B **62**, 585–593 (2000)
Chambers, L.: Practical Handbook of Genetic Algorithms: Complex Coding Systems, vol. III, CRC Press, USA (1998)
Chawla, J.S.: A note on ridge regression. Stat. Probab. Lett. **9**, 343–345 (1990)
Chong, I.-G., Jun, C.-H.: Performance of some variable selection methods when multicollinearity is present. Chemometr. Intell. Lab. Syst. **78**, 103–112 (2005)
Coutis, C.: Partial least squares algorithm yields shrinkage estimators. Ann. Stat. **24**, 816–824 (1996)
Dagenais, M.G.: Extension of the ridge regression technique to non-linear models with additive errors. Econ. Lett. **12**, 169–174 (1983)
Danilov, D., Magnus, J.R.: On the harm that ignoring pretesting can cause. J. Econometr. **122**, 27–46 (2004)
Denham, M.C.: Prediction intervals in partial least squares. J. Chemometr. **11**, 39–52 (1997)
Depczynski, U., Frost, V.J., Molt, K.: Genetic algorithms applied to the selection of factors in principal component regression. Anal. Chim. Acta **420**, 217–227 (2000)
Durand, J.-F., Sabatier, R.: Additive spline for partial least squares regression. J. Am. Stat. Assoc. **92**, 1546–1554 (1997)
Edwards, D., Havranek, T.: A fast model selection procedure for large families of models. J. Am. Stat. Assoc. **82**, 205–213 (1987)
Efron, B., Hastie, T., Johnstone, I., Tibshirani. R.: Least angle regression. Ann. Stat. **32**, 407–499 (2004)
Efroymson, M.A.: Multiple regression analysis. In: Ralston, A., Wilf, H.S. (eds.) Mathematical Methods for Digital Computers, vol. 1, Wiley, New York, USA (1960)
Faber, N.M., Song, X.-H., Hopke, P.K.: Sample specific standard error of prediction for partial least squares regression. Trends Analyt. Chem. **22**, 330–334 (2003)

Fan, J., Li, R.: Variable selection via nonconcave penalized likelihood and its oracle properties. J. Am. Stat. Assoc. **96**, 1348–1360 (2001)
Farebrother, R.W.: Further results on the mean square error of ridge estimation. J. Roy. Stat. Soc. B **38**, 248–250 (1976)
Fletcher, R., Powell, M.J.D.: A rapidly convergent descent method for minimization. Comput. J. **6**, 163–168 (1963)
Frank, I.E., Friedman, J.H., Wold, S., Hastie, T., Mallows, C.: A statistical view of some chemometrics regression tools. Technometrics **35**(2), 109–148 (1993)
Garthwaite, P.H.: An interpretation of partial least squares. The J. Am. Stat. Assoc. **89**, 122–127 (1994)
Gentle, J.E.: Numerical Linear Algebra for Applications in Statistics. Springer, New York, USA (1998)
Gruber, M.H.J.: Improving efficiency by shrinkage: the James-Stein and ridge regression estimators. Marcel Dekker, Incorporation, New York, USA (1998)
Gunst, R.F., Mason, R.L.: Regression Analysis and Its Application: A Data-Oriented Approach. Marcel Dekker, Incorporation, New York, USA (1980)
Härdle, W.: Applied Nonparametric Regression. Cambridge University Press, Cambridge, UK (1992)
Härdle, W., Simar, L.: Applied Multivariate Statistical Analysis. Springer, Heidelberg, Germany (2003)
Hawkins, D.M., Yin, X.: A faster algorithm for ridge regression of reduced rank data. Comput. Stat. Data Anal. **40**, 253–262 (2002)
Heij, C., Groenen, P.J.F., van Dijk, D.: Forecast comparison of principal component regression and principal covariate regression. Comput. Stat. Data Anal. **51**, 3612–3625 (2007)
Helland, I.S.: Some theoretical aspects of partial least squares regression. Chemometr. Intell. Lab. Syst. **58**, 97–107 (2001)
Helland, I.S., Almoy, T.: Comparison of prediction methods when only a few components are relevant. J. Am. Stat. Assoc. **89**, 583–591 (1994)
Hocking, R.R.: Methods and Applications of Linear Models: Regression and the Analysis of Variance, 2nd edn. Wiley, New York, USA (1996)
Hoerl, A.E., Kennard, R.W.: Ridge regression: biased estimation of nonorthogonal problems. Technometrics **12**, 55–67 (1970)
Hoerl, A.E., Kennard, R.W., Baldwin, K.F.: Ridge regression: some simulations. Comm. Stat. **4**, 105–123 (1975)
Hughes, A.W., Maxwell, L.K.: Model selection using AIC in the presence of one-sided information. J. Stat. Plann. Infer. **115**, 379–411 (2003)
Hwang, J.T.G., Nettleton, D.: Principal components regression with data-chosen components and related methods. Technometrics **45**, 70–79 (2003)
Ibrahim, J.G., Ming-Hui, C.: Predictive variable selection for the multivariate linear model. Biometrics **53**, 465–478 (1997)
Jian, W., Liu, X.: Consistent model selection based on parameter estimates. J. Stat. Plann. Infer. **121**, 265–283 (2004)
Jollife, I.T.: A note on the use of the principle components in regression. Appl. Stat. **31**(3), 300–303 (1982)
Jong, S.: SIMPLS: An alternative approach to partial least squares regression. Chemometr. Intell. Lab. Syst. **18**, 251–263 (1993)
Jong, S.: PLS shrinks. J. Chemometr. **9**, 323–326 (1995)
Judge, G.G., Bock, M.E.: Biased estimation. In: Griliches, Z., Intriligator, M.D. (eds.) Handbook of Econometrics. vol. 1, North-Holland Publishing Company, Amsterdam (1983)
Kadiyala, K.: A class of almost unbiased and efficient estimators of regression coefficients. Econ. Lett. **16**, 293–296 (1984)
Kennedy, W.J., Gentle, J.E.: Stat. Comput. Marcel Dekker, Incorporation, New York, USA (1980)
Kibria, G.: On preliminary test ridge regression estimators for linear restrictions in a regression model with non-normal disturbances. Comm. Stat. Theor. Meth. **25**, 2349–2369 (1996)

Kim, M., Hill, R.C.: Shrinkage estimation in nonlinear regression: the Box-Cox transformation. J. Econometr. **66**, l–33 (1995)
Knight, K., Fu, W.: Asymptotics for Lasso-type estimators. Ann. Stat. **28**, 1356–1389 (2000)
Leardi, R., Gonzáles, A.L.: Genetic algorithms applied to feature selection in PLS regression: how and when to use them. Chemometr. Intell. Lab. Syst. **41**, 195–207 (1998)
Leamer, E.E.: Model choice and specification analysis. In: Griliches, Z., Intriligator, M.D. (eds.) Handbook of Econometrics. vol. 1, North-Holland Publishing Company, Amsterdam (1983)
Li, K.-C.: Asymptotic optimality for C_p, C_L, cross-validation and generalized cross-validation: discrete index set. Ann. Stat. **15**, 958–975 (1987)
Li, B., Morris, J., Martin, E.B.: Model section for partial least squares regression. Chemometr. Intell. Lab. Syst. **64**, 79–89 (2002)
Li, L., Cook, R.D., Nachtsheim, C.J.: Model-free variable selection. J. Roy. Stat. Soc. B **67**, 285–299 (2005)
Lipovetski, S.: Enhanced ridge regression. Math. Comput. Model. **51**, 338–348 (2010)
Magnus, J.R.: The traditional pretest estimator. Theor. Probab. Appl. **44**(2), 293–308 (1999)
Magnus, J.R.: Estimation of the mean of a univariate normal distribution with known variance. Econometr. J. **5**, 225–236 (2002)
Magnus, J.R., Powell, O., Prüfer, P.: A comparison of two model averaging techniques with an application to growth empirics. J. Econometr. **154**, 139–153 (2010)
Malthouse, E.C., Tamhane, A.C., Mah, R.S.H.: Nonlinear partial least squares. Comput. Chem. Eng. **21**(8), 875–890 (1997)
Marquardt, D.W.: An algorithm for least-squares estimation of nonlinear parameters. J. Soc. Ind. Appl. Math. **11**, 431–441 (1963)
McDonald, G.C., Schwing, R.C.: Instabilities of regression estimates relating air pollution to mortality. Technometrics **15**, 463–482 (1973)
Miller, A.J.: Selection of subsets of regression variables. J. Roy. Stat. Soc. A **147**(3), 389–425 (1984)
Miller, A.: Subset Selection in Regression, Chapman & Hall/CRC, USA (2002)
Montgomery, D.C., Peck, E.A., Vining, G.G.: Introduction to Linear Regression Analysis, 3rd edn. Wiley, New York, USA (2001)
Ngo, S.H., Kemény, S., Deák, A.: Performance of the ridge regression methods as applied to complex linear and nonlinear models. Chemometr. Intell. Lab. Syst. **67**, 69-78 (2003)
Ohtani, K.: On small sample properties of the almost unbiased generalized ridge estimator. Comm. Stat. Theor. Meth. **22**, 2733–2746 (1986)
Osborne, M.R., Presnell, B., Turlach. B.A.: On the Lasso and its dual. J. Comput. Graph. Stat. **9**, 319–337 (1999)
Osten, D.W.: Selection of optimal regression models via cross-validation. J. Chemometr. **2**, 39–48 (1988)
Park, M.-Y., Hastie, T.: An L1 regularization-path algorithm for generalized linear models. J. Roy. Stat. Soc. B **69**, 659–677 (2007)
Phatak, A., Reilly, P.M., Pendilis, A.: The asymptotic variance of the univariate PLS estimator. Lin. Algera Appl. **354**, 245–253 (2002)
Pötscher, B.M., Leeb, H.: On the distribution of penalized maximum likelihood estimators: the LASSO, SCAD, and thresholding. J. Multivariate Anal. **100**, 2065–2082 (2009)
Rao, C.R., Toutenberg, H.: Linear Models, Springer, New York, USA (1999)
Rao, C.R., Wu, Y.: A strongly consistent procedure for model selection in a regression problem. Biometrika **76**, 369–374 (1989)
Qin, S., McAvoy, T.: Nonlinear PLS modeling using neural networks. Comput. Chem. Eng. **16**, 379–391 (1992)
Qin, S.J.: Recursive PLS algorithms for adaptive data modeling. Comput. Chem. Eng. **22**(4), 503–514 (1997)
Schwarz, G.: Estimating the dimension of a model. Ann. Stat. **6**, 461–464 (1978)
Seber, G.A.F., Wild, C.J.: Nonlinear Regression, Wiley, New York, USA (2003)
Shao, J.: Linear model selection by cross-validation. J. Am. Stat. Assoc. **88**, 486–494 (1993)

Shao, J.: An asymptotic theory for linear model selection. Stat. Sin. **7**, 221–264 (1997)
Shen, X., Huang, H.-C., Ye, J.: Inference after model selection. J. Am. Stat. Assoc. **99**, 751–762 (2004)
Shen, X., Ye, J.: Adaptive model selection. J. Am. Stat. Assoc. **97**, 210–221 (2002)
Shi, P., Tsai, C.-L.: A note on the unification of the Akaike information criterion. J. Roy. Stat. Soc. B **60**, 551–558 (1998)
Shiaishi, T., Konno, Y.: On construction of improved estimators in multiple-design multivariate linear models under general restrictions. Ann. Inst. Stat. Math. **46**, 665–674 (1995)
Shibata, R.: An optimal selection of regression variables. Biometrika **68**, 45–54 (1981)
Shibata, R.: Approximate efficiency of a selection procedure for the number of regression variables. Biometrika **71**, 43–49 (1984)
Singh, R.K., Pandey, S.K., Srivastava, V.K.: A generalized class of shrinkage estimators in linear regression when disturbances are not normal. Comm. Stat. Theor. Meth. **23**, 2029–2046 (1994)
Spanos, A., McGuirk, A.: The problem of near-multicollinearity revisited: erratic vs systematic volatility. J. Econometr. **108**, 365–393 (2002)
Stoica, P., Söderström, T.: Partial least squares: a first-order analysis. Scand. J. Stat. **25**, 17–26 (1998)
Stone, M.: Cross-validatory choice and assessment of statistical predictions. J. Roy. Stat. Soc. B **36**, 111–147 (1974)
Sundberg, R.: Continuum regression and ridge regression. J. Roy. Stat. Soc. B **55**, 653–659 (1993)
Swamy, P.A.V.B., Mehtam J,S,, Rappoport, P.N.: Two methods of evaluating Hoerl and Kennard's ridge regression. Comm. Stat. A **12**, 1133–1155
Tateishi, S., Matsui, H., Konishi, S.: Nonlinear regression modeling via the lasso-type regularization. J. Stat. Plann. Infer. **140**, 1125–1134 (2010)
Thisted, R.A.: Elements of Statistical Computing. Chapman and Hall, London New York (1988)
Tibshirani, R.: Regression shrinkage and selection via Lasso. J. Roy. Stat. Soc. B **58**, 267–288 (1996)
Tibshirani, R., Saunders, M., Rosset, S., Zhu, J., Knight, K.: Sparsity and smoothness via the fused Lasso. J. Roy. Stat. Soc. B **67**, 91–108 (2005)
Trygg, J., Wold, S.: Orthogonal projections to latent structures, O-PLS. J. Chemometr. **16**(3), 119–128 (2002)
Ullah, A., Sristava, V.K., Chandra, R.: Properties of shrinkage estimators in linear regression when disturbances are not normal. J. Econometr. **21**, 289–402 (1983)
Vinod, H.D., Ullah, A.: Recent Advances in Regression Methods. Marcel Dekker Incorporation, New York, USA (1981)
Wan, A.T.K.: On generalized ridge regression estimators under collinearity and balanced loss. Appl. Math. Comput. **129**, 455–467 (2002)
Wang, H., Li, G., Tsai, C.-L.: Regression coefficient and autoregressive order shrinkage and selection via the lasso. J. Roy. Stat. Soc. B **69**, 63–78 (2007)
Wang, S.G., Chow, S.C.: A note on adaptive generalized ridge regression estimator. Stat. Probab. Lett. **10**, 17–21 (1990)
Wang, S.G., Tse, S.K., Chow, S.C.: On the measures of multicollinearity in least squares regression. Stat. Probab. Lett. **9**, 347–355 (1990)
Wasserman, G.S., Sudjianto, A.: All subsets regression using a generic search algorithm. Comput. Ind. Eng. **27**, 489–492 (1994)
Wegelin, J.A.: A survey of partial least squares (PLS) methods, with emphasis on the two-block case. Technical Report 371, Department of Statistics, University of Washington, Seattle (2000)
Weiss, R.E.: The influence of variable selection: a bayesian diagnostic perspective. J. Am. Stat. Assoc. **90**, 619–625 (1995)
Wentzell, P.D., Montoto, L.V.: Comparison of principal components regression and partial least squares regression through generic simulations of complex mixtures. Chemometr. Intell. Lab. Syst. **65**, 257–279 (2003)
Wold, H.: Estimation of principle components and related models by iterative least squares. In: Krishnaiaah (eds.) Multivariate analysis. Academic Press, New York (1966)

Wold, S.: Cross-validation estimation of the number of components in factor and principal components analysis. Technometrics **24**, 397–405 (1978)
Wold, S., Kettaneh-Wold, N., Skagerberg, B.: Nonlinear PLS modelling. Chemometr. Intell. Lab. Syst. **7**, 53–65 (1989)
Wold, S.: Nonlinear partial least squares modelling II. Spline inner relation. Chemometr. Intell. Lab. Syst. **14**, 71–84 (1992)
Wold S., Trygg J., Berglund A., and Atti H.: Some recent developments in PLS modeling. Chemometrics Intelligent Laboratory System **58**, 131–150 (2001)
Yanagihara, H., Satoh, K.: An unbiased Cp criterion for multivariate ridge regression. J. Multivariate Anal. **101**, 1226–1238 (2010)
Zhang, P.: Inference after variable selection in linear regression models. Biometrika **79**(4), 741–746 (1992)
Zhang, M.H., Xu, Q.S., Massart, D.L.: Averaged and weighted average partial least squares. Anal. Chim. Acta **504**, 279–289 (2004)
Zheng, X., Loh, W.-Y.: Consistent variable selection in linear models. J. Am. Stat. Assoc. **90**, 151–156 (1995)
Zou, H.: The Adaptive Lasso and Its Oracle Properties. J. Am. Stat. Assoc. **101**, 1418–1429 (2006)

Chapter 24
Generalized Linear Models

Marlene Müller

24.1 Introduction

Generalized linear models (GLM) extend the concept of the well understood linear regression model. The linear model assumes that the conditional expectation of Y (the dependent or response variable) is equal to a linear combination $\boldsymbol{X}^\top\boldsymbol{\beta}$, i.e.

$$E(Y|\boldsymbol{X}) = \boldsymbol{X}^\top\boldsymbol{\beta}.$$

This could be equivalently written as $Y = \boldsymbol{X}^\top\boldsymbol{\beta} + \varepsilon$. Unfortunately, the restriction to linearity cannot take into account a variety of practical situations. For example, a continuous distribution of the error ε term implies that the response Y must have a continuous distribution as well. Hence, the linear regression model may fail when dealing with binary Y or with counts.

Example 1. (Bernoulli responses)
Let us illustrate a binary response model (Bernoulli Y) using a sample on credit worthiness. For each individual in the sample we know if the granted loan has defaulted or not. The responses are coded as

$$Y = \begin{cases} 1 & \text{loan defaults,} \\ 0 & \text{otherwise.} \end{cases}$$

The term of interest is how credit worthiness depends on observable individual characteristics $\boldsymbol{X}$ (age, amount and duration of loan, employment, purpose of loan, etc.). Recall that for a Bernoulli variable $P(Y = 1|\boldsymbol{X}) = E(Y|\boldsymbol{X})$ holds. Hence, the default probability $P(Y = 1|\boldsymbol{X})$ equals a regression of Y on $\boldsymbol{X}$. A useful approach

M. Müller (✉)
Beuth University of Applied Sciences, Berlin, Germany
e-mail: marlene.mueller@beuth-hochschule.de

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_24,

is the following *logit* model:

$$P(Y = 1|\boldsymbol{X} = \boldsymbol{x}) = \frac{1}{1 + \exp(-\boldsymbol{x}^\top \boldsymbol{\beta})}.$$

Here the function of interest $E(Y|\boldsymbol{X})$ is linked to a linear function of the explanatory variables by the logistic cumulative distribution function (cdf) $F(u) = 1/(1 + e^{-u}) = e^u/(1 + e^u)$. □

The term *generalized linear models* (GLM) goes back to Nelder and Wedderburn (1972) and McCullagh and Nelder (1989) who show that if the distribution of the dependent variable Y is a member of the exponential family, then the class of models which connects the expectation of Y to a linear combination of the variables $\boldsymbol{X}^\top \boldsymbol{\beta}$ can be treated in a unified way. In the following sections we denote the function which relates $\mu = E(Y|\boldsymbol{X})$ and $\eta = \boldsymbol{X}^\top \boldsymbol{\beta}$ by $\eta = G(\mu)$ or

$$E(Y|\boldsymbol{X}) = G^{-1}(\boldsymbol{X}^\top \boldsymbol{\beta}).$$

This function G is called *link function*. For all considered distributions of Y there exists at least one canonical link function and typically a set of frequently used link functions.

24.2 Model Characteristics

The generalized linear model is determined by two components:

- The distribution of Y.
- The link function.

In order to define the GLM methodology as a specific class of nonlinear models (for a general approach to nonlinear regression see Chap. III.23), we assume that the distribution of Y is a member of the *exponential family*. The exponential family covers a large number of distributions, for example discrete distributions as the Bernoulli, binomial and Poisson which can handle binary and count data or continuous distributions as the normal, Gamma or Inverse Gaussian distribution.

24.2.1 Exponential Family

We say that a distribution is a member of the exponential family if its probability mass function (if Y discrete) or its density function (if Y continuous) has the following form:

$$f(y, \boldsymbol{\theta}, \psi) = \exp\left\{\frac{y\boldsymbol{\theta} - b(\boldsymbol{\theta})}{a(\psi)} + c(y, \psi)\right\}. \tag{24.1}$$

The functions $a(\bullet)$, $b(\bullet)$ and $c(\bullet)$ will vary for different Y distributions. Our parameter of interest is $\boldsymbol{\theta}$, which is also called the *canonical parameter* McCullagh and Nelder (1989). The additional parameter ψ, that is only relevant for some of the distributions, is considered as a nuisance parameter.

Example 2. (Normal distribution)
Suppose Y is normally distributed with $Y \sim N(\mu, \sigma^2)$. The probability density function $f(y) = \exp\left\{-(y-\mu)^2/(2\sigma^2)\right\}/(\sqrt{2\pi}\sigma)$ can be written as in (24.1) by setting $\boldsymbol{\theta} = \mu$ and $\psi = \sigma$ and $a(\psi) = \psi^2$, $b(\boldsymbol{\theta}) = \boldsymbol{\theta}^2/2$, and $c(y, \psi) = -y^2/(2\psi^2) - \log(\sqrt{2\pi}\psi)$. □

Example 3. (Bernoulli distribution)
If Y is Bernoulli distributed its probability mass function is

$$P(Y = y) = \mu^y(1-\mu)^{1-y} = \begin{cases} \mu & \text{if} \quad y = 1, \\ 1-\mu & \text{if} \quad y = 0. \end{cases}$$

This can be transformed into $P(Y = y) = \exp(y\boldsymbol{\theta})/(1 + e^{\boldsymbol{\theta}})$ using the *logit* transformation $\boldsymbol{\theta} = \log\{\mu/(1-\mu)\}$ equivalent to $\mu = e^{\boldsymbol{\theta}}/(1 + e^{\boldsymbol{\theta}})$. Thus we obtain an exponential family with $a(\psi) = 1$, $b(\boldsymbol{\theta}) = -\log(1-\mu) = \log(1 + e^{\boldsymbol{\theta}})$, and $c(y, \psi) = 0$. □

Table 24.1 lists some probability distributions that are typically used for a GLM. For the binomial and negative binomial distribution the additional parameter k is assumed to be known. Note also that the Bernoulli, geometric and exponential distributions are special cases of the binomial, negative binomial and Gamma distributions, respectively.

24.2.2 Link Function

After having specified the distribution of Y, the link function G is the second component to choose for the GLM. Recall the model notation $\eta = \boldsymbol{X}^\top\boldsymbol{\beta} = G(\mu)$. In the case that the canonical parameter $\boldsymbol{\theta}$ equals the linear predictor η, i.e. if

$$\eta = \boldsymbol{\theta},$$

the link function is called the *canonical link* function. For models with a canonical link the estimation algorithm simplifies as we will see in Subsection 24.3.3. Table 24.2 shows in its second column the canonical link functions of the exponential family distributions presented in Table 24.1.

Table 24.1 GLM distributions

	Range of y			Variance terms	
		$f(y)$	$\mu(\boldsymbol{\theta})$	$V(\mu)$	$a(\psi)$
Bernoulli $B(\mu)$	$\{0,1\}$	$\mu^y(1-\mu)^{1-y}$	$\frac{e^{\boldsymbol{\theta}}}{1+e^{\boldsymbol{\theta}}}$	$\mu(1-\mu)$	1
Binomial $B(k,\mu)$	$\{0,\ldots,k\}$	$\binom{k}{y}\mu^y(1-\mu)^{k-y}$	$\frac{ke^{\boldsymbol{\theta}}}{1+e^{\boldsymbol{\theta}}}$	$\mu\left(1-\frac{\mu}{k}\right)$	1
Poisson $P(\mu)$	$\{0,1,2,\ldots\}$	$\frac{\mu^y}{y!}e^{-\mu}$	$\exp(\boldsymbol{\theta})$	μ	1
Geometric $Geo(\mu)$	$\{0,1,2,\ldots\}$	$\left(\frac{\mu}{1+\mu}\right)^y\left(\frac{1}{1+\mu}\right)$	$\frac{e^{\boldsymbol{\theta}}}{1-e^{\boldsymbol{\theta}}}$	$\mu+\mu^2$	1
Negative Binomial $NB(\mu,k)$	$\{0,1,2,\ldots\}$	$\binom{k+y-1}{y}\left(\frac{\mu}{k+\mu}\right)^y\left(\frac{k}{k+\mu}\right)$	$\frac{ke^{\boldsymbol{\theta}}}{1-e^{\boldsymbol{\theta}}}$	$\mu+\frac{\mu^2}{k}$	1
Exponential $Exp(\mu)$	$(0,\infty)$	$\frac{1}{\mu}\exp\left(-\frac{x}{\mu}\right)$	$-1/\boldsymbol{\theta}$	μ^2	1
Gamma $G(\mu,\psi)$	$(0,\infty)$	$\frac{1}{\Gamma(\psi)}\left(\frac{\psi}{\mu}\right)^{\psi}\exp\left(-\frac{\psi y}{\mu}\right)y^{\psi-1}$	$-1/\boldsymbol{\theta}$	μ^2	$\frac{1}{\psi}$
Normal $N(\mu,\psi^2)$	$(-\infty,\infty)$	$\frac{\exp\{-(y-\mu)^2/(2\psi^2)\}}{\sqrt{2\pi}\psi}$	$\boldsymbol{\theta}$	1	ψ^2
Inverse Gaussian $IG(\mu,\psi^2)$	$(0,\infty)$	$\frac{\exp\{-(y-\mu)^2/(2\mu^2y\psi^2)\}}{\sqrt{2\pi y^3}\psi}$	$\frac{1}{\sqrt{-2\boldsymbol{\theta}}}$	μ^3	ψ^2

Example 4. (Canonical link for Bernoulli Y)
For Bernoulli Y we have $\mu = e^{\boldsymbol{\theta}}/(1+e^{\boldsymbol{\theta}})$, hence the canonical link is given by the logit transformation $\eta = \log\{\mu/(1-\mu)\}$. □

What link functions could we choose apart from the canonical? For most of the models exists a number of specific link functions. For Bernoulli Y, for example, any smooth cdf can be used. Typical links are the logistic and standard normal (Gaussian) cdfs which lead to logit and *probit* models, respectively. A further alternative for Bernoulli Y is the complementary log–log link $\eta = \log\{-\log(1-\mu)\}$.

A flexible class of link functions for positive Y observations is the class of power functions. These links are given by the Box-Cox transformation

Table 24.2 Characteristics of GLMs

	Canonical link $\boldsymbol{\theta}(\mu)$	Deviance $D(\boldsymbol{y}, \boldsymbol{\mu})$
Bernoulli $B(\mu)$	$\log\left(\frac{\mu}{1-\mu}\right)$	$2\sum\left[y_i \log\left(\frac{y_i}{\mu_i}\right) + (1-y_i)\log\left(\frac{1-y_i}{1-\mu_i}\right)\right]$
Binomial $B(k, \mu)$	$\log\left(\frac{\mu}{k-\mu}\right)$	$2\sum\left[y_i \log\left(\frac{y_i}{\mu_i}\right) + (k-y_i)\log\left(\frac{k-y_i}{k-\mu_i}\right)\right]$
Poisson $P(\mu)$	$\log(\mu)$	$2\sum\left[y_i \log\left(\frac{y_i}{\mu_i}\right) - (y_i - \mu_i)\right]$
Geometric $Geo(\mu)$	$\log\left(\frac{\mu}{1+\mu}\right)$	$2\sum\left[y_i \log\left(\frac{y_i + y_i\mu_i}{\mu_i + y_i\mu_i}\right) - \log\left(\frac{1+y_i}{1+\mu_i}\right)\right]$
Negative Binomial $NB(\mu, k)$	$\log\left(\frac{\mu}{k+\mu}\right)$	$2\sum\left[y_i \log\left(\frac{y_i k + y_i\mu_i}{\mu_i k + y_i\mu_i}\right) - k\log\left\{\frac{k(k+y_i)}{k(k+\mu_i)}\right\}\right]$
Exponential $Exp(\mu)$	$\frac{1}{\mu}$	$2\sum\left[\frac{y_i - \mu_i}{\mu_i} - \log\left(\frac{y_i}{\mu_i}\right)\right]$
Gamma $G(\mu, \psi)$	$\frac{1}{\mu}$	$2\sum\left[\frac{y_i - \mu_i}{\mu_i} - \log\left(\frac{y_i}{\mu_i}\right)\right]$
Normal $N(\mu, \psi^2)$	μ	$2\sum\left[(y_i - \mu_i)^2\right]$
Inverse Gaussian $IG(\mu, \psi^2)$	$\frac{1}{\mu^2}$	$2\sum\left[\frac{(y_i - \mu_i)^2}{y_i\mu_i^2}\right]$

(Box and Cox 1964), i.e. by $\eta = (\mu^\lambda - 1)/\lambda$ or $\eta = \mu^\lambda$ where we set in both cases $\eta = \log(\mu)$ for $\lambda = 0$.

24.3 Estimation

Recall that the least squares estimator for the ordinary linear regression model is also the maximum-likelihood estimator in the case of normally distributed error terms. By assuming that the distribution of Y belongs to the exponential family it is possible to derive maximum-likelihood estimates for the coefficients of a

GLM. Moreover we will see that even though the estimation needs a numerical approximation, each step of the iteration can be given by a weighted least squares fit. Since the weights are varying during the iteration the likelihood is optimized by an *iteratively reweighted least squares* algorithm.

24.3.1 Properties of the Exponential Family

To derive the details of the maximum-likelihood algorithm we need to discuss some properties of the probability mass or density function $f(\bullet)$. For the sake of brevity we consider f to be a density function in the following derivation. However, the conclusions will hold for a probability mass function as well.

First, we start from the fact that $\int f(y, \boldsymbol{\theta}, \psi)\, dy = 1$. Under suitable regularity conditions (it is possible to exchange differentiation and integration) this implies

$$\begin{aligned} 0 &= \frac{\partial}{\partial \boldsymbol{\theta}} \int f(y, \boldsymbol{\theta}, \psi)\, dy = \int \frac{\partial}{\partial \boldsymbol{\theta}} f(y, \boldsymbol{\theta}, \psi)\, dy \\ &= \int \left\{ \frac{\partial}{\partial \boldsymbol{\theta}} \log f(y, \boldsymbol{\theta}, \psi) \right\} f(y, \boldsymbol{\theta}, \psi)\, dy = E\left\{ \frac{\partial}{\partial \boldsymbol{\theta}} \ell(y, \boldsymbol{\theta}, \psi) \right\}, \end{aligned}$$

where $\ell(y, \boldsymbol{\theta}, \psi) = \log f(y, \boldsymbol{\theta}, \psi)$ denotes the *log-likelihood* function. The function derivative of ℓ with respect to $\boldsymbol{\theta}$ is typically called the *score* function for which it is known that

$$E\left\{ \frac{\partial^2}{\partial \boldsymbol{\theta}^2} \ell(y, \boldsymbol{\theta}, \psi) \right\} = -E\left\{ \frac{\partial}{\partial \boldsymbol{\theta}} \ell(y, \boldsymbol{\theta}, \psi) \right\}^2.$$

This and taking first and second derivatives of (24.1) results in

$$0 = E\left\{ \frac{Y - b'(\boldsymbol{\theta})}{a(\psi)} \right\}, \quad \text{and} \quad E\left\{ \frac{-b''(\boldsymbol{\theta})}{a(\psi)} \right\} = -E\left\{ \frac{Y - b'(\boldsymbol{\theta})}{a(\psi)} \right\}^2,$$

such that we can conclude

$$E(Y) = \mu = b'(\boldsymbol{\theta}), \tag{24.2}$$

$$\mathrm{Var}(Y) = V(\mu) a(\psi) = b''(\boldsymbol{\theta}) a(\psi). \tag{24.3}$$

Note that as a consequence from (24.1) the expectation of Y depends only on the parameter of interest $\boldsymbol{\theta}$. We also assume that the factor $a(\psi)$ is identical over all observations.

24.3.2 Maximum-Likelihood and Deviance Minimization

As pointed out before the estimation method of choice for $\boldsymbol{\beta}$ is maximum-likelihood. As an alternative the literature refers to the minimization of the *deviance*. We will see during the following derivation that both approaches are identical.

Suppose that we have observed a sample of independent pairs $(Y_i, \boldsymbol{X}_i)$ where $i = 1, \ldots, n$. For a more compact notation denote now the vector of all response observations by $\boldsymbol{Y} = (Y_1, \ldots, Y_n)^\top$ and their conditional expectations (given $\boldsymbol{X}_i$) by $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)^\top$. Recall that we study

$$E(Y_i|\boldsymbol{X}_i) = \mu_i = G(\boldsymbol{X}_i^\top \boldsymbol{\beta}) = G(\eta_i).$$

The sample log-likelihood of the vector $\boldsymbol{Y}$ is then given by

$$\ell(\boldsymbol{Y}, \boldsymbol{\mu}, \psi) = \sum_{i=1}^{n} \ell(Y_i, \boldsymbol{\theta}_i, \psi). \tag{24.4}$$

Here $\boldsymbol{\theta}_i$ is a function of $\eta_i = \boldsymbol{X}_i^\top \boldsymbol{\beta}$ and we use $\ell(Y_i, \boldsymbol{\theta}_i, \psi) = \log f(Y_i, \boldsymbol{\theta}_i, \psi)$ to denote the individual log-likelihood contributions for all observations i.

Example 5. (Normal log-likelihood)
For normal responses $Y_i \sim N(\mu_i, \sigma^2)$ we have $\ell(Y_i, \boldsymbol{\theta}_i, \psi) = -(Y_i - \mu_i)^2/(2\sigma^2) - \log\left(\sqrt{2\pi}\sigma\right)$. This gives the sample log-likelihood

$$\ell(\boldsymbol{Y}, \boldsymbol{\mu}, \sigma) = n \log\left(\frac{1}{\sqrt{2\pi}\sigma}\right) - \frac{1}{2\sigma^2} \sum_{i=1}^{n} (Y_i - \mu_i)^2. \tag{24.5}$$

Obviously, maximizing this log-likelihood is equivalent to minimizing the least squares criterion. □

Example 6. (Bernoulli log-likelihood)
The calculation in Example 3 shows that the individual log-likelihoods for the binary responses equal $\ell(Y_i, \boldsymbol{\theta}_i, \psi) = Y_i \log(\mu_i) + (1 - Y_i) \log(1 - \mu_i)$. This leads to

$$\ell(\boldsymbol{Y}, \boldsymbol{\mu}, \psi) = \sum_{i=1}^{n} \{Y_i \log(\mu_i) + (1 - Y_i) \log(1 - \mu_i)\} \tag{24.6}$$

for the sample version. □

The deviance defines an alternative objective function for optimization. Let us first introduce the *scaled deviance* which is defined as

$$D(\boldsymbol{Y}, \boldsymbol{\mu}, \psi) = 2\{\ell(\boldsymbol{Y}, \boldsymbol{\mu}^{\max}, \psi) - \ell(\boldsymbol{Y}, \boldsymbol{\mu}, \psi)\}. \tag{24.7}$$

Here $\boldsymbol{\mu}^{\max}$ (which typically equals $\boldsymbol{Y}$) is the vector that maximizes the saturated model, i.e. the function $\ell(\boldsymbol{Y}, \boldsymbol{\mu}, \psi)$ without imposing any restriction on $\boldsymbol{\mu}$. Since the term $\ell(\boldsymbol{Y}, \boldsymbol{\mu}^{\max}, \psi)$ does not depend on the parameter $\boldsymbol{\beta}$ we see that indeed the minimization of the scaled deviance is equivalent to the maximization of the sample log-likelihood (24.4).

If we now plug-in the exponential family form (24.1) into (24.4) we obtain

$$\ell(\boldsymbol{Y}, \boldsymbol{\mu}, \psi) = \sum_{i=1}^{n} \left\{ \frac{Y_i \boldsymbol{\theta}_i - b(\boldsymbol{\theta}_i)}{a(\psi)} - c(Y_i, \psi) \right\}. \tag{24.8}$$

Obviously, neither $a(\psi)$ nor $c(Y_i, \psi)$ depend on the unknown parameter vector $\boldsymbol{\beta}$. Therefore, it is sufficient to consider

$$\sum_{i=1}^{n} \{Y_i \boldsymbol{\theta}_i - b(\boldsymbol{\theta}_i)\} \tag{24.9}$$

for the maximization. The deviance analog of (24.9) is the (non-scaled) deviance function

$$D(\boldsymbol{Y}, \boldsymbol{\mu}) = D(\boldsymbol{Y}, \boldsymbol{\mu}, \psi)\, a(\psi). \tag{24.10}$$

The (non-scaled) deviance $D(\boldsymbol{Y}, \boldsymbol{\mu})$ can be seen as the GLM equivalent of the *residual sum of squares* (RSS) in linear regression as it compares the log-likelihood ℓ for the "model" μ with the maximal achievable value of ℓ.

24.3.3 Iteratively Reweighted Least Squares Algorithm

We will now minimize the deviance with respect to $\boldsymbol{\beta}$. If we denote the gradient of (24.10) by

$$\nabla(\boldsymbol{\beta}) = \frac{\partial}{\partial \boldsymbol{\beta}} \left[-2 \sum_{i=1}^{n} \{Y_i \boldsymbol{\theta}_i - b(\boldsymbol{\theta}_i)\} \right] = -2 \sum_{i=1}^{n} \left\{ Y_i - b'(\boldsymbol{\theta}_i) \right\} \frac{\partial}{\partial \boldsymbol{\beta}} \boldsymbol{\theta}_i \,, \tag{24.11}$$

our optimization problem consists in solving

$$\nabla(\boldsymbol{\beta}) = 0. \tag{24.12}$$

Note that this is (in general) a nonlinear system of equations in $\boldsymbol{\beta}$ and an iterative solution has to be computed. The smoothness of the link function allows us to compute the *Hessian* of $D(\boldsymbol{Y}, \boldsymbol{\mu})$, which we denote by $\mathcal{H}(\boldsymbol{\beta})$. Now a *Newton–Raphson* algorithm can be applied which determines the optimal $\widehat{\boldsymbol{\beta}}$ using the following iteration steps:

$$\widehat{\boldsymbol{\beta}}^{new} = \widehat{\boldsymbol{\beta}}^{old} - \left\{\mathcal{H}\left(\widehat{\boldsymbol{\beta}}^{old}\right)\right\}^{-1} \nabla\left(\widehat{\boldsymbol{\beta}}^{old}\right).$$

A variant of the Newton–Raphson is the *Fisher scoring* algorithm that replaces the Hessian by its expectation with respect to the observations Y_i:

$$\widehat{\boldsymbol{\beta}}^{new} = \widehat{\boldsymbol{\beta}}^{old} - \left\{E\mathcal{H}\left(\widehat{\boldsymbol{\beta}}^{old}\right)\right\}^{-1} \nabla\left(\widehat{\boldsymbol{\beta}}^{old}\right).$$

To find simpler representations for these iterations, recall that we have $\mu_i = G(\eta_i) = G(\boldsymbol{X}_i^\top \boldsymbol{\beta}) = b'(\boldsymbol{\theta}_i)$. By taking the derivative of the right hand term with respect to $\boldsymbol{\beta}$ this implies

$$b'(\boldsymbol{\theta}_i)\frac{\partial}{\partial \boldsymbol{\beta}}\boldsymbol{\theta}_i = G(\boldsymbol{X}_i^\top \boldsymbol{\beta})\, \boldsymbol{X}_i.$$

Using that $b''(\boldsymbol{\theta}_i) = V(\mu_i)$ as established in (24.3) and taking derivatives again, we finally obtain

$$\begin{aligned}
\frac{\partial}{\partial \boldsymbol{\beta}}\boldsymbol{\theta}_i &= \frac{G'(\eta_i)}{V(\mu_i)}\, \boldsymbol{X}_i \\
\frac{\partial^2}{\partial \boldsymbol{\beta}\boldsymbol{\beta}^\top}\boldsymbol{\theta}_i &= \frac{G''(\eta_i)V(\mu_i) - G'(\eta_i)^2 V'(\mu_i)}{V(\mu_i)^2}\, \boldsymbol{X}_i \boldsymbol{X}_i^\top.
\end{aligned}$$

From this we can express the gradient and the Hessian of the deviance by

$$\begin{aligned}
\nabla(\boldsymbol{\beta}) &= -2\sum_{i=1}^{n} \{Y_i - \mu_i\}\, \frac{G'(\eta_i)}{V(\mu_i)}\, \boldsymbol{X}_i \\
\mathcal{H}(\boldsymbol{\beta}) &= 2\sum_{i=1}^{n} \left\{ \frac{G'(\eta_i)^2}{V(\mu_i)} - \{Y_i - \mu_i\} \frac{G''(\eta_i)V(\mu_i) - G'(\eta_i)^2 V'(\mu_i)}{V(\mu_i)^2} \right\} \boldsymbol{X}_i \boldsymbol{X}_i^\top.
\end{aligned}$$

The expectation of $\mathcal{H}(\boldsymbol{\beta})$ in the Fisher scoring algorithm equals

$$E\mathcal{H}(\boldsymbol{\beta}) = 2\sum_{i=1}^{n} \left\{ \frac{G'(\eta_i)^2}{V(\mu_i)} \right\} \boldsymbol{X}_i \boldsymbol{X}_i^\top.$$

Let us consider only the Fisher scoring algorithm for the moment. We define the weight matrix

$$\mathbf{W} = \operatorname{diag}\left(\frac{G'(\eta_1)^2}{V(\mu_1)}, \ldots, \frac{G'(\eta_n)^2}{V(\mu_n)} \right)$$

and the vectors $\widetilde{\boldsymbol{Y}} = (\widetilde{Y}_1, \ldots, \widetilde{Y}_n)^\top$, $\boldsymbol{Z} = (Z_1, \ldots, Z_n)^\top$ by

$$\widetilde{Y}_i = \frac{Y_i - \mu_i}{G'(\eta_i)}\,, \quad Z_i = \eta_i + \widetilde{Y}_i = \boldsymbol{X}_i^\top \boldsymbol{\beta}^{old} + \frac{Y_i - \mu_i}{G'(\eta_i)}\,.$$

Denote further by $\mathbf{X}$ the design matrix given by the rows $x_i^\top$. Then, the Fisher scoring iteration step for $\boldsymbol{\beta}$ can be rewritten as

$$\boldsymbol{\beta}^{new} = \boldsymbol{\beta}^{old} + (\mathbf{X}^\top \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{W} \widetilde{\boldsymbol{Y}} = (\mathbf{X}^\top \mathbf{W} \mathbf{X})^{-1} \mathbf{X}^\top \mathbf{W} \boldsymbol{Z}\,. \tag{24.13}$$

This immediately shows that each Fisher scoring iteration step is the result of a weighted least squares regression of the *adjusted dependent variables* Z_i on the explanatory variables $\boldsymbol{X}_i$. Since the weights are recalculated in each step we speak of the *iteratively reweighted least squares* (IRLS) algorithm. For the Newton–Raphson algorithm a representation equivalent to (24.13) can be found, only the weight matrix $\mathbf{W}$ differs.

The iteration will be stopped when the parameter estimate and/or the deviance do not change significantly anymore. We denote the final parameter estimate by $\widehat{\boldsymbol{\beta}}$.

24.3.4 Remarks on the Algorithm

Let us first note two special cases for the algorithm:

- In the linear regression model, where we have $G' \equiv 1$ and $\mu_i = \eta_i = \boldsymbol{X}_i^\top \boldsymbol{\beta}$, no iteration is necessary. Here the ordinary least squares estimator gives the explicit solution of (24.12).
- In the case of a canonical link function we have $b'(\boldsymbol{\theta}_i) = G(\boldsymbol{\theta}_i) = G(\eta_i)$ and hence $b''(\boldsymbol{\theta}_i) = G'(\eta_i) = V(\mu_i)$. Therefore the Newton–Raphson and the Fisher scoring algorithms coincide.

There are several further remarks on the algorithm which concern in particular starting values and the computation of relevant terms for the statistical analysis:

- Equation (24.13) implies that in fact we do not need a starting value for $\boldsymbol{\beta}$. Indeed the adjusted dependent variables Z_i can be equivalently initialized by using appropriate values for $\eta_{i,0}$ and $\mu_{i,0}$. Typically, the following initialization is used McCullagh and Nelder (1989):
  - ⋆ For all but binomial models set $\mu_{i,0} = Y_i$ and $\eta_{i,0} = G(\mu_{i,0})$.
  - ⋆ For binomial models set $\mu_{i,0} = (Y_i + \frac{1}{2})/(k+1)$ and $\eta_{i,0} = G(\mu_{i,0})$. (Recall that this holds with $k = 1$ in the Bernoulli case.)

  The latter definition is based on the observation that G can not be applied to binary data. Therefore a kind of smoothing is used to obtain $\mu_{i,0}$ in the binomial case.

- During the iteration the convergence can be controlled by checking the relative change in the coefficients

$$\sqrt{\frac{(\boldsymbol{\beta}^{new} - \boldsymbol{\beta}^{old})^\top (\boldsymbol{\beta}^{new} - \boldsymbol{\beta}^{old})}{\boldsymbol{\beta}^{old\top} \boldsymbol{\beta}^{old}}} < \epsilon$$

and/or the relative change in the deviance

$$\left| \frac{D(\boldsymbol{Y}, \boldsymbol{\mu}^{new}) - D(\boldsymbol{Y}, \boldsymbol{\mu}^{old})}{D(\boldsymbol{Y}, \boldsymbol{\mu}^{old})} \right| < \epsilon \,.$$

- An estimate $\widehat{\psi}$ for the dispersion parameter ψ can be obtained from either the Pearson χ^2 statistic

$$\widehat{a}(\psi) = \frac{1}{n-p} \sum_{i=1}^{n} \frac{(Y_i - \widehat{\mu}_i)^2}{V(\widehat{\mu}_i)}, \tag{24.14}$$

or using deviance

$$\widehat{a}(\psi) = \frac{D(\boldsymbol{Y}, \boldsymbol{\mu})}{n-p} \,. \tag{24.15}$$

Here we use p for the number of estimated parameters and $\widehat{\mu}_i$ for the estimated regression function at the ith observation. Similarly, $\widehat{\boldsymbol{\mu}}$ is the estimated $\boldsymbol{\mu}$. Both estimators for $a(\psi)$ coincide for normal linear regression and follow an exact χ^2_{n-p} distribution then. The number $n-p$ (number of observations minus number of estimated parameters) is denoted as the *degrees of freedom* of the deviance.

- Typically, software for GLM allows for offsets and weights in the model. For details on the inclusion of weights we refer to Subsection 24.5.1. Offsets are deterministic components of η which can vary over the observations i. The model that is then fitted is

$$E(Y_i | \boldsymbol{X}_i) = G(\boldsymbol{X}_i^\top \boldsymbol{\beta} + o_i).$$

Offsets may be used to fit a model where a part of the coefficients is known. The iteration algorithm stays unchanged except for the fact that the optimization is only necessary with respect to the remaining unknown coefficients.

- Since the variance of Y_i will usually depend on $\boldsymbol{X}_i$ we cannot simply analyze residuals of the form $Y_i - \widehat{\mu}_i$. Instead, appropriate transformations have to be used. Classical proposals are Pearson residuals

$$r_i^P = \frac{Y_i - \widehat{\mu}_i}{\sqrt{V(\widehat{\mu}_i)}} \,,$$

deviance residuals

$$r_i^D = \text{sign}(Y_i - \widehat{\mu}_i) \sqrt{d_i} \,,$$

where d_i is the contribution of the ith observation to the deviance, and Anscombe residuals

$$r_i^A = \frac{A(Y_i) - A(\widehat{\mu}_i)}{A'(\widehat{\mu}_i)\sqrt{V(\widehat{\mu}_i)}},$$

where $A(\mu) = \int^{\mu} V^{-1/3}(u)\, du$.

24.3.5 Model Inference

The resulting estimator $\widehat{\boldsymbol{\beta}}$ has an asymptotic normal distribution (except of course for the normal linear regression case when this is an exact normal distribution).

Theorem 1.
Under regularity conditions we have for the estimated coefficient vector

$$\sqrt{n}(\widehat{\boldsymbol{\beta}} - \boldsymbol{\beta}) \to N(0, \Sigma) \quad \text{as } n \to \infty.$$

As a consequence for the scaled deviance and the log-likelihood approximately hold $D(\boldsymbol{Y}, \widehat{\boldsymbol{\mu}}, \psi) \sim \chi^2_{n-p}$ *and* $2\{\ell(\boldsymbol{Y}, \widehat{\boldsymbol{\mu}}, \psi) - \ell(\boldsymbol{Y}, \boldsymbol{\mu}, \psi)\} \sim \chi^2_p$. □

For details on the necessary conditions see for example Fahrmeir and Kaufmann (1984). Note also that the asymptotic covariance ° for the coefficient estimator $\widehat{\boldsymbol{\beta}}$ is the inverse of the Fisher information matrix, i.e.

$$\mathbf{I} = -E\left\{\frac{\partial^2}{\partial \boldsymbol{\beta}\boldsymbol{\beta}^T}\ell(Y, \mu, \psi)\right\}.$$

Since $\mathbf{I}$ can be estimated by the negative Hessian of the log-likelihood or its expectation, this suggests the estimator

$$\widehat{\Sigma} = a(\widehat{\psi})\left[\frac{1}{n}\sum_{i=1}^{n}\left\{\frac{G'(\eta_{i,last})^2}{V(\mu_{i,last})}\right\}\boldsymbol{X}_i\boldsymbol{X}_i^{\top}\right]^{-1}.$$

Using the estimated covariance we are able to test hypotheses about the components of $\boldsymbol{\beta}$.

For model choice between two nested models a likelihood ratio test (LR test) is used. Assume that $\mathcal{M}_0$ (p_0 parameters) is a submodel of the model $\mathcal{M}$ (p parameters) and that we have estimated them as $\widehat{\boldsymbol{\mu}}_0$ and $\widehat{\boldsymbol{\mu}}$. For one-parameter exponential families (without a nuisance parameter ψ) we use that asymptotically

$$D(\boldsymbol{Y}, \boldsymbol{\mu}_0) - D(\boldsymbol{Y}, \boldsymbol{\mu}) \sim \chi^2_{p-p_0}. \tag{24.16}$$

The left hand side of (24.16) is a function of the ratio of the two likelihoods deviance difference equals minus twice the log-likelihood difference. In a two-parameter

exponential family (ψ is to be estimated) one can approximate the likelihood ratio test statistic by

$$\frac{(n-p)\{D(\boldsymbol{Y},\boldsymbol{\mu}_0)-D(\boldsymbol{Y},\boldsymbol{\mu})\}}{(p-p_0)D(\boldsymbol{Y},\boldsymbol{\mu})} \sim F_{p-p_0,n-p} \tag{24.17}$$

using the analog to the normal linear regression case (Venables and Ripley 2002, Chap. 7).

Model selection procedures for possibly non-nested models can be based on Akaike's information criterion Akaike (1973)

$$AIC = D(\boldsymbol{Y},\widehat{\boldsymbol{\mu}},\widehat{\psi}) + 2p,$$

or Schwarz' Bayes information criterion (Schwarz 1978)

$$BIC = D(\boldsymbol{Y},\widehat{\boldsymbol{\mu}},\widehat{\psi}) + \log(n)p,$$

where again p denotes the number of estimated parameters. For a general overview on model selection techniques see also Chap. III.16 of this handbook.

24.4 Practical Aspects

To illustrate the GLM in practice we recall Example 1 on credit worthiness. The credit data set that we use Fahrmeir and Tutz (1994) contains $n = 1{,}000$ observations on consumer credits and a variety of explanatory variables. We have selected a subset of eight explanatory variables for the following examples.

The model for credit worthiness is based on the idea that default can be predicted from the individual and loan characteristics. We consider criteria as age, information on previous loans, savings, employment and house ownership to characterize the credit applicants. Amount and duration of the loan are prominent features of the granted loans. Some descriptive statistics can be found in Table 24.3. We remark that we have categorized the durations (months) into intervals since most of the realizations are multiples of 3 or 6 months.

We are at the first place interested in estimating the probability of credit default in dependence of the explanatory variables $\boldsymbol{X}$. Recall that for binary Y it holds $P(Y = 1|\boldsymbol{X}) = E(Y|\boldsymbol{X})$. Our first approach is a GLM with logit link such that $P(Y = 1|\boldsymbol{X}) = \exp(\boldsymbol{X}^\top\boldsymbol{\beta})/\{1+\exp(\boldsymbol{X}^\top\boldsymbol{\beta})\}$.

Example 7. (Credit default on AGE)
We initially estimate the default probability solely related to age, i.e. the model

$$P(Y = 1|\text{AGE}) = \frac{\exp(\beta_0+\beta_1\text{AGE})}{1+\exp(\beta_0+\beta_1\text{AGE})}$$

or equivalently $\text{logit}\{P(Y = 1|\text{AGE})\} = \beta_0 + \beta_1 \text{AGE}$. The resulting estimates of the constant β_0 and the slope parameter β_1 are displayed in Table 24.4 together with summary statistics on the model fit.

From the table we see that the estimated coefficient of AGE has a negative sign. Since the link function and its inverse are strictly monotone increasing, we can conclude that the probability of default must thus be decreasing with increasing AGE. Figure 24.1 shows on the left frequency barplots of AGE separately for $Y = 1$ and $Y = 0$. From the observed frequencies we can recognize clearly the decreasing propensity to default. The right graph in Fig. 24.1 displays the estimated probabilities $P(Y = 1|\text{AGE})$ using the fitted logit model which are indeed decreasing.

The t-values ($\sqrt{n}\,\widehat{\beta}_j / \sqrt{\widehat{\Sigma}_{jj}}$) show that the coefficient of AGE is significantly different from 0 while the estimated constant is not. The test that is used here is an approximative t-test such that $z_{1-\alpha/2}$-quantile of the standard normal can be used as critical value. This implies that at the usual 5% level we compare the absolute value of the t-value with $z_{0.975} \approx 1.96$.

A more general approach to test for the significance of AGE is to compare the fitted model with a model that involves only a constant default probability. Typically software packages report the deviance of this model as null deviance or similar. In our case we find a null deviance of 1221.7 at 999 degrees of freedom. If we apply

Table 24.3 Credit data

Variable	Yes	No	(in %)	
Y (observed default)	30.0	70.0		
PREVIOUS (no problem)	38.1	61.9		
EMPLOYED ($\geq$ 1 year)	93.8	6.2		
DURATION (9, 12]	21.6	78.4		
DURATION (12, 18]	18.7	81.3		
DURATION (18, 24]	22.4	77.6		
DURATION $\geq$ 24	23.0	77.0		
SAVINGS	18.3	81.7		
PURPOSE (buy a car)	28.4	71.6		
HOUSE (owner)	15.4	84.6		
	Min.	Max.	Mean	Std.Dev.
AMOUNT (in DM)	250	18424	3271.248	2822.752
AGE (in years)	19	75	35.542	11.353

Table 24.4 Credit default on AGE (logit model)

Variable	Coefficient	t-value
Constant	−0.1985	−0.851
AGE	−0.0185	−2.873
Deviance	1213.1	
df	998	
AIC	1217.1	
Iterations	4	

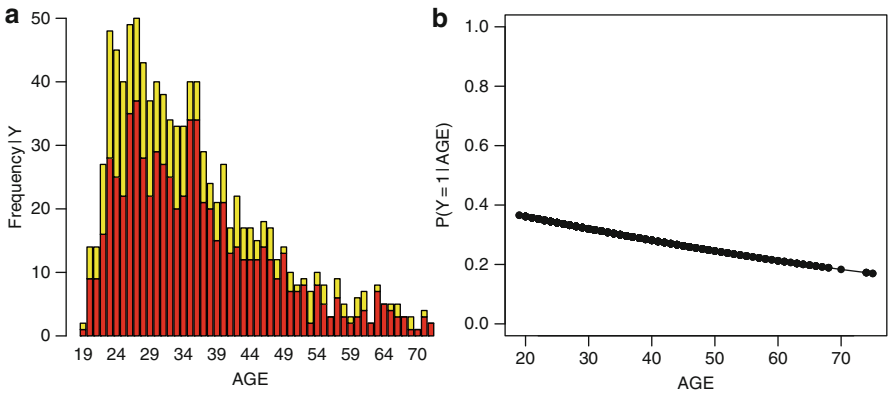


Fig. 24.1 Credit default on AGE, *left*: frequency barplots of AGE for $Y = 1$ (*yellow*) and $Y = 0$ (*red*), *right*: estimated probabilities

the LR test statistic (24.16) to compare the null deviance to the model deviance of 1213.1 at 998 degrees of freedom, we find that constant model is clearly rejected at a significance level of 0.33%. □

Models using different link functions cannot be directly compared as the link functions might be differently scaled. In our binary response model for example a logit or a probit link function may be reasonable. However, the variance parameter of the standard logistic distribution is $\pi^2/3$ whereas that of the standard normal is 1. We therefore need to rescale one of the link functions in order to compare the resulting model fits. Figure 24.2 shows the standard logistic cdf (the inverse logit link) against the cdf of $N(0, \pi^2/3)$. The functions in the left graph of Fig. 24.2 are hardly distinguishable. If we zoom in (right graph) we see that the logistic cdf vanishes to zero at the left boundary at a lower rate. This holds similarly for the right boundary and explains the ability of logit models to (slightly) better handle the case of extremal observations.

Example 8. (Probit versus logit)
If we want to compare the estimated coefficients from a probit to that of the logit model we need to rescale the probit coefficients by $\pi/\sqrt{3}$. Table 24.5 shows the results of a probit for credit default on AGE. The resulting rescaled coefficient for AGE in is of similar size as that for the logit model (cf. Table 24.4) while the constant is not significantly different from 0 in both fits. The deviance and the AIC of the probit fit are slightly larger.

A Newton–Raphson iteration (instead of the Fisher scoring reported in Table 24.5) does give somewhat different coefficients but returns nearly the same value of the deviance (1213.268 for Newton–Raphson versus 1213.265 for Fisher scoring). □

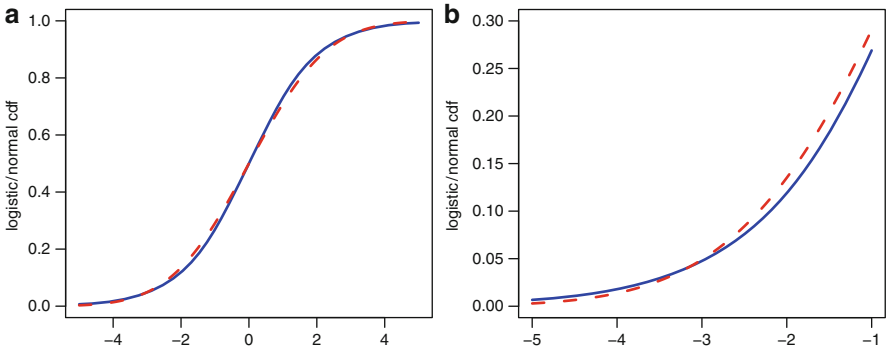


Fig. 24.2 Logit (*solid blue*) versus appropriately rescaled probit link (*dashed red*), *left*: on the range $[-5, 5]$, *right*: on the range of $[-5, -1]$

Table 24.5 Credit default on AGE (probit model), original and rescaled coefficients for comparison with logit

Variable	Coefficient		t-value
	(Original)	(Rescaled)	
Constant	−0.1424	−0.2583	−1.022
AGE	−0.0109	−0.0197	−2.855
Deviance	1213.3		
df	998		
AIC	1217.3		
Iterations	4	(Fisher scoring)	

The next two examples intend to analyze if the fit could be improved by using a nonlinear function on AGE instead of $\eta = \beta_0 + \beta_1 \text{AGE}$. Two principally different approaches are possible:

- Include higher order terms of AGE into η.
- Categorize AGE in order to fit a stepwise constant η function.

Example 9. (Credit default on polynomial AGE)
We fit two logit models using second and third order terms in AGE. The estimated coefficients are presented in Table 24.6. A comparison of the quadratic fit and the linear fit from Example 7 using the LR test statistic (24.16) shows that the linear fit is rejected at a significance level of 3%. A subsequent comparison of the quadratic against the cubic fit no significant improvement by the latter model. Thus, the quadratic term for AGE improves the fit whereas the cubic term does not show any further statistically significant improvement. This result is confirmed when we compare the AIC values of both models which are practically identical. Figure 24.3 shows the estimated default probabilities for the quadratic (left) and cubic AGE fits. We find that the curves are of similar shape. □

Table 24.6 Credit default on polynomial AGE (logit model)

Variable	Coefficient	t-value	Coefficient	t-value
Constant	1.2430	1.799	0.4092	1.909
AGE	−0.0966	−2.699	−0.3240	−1.949
AGE**2	$9.56 \cdot 10^{-4}$	2.234	$6.58 \cdot 10^{-3}$	1.624
AGE**3	–	–	$-4.33 \cdot 10^{-5}$	−1.390
Deviance	1208.3		1206.3	
df	997		996	
AIC	1214.3		1214.3	
Iterations	4		4	

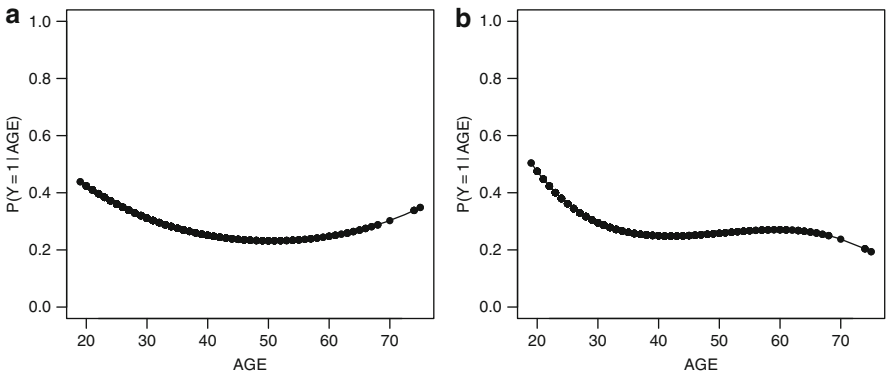


Fig. 24.3 Credit default on polynomial AGE, *left*: estimated probabilities from quadratic function, *right*: estimated probabilities from cubic function

To incorporate a possible nonlinear impact of a variable in the index function, we can alternatively categorize this variable. Another term for this is the construction of *dummy* variables. The most classical form of the categorization consists in using a design matrix that sets a value of 1 in the column corresponding to the category if the category is true and 0 otherwise. To obtain a full rank design matrix we omit one column for the reference category. In our example we leave out the first category which means that all estimated coefficients have to be compared to the zero coefficient of the reference category. Alternative categorization setups are given by omitting the constant, the sum coding (restrict the coefficients to sum up to 0), and the Helmert coding.

Example 10. (Credit default on categorized AGE)
We have chosen the intervals $(18, 23], (23, 28], \ldots, (68, 75]$ as categories. Except for the last interval all of them are of the same length. The first interval $(18, 23]$ is chosen for the reference such that we will estimate coefficients only for the remaining 10 intervals.

Frequency barplots for the intervals and estimated default probabilities are displayed in Fig. 24.4. The resulting coefficients for this model are listed in

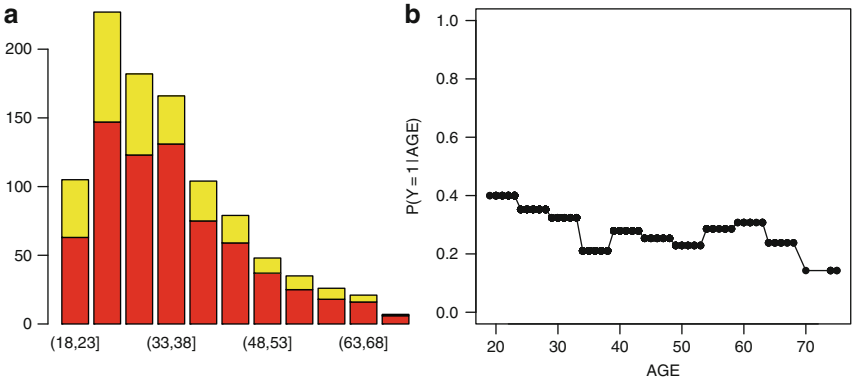


Fig. 24.4 Credit default on categorized AGE, *left*: frequency barplots of categorized AGE for $Y = 1$ (*yellow*) and $Y = 0$ (*red*), *right*: estimated probabilities

Table 24.7 Credit default on categorized AGE (logit model)

Variable	Coefficients	t-values
Constant	−0.4055	−2.036
AGE (23,28]	−0.2029	−0.836
AGE (28,33]	−0.3292	−1.294
AGE (33,38]	−0.9144	−3.320
AGE (38,43]	−0.5447	−1.842
AGE (43,48]	−0.6763	−2.072
AGE (48,53]	−0.8076	−2.035
AGE (53,58]	−0.5108	−1.206
AGE (58,63]	−0.4055	−0.864
AGE (63,68]	−0.7577	−1.379
AGE (68,75]	−1.3863	−1.263
Deviance	1203.2	
df	989	
AIC	1225.2	
Iterations	4	

Table 24.7. We see here that all coefficient estimates are negative. This means, keeping in mind that the group of youngest credit applicants is the reference, that all applicants from other age groups have an (estimated) lower default probability. However, we do not have a true decrease in the default probabilities with AGE since the coefficients do not form a decreasing sequence. In the range from age 33 to 63 we find two local minima and maxima for the estimated default probabilities.

It is interesting to note that the deviance of the categorized AGE fit is the smallest that we obtained up to now. This is explained by the fact that we have fitted the most flexible model here. Unfortunately, this flexibility pays with the number of parameters. The AIC criterion as a compromise between goodness-of-fit and number of parameters states that all previous fitted models are preferable.

Nevertheless, categorization is a valuable tool to explore if there are nonlinear effects. A related technique is local regression smoothing which is shortly reviewed in Subsection 24.5.8. □

The estimation of default probabilities and the prediction of credit default should incorporate more than only one explanatory variable. Before fitting the full model with all available information, we discuss the modeling of interaction effects.

Example 11. (Credit default on AGE and AMOUNT)
The variable AMOUNT is the second continuous explanatory variable in the credit data set. (Recall that duration is quantitative as well but quasi-discrete.) We will therefore use AGE and AMOUNT to illustrate the effects of the simultaneous use of two explanatory variables. A very simple model is of course $\text{logit}\{P(Y = 1|\text{AGE,AMOUNT})\} = \beta_0 + \beta_1\text{AGE} + \beta_2\text{AMOUNT}$. This model, however, separates the impact of AGE and AMOUNT into additive components. The effect of having both characteristics simultaneously is modeled by adding the multiplicative interaction term AGE∗AMOUNT. On the other hand we have seen that at least AGE should be complemented by a quadratic term. For that reason we compare the linear interaction model $\text{logit}\{P(Y = 1|\text{AGE,AMOUNT})\} = \beta_0 + \beta_1\text{AGE} + \beta_2\text{AMOUNT} + \beta_3\text{AGE} * \text{AMOUNT}$ with a specification using quadratic terms and a third model specification using both, quadratic and interaction terms.

Table 24.8 shows the results for all three fitted models. The model with quadratic and interaction terms has the smallest AIC of the three fits. Pairwise LR tests show, however, that the largest of the three models is not significantly better than the model without the interaction term. The obtained surface on AGE and AMOUNT from the quadratic+interaction fit is displayed in Fig. 24.5. □

Let us remark that interaction terms can also be defined for categorical variables. In this case interaction is modeled by including dummy variables for all possible combinations of categories. This may largely increase the number of parameters to estimate.

Table 24.8 Credit default on AGE and AMOUNT (logit model)

Variable	Coefficient	t-value	Coefficient	t-value	Coefficient	t-value
Constant	0.0159	−0.044	1.1815	1.668	1.4864	2.011
AGE	−0.0350	−3.465	−0.1012	−2.768	−0.1083	−2.916
AGE**2	–	–	$9.86 \cdot 10^{-4}$	2.251	$9.32 \cdot 10^{-4}$	2.100
AMOUNT	$-2.80 \cdot 10^{-5}$	−0.365	$-7.29 \cdot 10^{-6}$	−0.098	$-1.18 \cdot 10^{-4}$	−1.118
AMOUNT**2	–	–	$1.05 \cdot 10^{-8}$	1.753	$9.51 \cdot 10^{-9}$	1.594
AGE*AMOUNT	$3.99 \cdot 10^{-6}$	1.951	–	–	$3.37 \cdot 10^{-6}$	1.553
Deviance	1185.1		1180.2		1177.7	
df	996		995		994	
AIC	1193.1		1190.2		1189.7	
Iterations	4		4		4	

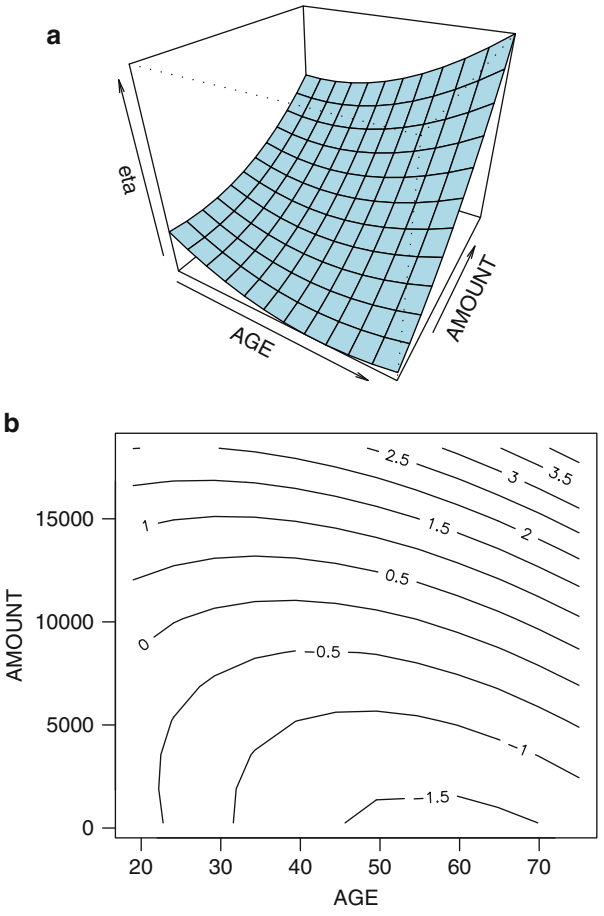


Fig. 24.5 Credit default on AGE and AMOUNT using quadratic and interaction terms, *left*: surface and *right*: contours of the fitted η function

Example 12. (Credit default on the full set of explanatory variables)
In a final analysis we present now the results for the full set of variables from Table 24.3. We first estimated a logit model using all variables (AGE and AMOUNT also with quadratic and interaction terms). Most of the estimated coefficients in the second column of Table 24.9 have the expected sign. For example, the default probability decreases if previous loan were paid back without problems, the credit applicant is employed and has some savings, and the loan is used to buy a car (rather than to invest the loan into goods which cannot serve as a security). A bit surprising is the fact that house owners seem to have higher default probabilities. This might be explained by the fact that house owners usually have additional obligations. The DURATION variable is categorized as described above. Again we have used the first category (loans up to 9 months) as reference. Since the series of DURATION coefficients is monotone increasing,

Table 24.9 Credit default on full set of variables (logit model)

Variable	Coefficient	t-value	Coefficient	t-value
Constant	1.3345	1.592	0.8992	1.161
AGE	−0.0942	−2.359	−0.0942	−2.397
AGE**2	$8.33 \cdot 10^{-4}$	1.741	$9.35 \cdot 10^{-4}$	1.991
AMOUNT	$-2.51 \cdot 10^{-4}$	−1.966	$-1.67 \cdot 10^{-4}$	−1.705
AMOUNT**2	$1.73 \cdot 10^{-8}$	2.370	$1.77 \cdot 10^{-8}$	2.429
AGE*AMOUNT	$2.36 \cdot 10^{-6}$	1.010	–	–
PREVIOUS	−0.7633	−4.652	−0.7775	−4.652
EMPLOYED	−0.3104	−1.015	–	–
DURATION (9, 12]	0.5658	1.978	0.5633	1.976
DURATION (12, 18]	0.8979	3.067	0.9127	3.126
DURATION (18, 24]	0.9812	3.346	0.9673	3.308
DURATION ≥ 24	1.5501	4.768	1.5258	4.710
SAVINGS	−0.9836	−4.402	−0.9778	−4.388
PURPOSE	−0.3629	−2.092	−0.3557	−2.051
HOUSE	0.6603	3.155	0.7014	3.396
Deviance	1091.5		1093.5	
df	985		987	
AIC	1121.5		1119.5	
Iterations	4		4	

we can conclude that longer duration increases the default probability. This is also plausible.

After fitting the full model we have run an automatic stepwise model selection based on AIC. This reveals that the insignificant terms AGE*AMOUNT and EMPLOYED should be omitted. The fitted coefficients for this final model are displayed in the fourth column of Table 24.9. □

24.5 Complements and Extensions

For further reading on GLM we refer to the textbooks of Dobson (2001), McCullagh and Nelder (1989) and Hardin and Hilbe (2001) (the latter with a special focus on STATA). Venables and Ripley (2002, Chap. 7) and Gill (2000) present the topic of generalized linear models in a very compact form. Collett (1991), Agresti (1996), Cox and Snell (1989), and Bishop et al. (1975) are standard references for analyzing categorical responses. We recommend the monographs of Fahrmeir and Tutz (1994) and Lindsey (1997) for a detailed introduction to GLM with a focus on multivariate, longitudinal and spatial data. In the following sections we will shortly review some specific variants and enhancements of the GLM.

24.5.1 Weighted Regression

Prior weights can be incorporated to the generalized linear model by considering the exponential density in the form

$$f(y_i, \boldsymbol{\theta}_i, \psi) = \exp\left[\frac{w_i\{y\boldsymbol{\theta} - b(\boldsymbol{\theta})\}}{a(\psi)} + c(y, \psi, w_i)\right].$$

This requires to optimize the sample log-likelihood

$$\ell(\boldsymbol{Y}, \boldsymbol{\mu}, \psi) = \sum_{i=1}^{n} w_i \left\{\frac{Y_i\boldsymbol{\theta}_i - b(\boldsymbol{\theta}_i)}{a(\psi)} - c(Y_i, \psi, w_i)\right\}$$

or its equivalent, the deviance.

The weights w_i can be 0 or 1 in the simplest case that one wants to exclude specific observations from the estimation. The typical case of applying weights is the case of repeated independent realizations.

24.5.2 Overdispersion

Overdispersion may occur in one-parameter exponential families where the variance is supposed to be a function of the mean. This concerns in particular the binomial or Poisson families where we have $EY = \mu$ and $\mathrm{Var}(Y) = \mu(1 - \mu/k)$ or $\mathrm{Var}(Y) = \mu$, respectively. Overdispersion means that the actually observed variance from the data is larger than the variance imposed by the model. The source for this may be a lack of independence in the data or a misspecification of the model. One possible approach is to use alternative models that allows for a nuisance parameter in the variance, as an example think of the negative binomial instead of the Poisson distribution. For detailed discussions on overdispersion see Collett (1991) and Agresti (1990).

24.5.3 Quasi- or Pseudo-Likelihood

Let us remark that in the case that the distribution of Y itself is unknown but its two first moments can be specified, the quasi-likelihood function may replace the log-likelihood function. This means we still assume that

$$E(Y) = \mu,$$
$$\mathrm{Var}(Y) = a(\psi)\, V(\mu).$$

The quasi-likelihood function is defined through

$$\ell(y, \boldsymbol{\theta}, \psi) = \frac{1}{a(\psi)} \int_{\mu(\boldsymbol{\theta})}^{y} \frac{(s-y)}{V(s)}\, ds\,, \tag{24.18}$$

cf. Nelder and Wedderburn (1972). If Y comes from an exponential family then the derivatives of the log-likelihood and quasi-likelihood function coincide. Thus, (24.18) establishes in fact a generalization of the likelihood approach.

24.5.4 Multinomial Responses

A multinomial model (or nominal logistic regression) is applied if the response for each observation i is one out of more than two alternatives (categories). For identification one of the categories has to be chosen as reference category; without loss of generality we use here the first category. Denote by π_j the probability $P(Y = j|\boldsymbol{X})$, then we can consider the logits with respect to the first category, i.e.

$$\operatorname{logit}(\pi_j) = \log\left(\frac{\pi_j}{\pi_1}\right) = \boldsymbol{X}_j^\top \boldsymbol{\beta}_j\,.$$

The terms $\boldsymbol{X}_j$ and $\boldsymbol{\beta}_j$ indicate that the explanatory variables and their corresponding coefficients may depend on category j. Equivalently we can define the model by

$$P(Y = 1|\boldsymbol{X}) = \frac{1}{1 + \sum_{k=2}^{J} \exp(\boldsymbol{X}_k^\top \boldsymbol{\beta}_k)}$$

$$P(Y = j|\boldsymbol{X}) = \frac{\boldsymbol{X}_j^\top \boldsymbol{\beta}}{1 + \sum_{k=2}^{J} \exp(\boldsymbol{X}_k^\top \boldsymbol{\beta}_k)}\,.$$

It is easy to recognize that the logit model is a special case of the multinomial model for exactly two alternatives.

If the categories are ordered in some natural way then this additional information can be taken into account. A latent variable approach leads to the cumulative logit model or the ordered probit model. We refer here to Dobson (2001, Sect. 8.4) and Greene (2000, Chap. 21) for ordinal logistic regression and ordered probit analysis, respectively.

24.5.5 Contingency Tables

The simplest form of a contingency table

Category	1	2	...	J	$\sum$
Frequency	Y_1	Y_2	...	Y_J	n

with one factor and a predetermined sample size n of observations is appropriately described by a multinomial distribution and can hence be fitted by the multinomial logit model introduced in Subsection 24.5.4. We could be for instance be interested in comparing the trivial model $EY_1 = \ldots = EY_J = \mu$ to the model $EY_2 = \mu_2, \ldots, EY_J = \mu_J$ (again we use the first category as reference). As before further explanatory variables can be included into the model.

Two-way or higher dimensional contingency tables involve a large variety of possible models. Let explain this with the help of the following two-way setup:

Category	1	2	...	J	$\sum$
1	Y_{11}	Y_{12}	...	Y_{1J}	$n_{1\bullet}$
2	Y_{21}	Y_{22}	...	Y_{2J}	$n_{2\bullet}$
$\vdots$	$\vdots$	$\vdots$	$\ddots$	$\vdots$	$\vdots$
K	Y_{K1}	Y_{K2}	...	Y_{KJ}	$n_{K\bullet}$
$\sum$	$n_{\bullet 1}$	$n_{\bullet 2}$	...	$n_{\bullet J}$	n

Here we assume to have two factors, one with realizations $1, \ldots, J$, the other with realizations $1, \ldots, K$. If the Y_{jk} are independent Poisson variables with parameters μ_{jk}, then their sum is a Poisson variable with parameter $E(n) = \mu = \sum \mu_{jk}$. The Poisson assumption implies that the number of observations n is a random variable. Conditional on n, the joint distribution of the Y_{jk} is the multinomial distribution. Without additional explanatory variables, one is typically interested in estimating models of the type

$$\log(EY_{jk}) = \beta_0 + \beta_j + \beta_k$$

in order to compare this with the saturated model $\log(EY_{jk}) = \beta_0 + \beta_j + \beta_k + \beta_{jk}$. If the former model holds then the two factors are independent. Another hypothetical model could be of the form $\log(EY_{jk}) = \beta_0 + \beta_j$ to check whether the second factor matters at all. As in the multinomial case, further explanatory variables can be included. This type of models is consequently termed log-linear model. For more details see for example Dobson (2001, Chap. 9) and McCullagh and Nelder (1989, Chap. 6).

24.5.6 Survival Analysis

Survival data are characterized by non-negative observations which typically have a skewed distribution. An additional complication arises due to the fact that the observation period may end before the individual fails such that censored data may

occur. The exponential distribution with density $f(y, \boldsymbol{\theta}) = \boldsymbol{\theta} e^{-\boldsymbol{\theta} y}$ is a very simple example for a survival distribution. In this special case the survivor function (the probability to survive beyond y) is given by $S(y) = e^{-\boldsymbol{\theta} y}$ and the hazard function (the probability of death within y and $y + dy$ after survival up to y) equals $h(y, \boldsymbol{\theta}) = \boldsymbol{\theta}$. Given additional explanatory variables this function is typically modeled by

$$h(y, \boldsymbol{\theta}) = \exp(\boldsymbol{X}^\top \boldsymbol{\beta}).$$

Extensions of this model are given by using the Weibull distribution leading to non-constant hazards and Cox' proportional hazards model Cox (1972) which uses a semiparametric approach. More material on survival analysis can be found in Chap. III.27.

24.5.7 Clustered Data

Clustered data in relation to regression models mean that data from known groups ("clusters") are observed. Often these are the result of repeated measurements on the same individuals at different time points. For example, imagine the analysis of the effect of a medical treatment on patients or the repeated surveying of households in socio-economic panel studies. Here, all observations on the same individual form a cluster. We speak of longitudinal or panel data in that case. The latter term is typically used in the econometric literature.

When using clustered data we have to take into account that observations from the same cluster are correlated. Using a model designed for independent data may lead to biased results or at least significantly reduce the efficiency of the estimates.

A simple individual model equation could be written as follows:

$$E(Y_{ij} | \boldsymbol{X}_{ij}) = G^{-1}(\boldsymbol{X}_{ij}^\top \boldsymbol{\beta}_j).$$

Here i is used to denote the ith individual observation in the jth cluster. Of course more complex specifications, for example with hierarchical clusters, can be formulated as well.

There is a waste amount of literature which deals with many different possible model specifications. A comprehensive resource for linear and nonlinear mixed effect models (LME, NLME) for continuous responses is Pinheiro and Bates (2000). The term "mixed" here refers to the fact that these models include additional random and/or fixed effect components to allow for correlation within and heterogeneity between the clusters.

For generalized linear mixed models (GLMM), i.e. clustered observations with responses from GLM-type distribution, several approaches are possible. For repeated observations, Liang and Zeger (1986) and Zeger and Liang (1986) propose to use generalized estimating equations (GEE) which result in a quasi-likelihood

estimator. They show that the correlation matrix of $\boldsymbol{Y}_j$, the response observations from one cluster, can be replaced by a "working correlation" as long as the moments of $\boldsymbol{Y}_j$ are correctly specified. Useful working correlations depend on a small number of parameters. For longitudinal data an autoregressive working correlation can be used for example. For more details on GEE see also the monograph by Diggle et al. (2002). In the econometric literature longitudinal or panel data are analyzed with a focus on continuous and binary responses. Standard references for econometric panel data analyses are Hsiao (1990) and Arellano (2003). Models for clustered data with complex hierarchical structure are often denoted as multilevel models. We refer to the monograph of Goldstein (2003) for an overview.

24.5.8 Semiparametric Generalized Linear Models

Nonparametric components can be incorporated into the GLM at different places. For example, it is possible to estimate a single index model

$$E(Y|\boldsymbol{X}) = g(\boldsymbol{X}^\top \boldsymbol{\beta})$$

which differs from the GLM by its unknown smooth link function $g(\bullet)$. The parameter vector $\boldsymbol{\beta}$ in this model can then be only identified up to scale. The estimation of such models has been studied e.g. by Ichimura (1993), Weisberg and Welsh (1994) and Gallant and Nychka (1987).

Local regression in combination with likelihood-based estimation is introduced in Loader (1999). This concerns models of the form

$$E(Y|\boldsymbol{X}) = G^{-1}\{m(\boldsymbol{X})\},$$

where m is an unknown smooth (possibly multidimensional) function. Further examples of semiparametric GLM are generalized additive and generalized partial linear models (GAM, GPLM). These models are able to handle (additional) nonparametric components in the function η. For example, the GAM is specified in this simplest form by

$$E(Y|\boldsymbol{X}) = G^{-1}\left\{\beta_0 + \sum_{j=1}^{p} m_j(X_j)\right\}.$$

Here the m_j denote univariate (or low dimensional) unknown smooth functions which have to be estimated. For their identification is should be assumed, that $Em(X_j) = 0$. The generalized partial linear model combines a linear and a nonparametric function in the function η and is specified as

$$E(Y|X) = G^{-1}\left\{X_1^\top \beta + m(X_2)\right\}.$$

Example 13. (Semiparametric credit model)
We have fitted a generalized partial linear model as a variant of the final model from Example 12. The continuous variables AGE and AMOUNT were used as arguments for the nonparametric component. All other variables of the final model have been included to the linear part of the index function η. Figure 24.6 shows the estimated nonparametric function of AGE and AMOUNT. Although the stepwise model selection in Example 12 indicated that there is no interaction between AGE and AMOUNT, we see now that this interaction could be in fact of some more sophisticated form. The estimation was performed using a generalization of the Speckman (1988) estimator to generalized models. The local kernel weights are calculated from a Quartic (Biweight) kernel function using bandwidths approximately

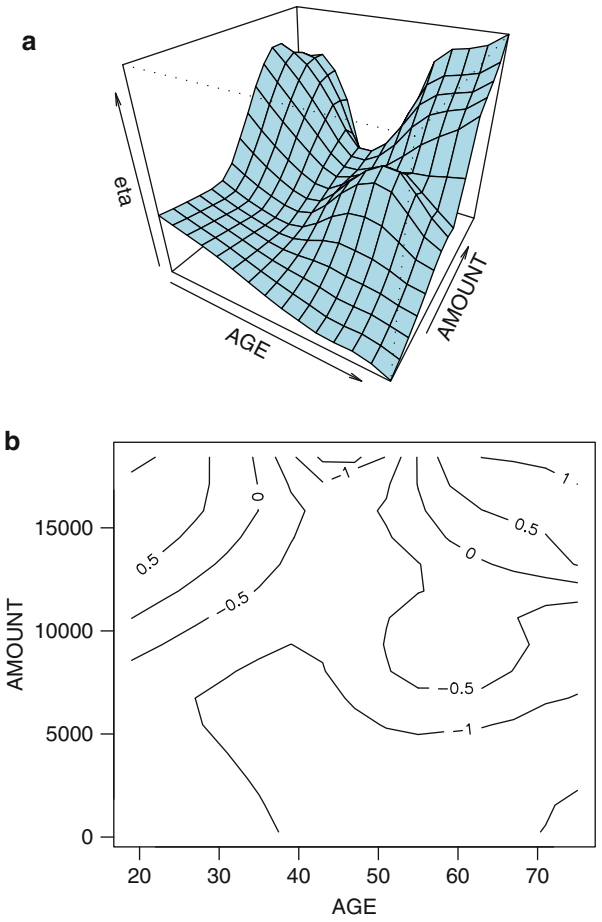


Fig. 24.6 Credit default on AGE and AMOUNT using a nonparametric function, *left*: surface and *right*: contours of the fitted function on AGE and AMOUNT

equal to 33.3% of the ranges of AGE and AMOUNT, respectively. Details on the used kernel based estimation can be found in Severini and Staniswalis (1994) and Müller (2001). □

Some more material on semiparametric regression can be found in Chaps. III.20 and III.21 of this handbook. For a detailed introduction to semiparametric extensions of GLM we refer to the textbooks by Hastie and Tibshirani (1990), Härdle et al. (2004), Ruppert et al. (1990), and Green and Silverman (1994).

References

Agresti, A.: Categorical Data Analysis. Wiley, New York (1990)

Agresti, A.: An Introduction to Categorical Data Analysis. Wiley, New York (1996)

Akaike, H.: Information theory and an extension of the maximum likelihood principle. In: Petrov, B.N., Csàdki, F. (eds.) Second Internationl Symposium on Information Theory, pp. 267–281. Akademiai Kiadó, Budapest (1973)

Arellano, M.: Panel Data Econometrics. Oxford University Press (2003)

Bishop, Y.M.M., Fienberg, S.E., Holland, P.W.: Discrete Multivariate Analysis: Theory and Practice. MIT Press, Cambridge, MA (1975)

Box, G., Cox, D.: An analysis of transformations. J. Roy. Stat. Soc. B **26**, 211–243 (1964)

Collett, D.: Modelling binary data. Chapman and Hall, London (1991)

Cox, D.R.: Regression models and life tables (with discussion). J. Roy. Stat. Soc. B **74**, 187–220 (1972)

Cox, D.R., Snell, E.J.: Analysis of Binary Data, *Monographs on Statistics and Applied Probability*. (2nd ed.), vol. 32, Chapman and Hall, London (1989)

Diggle, P., Heagerty, P., Liang, K.-L., Zeger, S.: Analysis of Longitudinal Date. (2nd ed.), Oxford University Press, Oxford, UK (2002)

Dobson, A.J.: *An Introduction to Generalized Linear Models*. (2nd ed.), Chapman and Hall, London (2001)

Fahrmeir, L., Kaufmann, H.: Consistency and asymptotic normality of the maximum-likelihood estimator in generalized linear models. Ann. Stat. **13**, 342–368 (1984)

Fahrmeir, L., Tutz, G.: Multivariate Statistical Modelling Based on Generalized Linear Models. Springer, New York (1994)

Gallant, A., Nychka, D.: Semi-nonparametric maximum likelihood estimation. Econometrica **55**(2), 363–390 (1987)

Gill, J.: Generalized Linear Models: A Unified Approach. Sage University Paper Series on Quantitative Applications in the Social Sciences, 07-134, Thousand Oaks, CA (2000)

Goldstein, H.: Multilevel Statistical Models. Hodder Arnold, London (2003)

Green, P.J., Silverman, B.W.: Nonparametric Regression and Generalized Linear Models, Monographs on Statistics and Applied Probability. vol. 58, Chapman and Hall, London (1994)

Greene, W.H.: Econometric Analysis. (4th ed.), Prentice Hall, Upper Saddle River, New Jersey (2000)

Hardin, J., Hilbe, J.: Generalized Linear Models and Extensions. Stata Press (2001)

Härdle, W., Müller, M., Sperlich, S., Werwatz, A.: Nonparametric and Semiparametric Modeling: An Introduction. Springer, New York (2004)

Hastie, T.J., Tibshirani, R.J.: Generalized Additive Models, *Monographs on Statistics and Applied Probability*. vol. 43, Chapman and Hall, London (1990)

Hsiao, C.: Analysis of Panel Data. Econometric Society Monographs No. 11. Cambridge University Press, Cambridge, (London/New York) (1990)

Ichimura, H.: Semiparametric least squares (SLS) and weighted SLS estimation of single–index models. J. Econometrics **58**, 71–120 (1993)
Liang, K.Y., Zeger, S.L.: Longitudinal data analysis using generalized linear models. Biometrika **73**, 13–22 (1986)
Lindsey, J.K.: Applying Generalized Linear Models. Springer, New York (1997)
Loader, C.: Local Regression and Likelihood. Springer, New York (1999)
McCullagh, P., Nelder, J.A.: Generalized Linear Models, *Monographs on Statistics and Applied Probability*. (2nd ed.), vol. 37, Chapman and Hall, London, (1989)
Müller, M.: Estimation and testing in generalized partial linear models – a comparative study. Stat. Comput. **11**, 299–309 (2001)
Nelder, J.A., Wedderburn, R.W.M.: Generalized linear models. J. Roy. Stat. Soc. A **135**(3), 370–384 (1972)
Pinheiro, J.C., Bates, D.M.: Mixed-Effects Models in S and S-PLUS. Springer, New York (2000)
Ruppert, D., Wand, M.P., Carroll, R.J.: Semiparametric Regression. Cambridge University Press, Cambridge, (London/New York) (1990)
Schwarz, G.: Estimating the dimension of a model. Ann. Stat. **6**, 461–464 (1978)
Severini, T.A., Staniswalis, J.G.: Quasi-likelihood estimation in semiparametric models. J. Am. Stat. Assoc. **89**, 501–511 (1994)
Speckman, P.E.: Regression analysis for partially linear models. J. Roy. Stat. Soc. B **50**, 413–436 (1988)
Turlach, B.A.: Computer-aided additive modeling. Doctoral Thesis, Université Catholique de Louvain, Belgium (1994)
Venables, W.N., Ripley, B.: *Modern Applied Statistics with S*. (4th ed.), Springer, New York (2002)
Weisberg, S., Welsh, A.H.: Adapting for the missing link. Ann. Stat. **22**, 1674–1700 (1994)
Zeger, S.L., Liang, K.Y.A.: Longitudinal data analysis for discrete and continuous outcomes. Biometrics **42**, 121–130 (1986)

Chapter 25
Robust Statistics

Laurie Davies and Ursula Gather

25.1 Robust Statistics; Examples and Introduction

25.1.1 Two Examples

The first example involves the real data given in Table 25.1 which are the results of an interlaboratory test. The boxplots are shown in Fig. 25.1 where the dotted line denotes the mean of the observations and the solid line the median.

We note that only the results of the Laboratories 1 and 3 lie below the mean whereas all the remaining laboratories return larger values. In the case of the median, 7 of the readings coincide with the median, 24 readings are smaller and 24 are larger. A glance at Fig. 25.1 suggests that in the absence of further information the Laboratories 1 and 3 should be treated as outliers. This is the course which we recommend although the issues involved require careful thought. For the moment we note simply that the median is a robust statistic whereas the mean is not.

The second example concerns quantifying the scatter of real valued observations $x_1, \ldots, x_n$. This example is partially taken from Huber (1981) and reports a dispute between (Eddington 1914, p. 147) and (Fisher 1920, p. 762) about the relative merits of

$$s_n = \left(\frac{1}{n}\sum (x_i - \bar{x})^2\right)^{\frac{1}{2}} \quad \text{and} \quad d_n = \frac{1}{n}\sum |x_i - \bar{x}| \,.$$

L. Davies (✉)
Universität Duisburg-Essen, Essen, Germany
e-mail: laurie.davies@uni-due.de

U. Gather
Technische Universität Dortmund, Dortmund, Germany
e-mail: gather@statistik.tu-dortmund.de

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_25,

Table 25.1 The results of an interlaboratory test involving 14 laboratories

1	2	3	4	5	6	7	9	9	10	11	12	13	14
1.4	5.7	2.64	5.5	5.2	5.5	6.1	5.54	6.0	5.1	5.5	5.9	5.5	5.3
1.5	5.8	2.88	5.4	5.7	5.8	6.3	5.47	5.9	5.1	5.5	5.6	5.4	5.3
1.4	5.8	2.42	5.1	5.9	5.3	6.2	5.48	6.1	5.1	5.5	5.7	5.5	5.4
0.9	5.7	2.62	5.3	5.6	5.3	6.1	5.51	5.9	5.3	5.3	5.6	5.6	

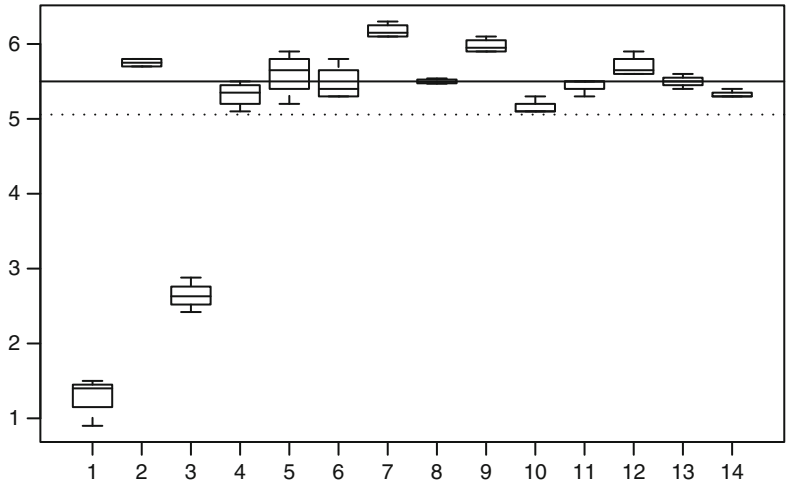


Fig. 25.1 A boxplot of the data of Table 25.1. The *dotted line* and the *solid line* denote respectively the mean and the median of the observations

Fisher argued that for normal observations the standard deviation s_n is about 12% more efficient than the mean absolute deviation d_n. In contrast Eddington claimed that his experience with real data indicates that d_n is better than s_n. In Tukey (1960) and Huber (1977) we find a resolution of this apparent contradiction. Consider the model

$$\mathcal{N}_\epsilon = (1-\epsilon)N\left(\mu, \sigma^2\right) + \epsilon N\left(\mu, 9\sigma^2\right), \tag{25.1}$$

where $N(\mu, \sigma^2)$ denotes a normal distribution with mean μ and variance σ^2 and $0 \leq \epsilon \leq 1$. For data distributed according to (25.1) one can calculate the asymptotic relative efficiency ARE of d_n with respect to s_n,

$$\mathrm{ARE}(\epsilon) = \lim_{n\to\infty} \mathrm{RE}_n(\epsilon) = \lim_{n\to\infty} \frac{\mathrm{Var}(s_n)/E(s_n)^2}{\mathrm{Var}(d_n)/E(d_n)^2}.$$

As Huber states, the result is disquieting. Already for $\epsilon \geq 0.002$ ARE exceeds 1 and the effect is apparent for samples of size 1,000. For $\epsilon = 0.05$ we have $\mathrm{ARE}(\epsilon) = 2.035$ and simulations show that for samples of size 20 the relative efficiency exceeds 1.5 and increases to 2.0 for samples of size 100. This is a severe

deficiency of s_n as models such as $\mathcal{N}_\epsilon$ with ϵ between 0.01 and 0.1 often give better descriptions of real data than the normal distribution itself. We quote Huber (1981)

> thus it becomes painfully clear that the naturally occurring deviations from the idealized model are large enough to render meaningless the traditional asymptotic optimality theory.

25.1.2 General Philosophy

The two examples of the previous section illustrate a general phenomenon. An optimal statistical procedure based on a particular family of models $\mathcal{M}_1$ can differ considerably from an optimal procedure based on another family $\mathcal{M}_2$ even though the families $\mathcal{M}_1$ and $\mathcal{M}_2$ are very close. This may be expressed by saying that optimal procedures are often unstable in that small changes in the data or the model can lead to large changes in the analysis. The basic philosophy of robust statistics is to produce statistical procedures which are stable with respect to small changes in the data or model and even large changes should not cause a complete breakdown of the procedure.

Any inspection of the data and the removal of aberrant observations may be regarded as part of robust statistics but it was only with Pearson (1931) that the consideration of deviations from models commenced. He showed that the exact theory based on the normal distribution for variances is highly nonrobust. There were other isolated papers on the problem of robustness (Bartlett 1935; Pearson 1929; Geary 1936, 1947; Box 1953; Box and Andersen 1955; Gayen 1950). Tukey (1960) initiated a wide spread interest in robust statistics which has continued to this day. The first systematic investigation of robustness is due to Huber (1964) and was expounded in Huber (1981). Huber's approach is functional analytic and he was the first to investigate the behaviour of a statistical functional over a full topological neighbourhood of a model instead of restricting the investigation to other parametric families as in (25.1). Huber considers three problems. The first is that of minimizing the bias over certain neighbourhoods and results in the median as the most robust location functional. For large samples deviations from the model have consequences which are dominated by the bias and so this is an important result. The second problem is concerned with what Tukey calls the statistical version of no free lunches. If we take the simple model of i.i.d. $N(\mu, 1)$ observations then the confidence interval for μ based on the mean is on average shorter than that based on any other statistic. If short confidence intervals are of interest then one can not only choose the statistic which gives the shortest interval but also the model itself. The new model must of course still be consistent with the data but even with this restriction the confidence interval can be made as small as desired (Davies 1995). Such a short confidence interval represents a free lunch and if we do not believe in free lunches then we must look for that model which *maximizes* the length of the confidence interval over a given family of models. If we take all distributions with variance 1 then the confidence interval for the $N(\mu, 1)$ distribution is the longest. Huber considers the same problem over the family $\mathcal{F} = \{F : d_{ko}(F, N(0, 1)) < \epsilon\}$ where d_{ko} denotes the Kolmogoroff metric. Under certain simplifying assumptions

Huber solves this problem and the solution is known as the Huber distribution (see Huber 1981). Huber's third problem is the robustification of the Neyman–Pearson test theory. Given two distributions P_0 and P_1 Neyman and Pearson (1933) derive the optimal test for testing P_0 against P_1. Huber considers full neighbourhoods $\mathcal{P}_0$ of P_0 and $\mathcal{P}_1$ of P_1 and then derives the form of the minimax test for the composite hypothesis of $\mathcal{P}_0$ against $\mathcal{P}_1$. The weakness of Huber's approach is that it does not generalize easily to other situations. Nevertheless it is the spirit of this approach which we adopt here. It involves treating estimators as functionals on the space of distributions, investigating where possible their behaviour over full neighbourhoods and always being aware of the danger of a free lunch.

Hampel (1968) introduced another approach to robustness, that based on the influence function $I(x, T, F)$ defined for a statistical functional T as follows

$$I(x, T, F) = \lim_{\epsilon \to 0} \frac{T((1-\epsilon)F + \epsilon\delta_x) - T(F)}{\epsilon}, \tag{25.2}$$

where δ_x denotes the point mass at the point x. The influence function has two interpretations. On the one hand it measures the infinitesimal influence of an observation situated at the point x on the value of the functional T. On the other hand if $P_n(F)$ denotes the empirical measure of a sample of n i.i.d. random variables with common distribution F then under appropriate regularity conditions

$$\lim_{n \to \infty} \sqrt{n}(T(P_n(F)) - T(F)) \stackrel{D}{=} N\left(0, \int I(x, T, F)^2 \, dF(x)\right), \tag{25.3}$$

where $\stackrel{D}{=}$ denotes equality of distribution. Given a parametric family $\mathcal{P}' = \{P_{\boldsymbol{\theta}} : \boldsymbol{\theta} \in \Theta\}$ of distributions we restrict attention to those functionals which are Fisher consistent that is

$$T(P_{\boldsymbol{\theta}}) = \boldsymbol{\theta}, \quad \boldsymbol{\theta} \in \Theta. \tag{25.4}$$

Hampel's idea was to minimize the asymptotic variance of T as an estimate of a parameter $\boldsymbol{\theta}$ subject to a bound on the influence function

$$\min_T \int I(x, T, P_{\boldsymbol{\theta}})^2 \, dP_{\boldsymbol{\theta}}(x) \quad \text{under (25.4) and } \sup_x |I(x, T, P_{\boldsymbol{\theta}})| \le k(\boldsymbol{\theta}), \tag{25.5}$$

where $k(\boldsymbol{\theta})$ is a given function of $\boldsymbol{\theta}$. Hampel complemented the infinitesimal part of his approach by considering also the global behaviour of the functional T. He introduced the concept of breakdown point which has had and continues to have a major influence on research in robust statistics. The approach based on the influence function was carried out in Hampel et al. (1986). The strength of the Hampel approach is that it can be used to robustify in some sense the estimation of parameters in any parametric model. The weaknesses are that (25.5) only bounds infinitesimally small deviations from the model and that the approach does not explicitly take into account the free lunch problem. Hampel is aware of this and recommends simple models but simplicity is an addition to and not an integral

part of his approach. The influence function is usually used as a heuristic tool and care must be taken in interpreting the results. For examples of situations where the heuristics go wrong we refer to Davies (1993).

Another approach which lies so to speak between that of Huber and Hampel is the so called shrinking neighbourhood approach. It has been worked out in full generality by Rieder (1994). Instead of considering neighbourhoods of a fixed size (Huber) or only infinitesimal neighbourhoods (Hampel) this approach considers full neighbourhoods of a model but whose size decreases at the rate of $n^{-1/2}$ as the sample size n tends to infinity. The size of the neighbourhoods is governed by the fact that for larger neighbourhoods the bias term is dominant whereas models in smaller neighbourhoods cannot be distinguished. The shrinking neighbourhoods approach has the advantage that it does not need any assumptions of symmetry. The disadvantage is that the size of the neighbourhoods goes to zero so that the resulting theory is only robustness over vanishingly small neighbourhoods.

25.1.3 Functional Approach

Although a statistic based on a data sample may be regarded as a function of the data a more general approach is often useful. Given a data set $(x_1, \ldots, x_n)$ we define the corresponding empirical distribution P_n by

$$P_n = \frac{1}{n} \sum_{i=1}^{n} \delta_{x_i} , \tag{25.6}$$

where δ_x denotes the unit mass in x. Although P_n clearly depends on the sample $(x_1, \ldots, x_n)$ we will usually suppress the dependency for the sake of clarity. With this notation we can now regard the arithmetic mean $\bar{x}_n = \sum_{i=1}^{n} x_i / n$ either as a function of the data or as a function T_{av} of the empirical measure P_n,

$$\bar{x}_n = \int x \, dP_n(x) = T_{av}(P_n) .$$

The function T_{av} can be extended to all measures P which have a finite mean

$$T_{av}(P) = \int x \, dP(x), \tag{25.7}$$

and is now a functional defined on a certain subset of the family $\mathcal{P}$ of probability measures on $\mathbb{R}$. This manner of treating statistics is one whose origins go back to von Mises (1937). In the context of robust statistics it was introduced by Huber (1964) and has proved very useful (see Fernholz 1983). Another example is given by the functional T_{sh} defined as the length of the shortest interval which carries a mass of at least $1/2$,

$$T_{sh}(P) = \operatorname{argmin}\{|I| : P(I) \geq 1/2, \, I \subset \mathbb{R}\}, \tag{25.8}$$

where $|I|$ denotes the length of the interval I. The idea of using the shortest half interval goes back to Tukey (see Andrews et al. 1972) who proposed using the mean of the observations contained in it as a robust location functional.

The space $\mathcal{P}$ may be metricized in many ways but we prefer the Kolmogoroff metric d_{ko} defined by

$$d_{ko}(P, Q) = \sup_{x \in \mathbb{R}} |P((-\infty, x]) - Q((-\infty, x])| . \tag{25.9}$$

The Glivenko–Cantelli theorem states

$$\lim_{n \to \infty} d_{ko}(P_n(P), P) = 0, \quad a.s., \tag{25.10}$$

where $P_n(P)$ denotes the empirical measure of the n random variables $X_1(P), \ldots, X_n(P)$ of the i.i.d. sequence $(X_i(P))_1^\infty$. In conjunction with (25.10) the metric d_{ko} makes it possible to connect analytic properties of a functional T and its statistical properties. As a first step we note that a functional T which is locally bounded in the Kolmogoroff metric

$$\sup\{|T(Q) - T(P)| : d_{ko}(P, Q) < \epsilon)\} < \infty , \tag{25.11}$$

for some $\epsilon > 0$ offers protection against outliers. On moving from local boundedness to continuity we see that if a functional T is continuous at P then the sequence $T(P_n(P))$ is a consistent statistic in that

$$\lim_{n \to \infty} T(P_n(P)) = T(P), \quad a.s.$$

Finally we consider a functional T which is differentiable at P, that is

$$T(Q) - T(P) = \int I(x, P, T)\, d(Q - P)(x) + o_P(d_{ko}(P, Q)) \tag{25.12}$$

for some bounded function $I(\cdot, P, T) : \mathbb{R} \to \mathbb{R}$ where, without loss of generality, $\int I(x, P, T)\, dP(x) = 0$ (see Clarke 1983). On putting

$$Q = Q_\epsilon = (1 - \epsilon)P + \epsilon \delta_x$$

it is seen that $I(x, P, T)$ is the influence function of (25.2). As

$$d_{ko}(P_n(P), P) = O_P(1/\sqrt{n}) \tag{25.13}$$

the central limit theorem (25.3) follows immediately. Textbooks which make use of this functional analytic approach are as already mentioned Huber (1981), Hampel et al. (1986), Rieder (1994), and also Staudte and Sheather (1990), a book which can be strongly recommended to students as a well written and at the same time deep introductory text.

25.2 Location and Scale in $\mathbb{R}$

25.2.1 Location, Scale and Equivariance

Changes in measurement units and baseline correspond to affine transformations on $\mathbb{R}$. We write

$$\mathcal{A} = \{A : \mathbb{R} \to \mathbb{R} \quad \text{with} \quad A(x) = ax + b,\ a \neq 0,\ b \in \mathbb{R}\}\,. \tag{25.14}$$

For any probability measure P and for any $A \in \mathcal{A}$ we define

$$P^A(B) = P(\{x : A(x) \in B\}), \quad B \in \mathcal{B}\,, \tag{25.15}$$

$\mathcal{B}$ denoting all Borel sets on $\mathbb{R}$. Consider a subset $\mathcal{P}'$ of $\mathcal{P}$ which is closed under affine transformations, that is

$$P \in \mathcal{P}' \Rightarrow P^A \in \mathcal{P}' \quad \text{for all } P \in \mathcal{P}',\ A \in \mathcal{A}\,. \tag{25.16}$$

A functional $T_l : \mathcal{P}' \to \mathbb{R}$ will be called a location functional on $\mathcal{P}'$ if

$$T_l(P^A) = A(T_l(P)), \quad A \in \mathcal{A},\ P \in \mathcal{P}'\,. \tag{25.17}$$

Similarly we define a functional $T_s : \mathcal{P}' \to \mathbb{R}_+$ to be a scale functional if

$$T_s(P^A) = |a| T_s(P), \quad A \in \mathcal{A},\ A(x) = ax + b,\ P \in \mathcal{P}'\,. \tag{25.18}$$

25.2.2 Existence and Uniqueness

The fact that the mean T_{av} of (25.7) cannot be defined for all distributions is an indication of its lack of robustness. More precisely the functional T_{av} is not locally bounded (25.11) in the metric d_{ko} at any distribution P. The median $\mathrm{MED}(P)$ can be defined at any distribution P as the mid-point of the interval of m-values for which

$$P((-\infty, m]) \geq 1/2 \quad \text{and} \quad P([m, \infty)) \geq 1/2\,. \tag{25.19}$$

Similar considerations apply to scale functionals. The standard deviation requires the existence of the second moment of a distribution. The median absolute deviation MAD (see Andrews et al. 1972) of a distribution can be well defined at all distributions as follows. Given P we define P' by

$$P'(B) = P(\{x : |x - \mathrm{MED}(P)| \in B\}), \quad B \in \mathcal{B}\,.$$

and set

$$\mathrm{MAD}(P) = \mathrm{MED}(P')\,. \tag{25.20}$$

25.2.3 M-Estimators

An important family of statistical functionals is the family of M-functionals introduced by Huber (1964) Let ψ and χ be functions defined on $\mathbb{R}$ with values in the interval $[-1,\ 1]$. For a given probability distribution P we consider the following two equations for m and s

$$\int \psi\left(\frac{x-m}{s}\right) dP(x) = 0 \tag{25.21}$$

$$\int \chi\left(\frac{x-m}{s}\right) dP(x) = 0. \tag{25.22}$$

If the solution exists and is uniquely defined we denote it by

$$T(P) = (T_l(P), T_s(P)) = (m, s)\,.$$

In order to guarantee existence and uniqueness conditions have to be placed on the functions ψ and χ as well as on the probability measure P. The ones we use are due to Scholz (1971) (see also Huber 1981) and are as follows:

($\boldsymbol{\psi 1}$) $\psi(-x) = -\psi(x)$ for all $x \in \mathbb{R}$.
($\boldsymbol{\psi 2}$) ψ is strictly increasing
($\boldsymbol{\psi 3}$) $\lim_{x\to\infty} \psi(x) = 1$
($\boldsymbol{\psi 4}$) ψ is continuously differentiable with derivative $\psi^{(1)}$.

($\boldsymbol{\chi 1}$) $\chi(-x) = \chi(x)$ for all $x \in \mathbb{R}$.
($\boldsymbol{\chi 2}$) $\chi : \mathbb{R}_+ \to [-1,\ 1]$ is strictly increasing
($\boldsymbol{\chi 3}$) $\chi(0) = -1$
($\boldsymbol{\chi 4}$) $\lim_{x\to\infty} \chi(x) = 1$
($\boldsymbol{\chi 5}$) χ is continuously differentiable with derivative $\chi^{(1)}$.

($\boldsymbol{\psi\chi 1}$) $\chi^{(1)}/\psi^{(1)} : \mathbb{R}_+ \to \mathbb{R}_+$ is strictly increasing.

If these conditions hold and P satisfies

$$\Delta(P) = \max_x P(\{x\}) < 1/2 \tag{25.23}$$

then (25.21) and (25.22) have precisely one solution. If we set

$$\mathcal{P}' = \{P : \Delta(P) < 1/2\}$$

then $\mathcal{P}'$ satisfies (25.16) and $T_l : \mathcal{P}' \to \mathbb{R}$ and $T_s : \mathcal{P}' \to \mathbb{R}_+$ are a location and a scale functional respectively. Two functions which satisfy the above conditions are

$$\psi(x) = \frac{\exp(x/c) - 1}{\exp(x/c) + 1} \tag{25.24}$$

$$\chi(x) = \frac{x^4 - 1}{x^4 + 1}\,, \tag{25.25}$$

where $c < 0.39$ is a tuning parameter. The restriction on c is to guarantee $(\boldsymbol{\psi\chi 1})$. Algorithms for calculating the solution of (25.21) and (25.22) are given in the Fortran library ROBETH (Marazzi 1992) which also contains many other algorithms related to robust statistics.

The main disadvantage of M-functionals defined by (25.21) and (25.22) is $(\boldsymbol{\psi\chi 1})$ which links the location and scale parts in a manner which may not be desirable. In particular there is a conflict between the breakdown behaviour and the efficiency of the M-functional (see below). There are several ways of overcoming this. One is to take the scale function T_s and then to calculate a second location functional by solving

$$\int \tilde{\psi}\left(\frac{x - m}{T_s(P)}\right) dP(x) = 0\,. \tag{25.26}$$

If now $\tilde{\psi}$ satisfies $(\boldsymbol{\psi 1})$–$(\boldsymbol{\psi 4})$ then this new functional will exist only under the assumption that the scale functional exists and is non-zero. Furthermore the functional can be made as efficient as desired by a suitable choice of $\tilde{\psi}$ removing the conflict between breakdown and efficiency. One possible choice for $T_s(P)$ is the MAD of (25.20) which is simple, highly robust and which performed well in the Princeton robustness study (Andrews et al. 1972).

In some situations there is an interest in downweighting outlying observations completely rather than in just bounding their effect. A downweighting to zero is not possible for a ψ-function which satisfies $(\boldsymbol{\psi 2})$ but can be achieved by using so called redescending ψ-functions such as Tukey's biweight

$$\tilde{\psi}(x) = x(1 - x^2)^2\{|x| \le 1\}\,. \tag{25.27}$$

In general there will be many solutions of (25.26) for such ψ-functions and to obtain a well defined functional some choice must be made. One possibility is to take the solution closest to the median, another is to take

$$\operatorname{argmin}_m \int \rho\left(\frac{x - m}{T_s(P)}\right) dP(x) \tag{25.28}$$

where $\rho^{(1)} = \tilde{\psi}$. Both solutions pose algorithmic problems. The effect of downweighting outlying observations to zero can be attained by using a so called one-step functional T_{om} defined by

$$T_{om}(P) = T_m(P) + T_s(P)\frac{\int \tilde{\psi}\left(\frac{x-T_m(P)}{T_s(P)}\right) dP(x)}{\int \tilde{\psi}^{(1)}\left(\frac{x-T_m(P)}{T_s(P)}\right) dP(x)} \tag{25.29}$$

where T_m is as above and $\tilde{\psi}$ is redescending. We refer to Hampel et al. (1986) and Rousseeuw and Croux (1994) for more details.

So far all scale functionals have been defined in terms of a deviation from a location functional. This link can be broken as follows. Consider the functional T_{ss} defined to be the solution s of

$$\int \chi\left(\frac{x-y}{s}\right) dP(x)\, dP(y) = 0\ , \tag{25.30}$$

where χ satisfies the conditions above. It may be shown that the solution is unique with $0 < s < \infty$, if

$$\sum_{a_i} P(\{a_i\})^2 < 1/4\ , \tag{25.31}$$

where the a_i denote the countably many atoms of P. The main disadvantage of this method is the computational complexity of (25.30) requiring as it does $O(n^2)$ operations for a sample of size n. If χ is of the form

$$\chi(x) = \begin{cases} a > 0, & |x| > 1 \\ b < 0, & |x| \le 1 \end{cases}\ ,$$

then T_{ss} reduces to a quantile of the $|x_i - x_j|$ and much more efficient algorithms exist which allow the functional to be calculated in $O(n \log n)$ operations (see Croux and Rousseeuw 1992; Rousseeuw and Croux 1992, 1993).

Although we have defined M-functionals as a solution of (25.21) and (25.22) there are sometimes advantages in defining them as a solution of a minimization problem. Consider the Cauchy distribution with density

$$f(x : \mu, \sigma) = \frac{1}{\pi}\frac{\sigma}{\sigma^2 + (x-\mu)^2}\,. \tag{25.32}$$

We now define $\boldsymbol{T}_c(P) = (T_{cm}(P), T_{cs}(P))$ by

$$\boldsymbol{T}_c(P) = \text{argmin}_{(m,s)}\left(-\int \log(\, f(x : m, s))\, dP(x) + \frac{1}{2}\log(s)\right)\,. \tag{25.33}$$

This is simply the standard maximum likelihood estimate for a Cauchy distribution but there is no suggestion here that the data are so distributed. If $\Delta(P) < 1/2$ it

can be shown that the solution exists and is unique. Moreover there exists a simple convergent algorithm for calculating $(T_{cm}(P), T_{cs}(P))$ for a data sample. We refer to Kent and Tyler (1991) for this and the multidimensional case to be studied below. By differentiating the right hand side of (25.33) it is seen that $(T_{cm}(P), T_{cs}(P))$ may be viewed as an M-functional with a redescending ψ-function.

Another class of functionals defined by a minimization problem is the class of S-functionals. Given a function $\rho : \mathbb{R} \to [0, 1]$ which is symmetric, continuous on the right and non-increasing on $\mathbb{R}_+$ with $\rho(1) = 1$ and $\lim_{x\to\infty} \rho(x) = 0$. We define $(T_{sm}(P), T_{ss}(P))$ by

$$(T_{sm}(P), T_{ss}(P)) = \operatorname{argmin}_{(m,s)} \left\{ s : \int \rho((x-m)/s)\, dP(x) \geq 1/2 \right\} . \quad (25.34)$$

A special case is a minor variation of the shortest-half functional of (25.8) which is obtained by taking ρ to be the indicator function of the interval $[0, 1)$. Although the existence of solutions of (25.34) is guaranteed if $\Delta(P) < 1/2$ the problem of uniqueness is not trivial and requires the existence of a density subject to certain conditions. If ρ is smooth then by differentiation it is seen that $(T_{sm}(P), T_{ss}(P))$ may be regarded as an M-functional with a redescending ψ-function given by $\tilde{\psi} = \rho^{(1)}$. The minimization problem (25.34) acts as a choice function. We refer to Davies (1987).

25.2.4 Bias and Breakdown

Given a location functional T_l the bias is defined by

$$b(T_l, P, \epsilon, d_{ko}) = \sup\{|T_l(Q) - T_l(P)| : d_{ko}(P, Q) < \epsilon\} , \quad (25.35)$$

where by convention $T_l(Q) = \infty$ if T_l is not defined at Q. For a scale functional T_s we set

$$b(T_s, P, \epsilon, d_{ko}) = \sup\{|\log(T_s(Q)/T_s(P))| : d_{ko}(P, Q) < \epsilon\} , \quad (25.36)$$

where again by convention $T_s(Q) = \infty$ if T_s is not defined at Q. A popular although weaker form of bias function based on the so called gross error neighbourhood is given by

$$b(T_l, P, \epsilon, GE) = \sup\{|T_l(Q) - T_l(P)| : Q = (1-\epsilon)P + \epsilon H, H \in \mathcal{P}\} \quad (25.37)$$

with a corresponding definition for $b(T_s, P, \epsilon, GE)$. We have

$$b(T_l, P, \epsilon, GE) \leq b(T_l, P, \epsilon, d_{ko}) . \quad (25.38)$$

We refer to Huber (1981) for more details.

The breakdown point $\epsilon^*(T_l, P, d_{ko})$ of T_l at P with respect to d_{ko} is defined by

$$\epsilon^*(T_l, P, d_{ko}) = \sup\{\epsilon : b(T_l, P, \epsilon, d_{ko}) < \infty\} \tag{25.39}$$

with the corresponding definitions for scale functionals and the gross error neighbourhood. Corresponding to (25.38) we have

$$\epsilon^*(T_l, P, d_{ko}) \leq \epsilon^*(T_l, P, GE)\,. \tag{25.40}$$

If a functional T_l has a positive breakdown point at a distribution P then it exhibits a certain degree of stability in a neighbourhood of P as may be seen as follows. Consider a sample $x_1, \ldots, x_n$ and add to it k further observations $x_{n+1}, \ldots, x_{n+k}$. If P_n and P_{n+k} denote the empirical measures based on $x_1, \ldots, x_n$ and $x_1, \ldots, x_{n+k}$ respectively then $d_{ko}(P_n, P_{n+k}) \leq k/(n+k)$. In particular if $k/(n+k) < \epsilon^*(T_l, P_n, d_{ko})$ then it follows that $T_l(P_{n+k})$ remains bounded whatever the added observations . This finite sample concept of breakdown was introduced by Donoho and Huber (1983). Another version replaces k observations by other values instead of adding k observations and is as follows. Let $x_1^k, \ldots, x_n^k$ denote a sample differing from $x_1, \ldots, x_n$ in at most k readings. We denote the empirical distributions by P_n^k and define

$$\epsilon^*(T_l, P_n, fsbp) = \max\left\{k/n : \left|T_l\left(P_n^k\right)\right| < \infty\right\}, \tag{25.41}$$

where P_n^k ranges over all possible $x_1^k, \ldots, x_n^k$. This version of the finite sample breakdown point is called the replacement version as k of the original observations can be replaced by arbitrary values. The two breakdown points are related (Zuo 2001). There are corresponding versions for scale functionals.

For location and scale functionals there exist upper bounds for the breakdown points. For location functionals T_l we have

Theorem 1.

$$\epsilon^*(T_l, P, d_{ko}) \leq 1/2, \tag{25.42}$$

$$\epsilon^*(T_l, P, GE) \leq 1/2, \tag{25.43}$$

$$\epsilon^*(T_l, P_n, fsbp) \leq \lfloor n/2 \rfloor / n. \tag{25.44}$$

We refer to Huber (1981). It may be shown that all breakdown points of the mean are zero whereas the median attains the highest possible breakdown point in each case.The corresponding result for scale functionals is more complicated. Whereas we know of no reasonable metric in (25.42) of Theorem 1 which leads to a different upper bound this is not the case for scale functionals. Huber (1981) shows that for the Kolmogoroff metric d_{ko} the corresponding upper bound is $1/4$ but is $1/2$ for the gross error neighbourhood. If we replace the Kolmogoroff metric d_{ko} by the standard Kuiper metric d_{ku} defined by

$$d_{ku}(P, Q) = \sup\{|P(I) - Q(I)| : I \text{ an interval}\} \tag{25.45}$$

then we again obtain an upper bound of $1/2$. For scale functionals T_s we have

Theorem 2.

$$\epsilon^*(T_s, P, d_{ku}) \leq (1 - \Delta(P))/2, \tag{25.46}$$

$$\epsilon^*(T_s, P, GE) \leq (1 - \Delta(P))/2, \tag{25.47}$$

$$\epsilon^*(T_s, P_n, fsbp) \leq (1 - \Delta(P))/2. \tag{25.48}$$

Similarly all breakdown points of the standard deviation are zero but, in contrast to the median, the MAD does not attain the upper bounds of (25.44). We have

$$\epsilon^*(\text{MAD}, P_n, fsbp) = \max\{0,\ 1/2 - \Delta(P_n)\}\,.$$

A simple modification of the MAD, namely

$$\text{MMAD}(P) = \min\{|I| : \tilde{P}(I) \geq (1 + \Delta(I))/2\}\,, \tag{25.49}$$

where $\tilde{P}(B) = P(\{x : |x - \text{MED}(P)| \in B\})$ and $\Delta(I) = \max\{P(\{x\}), x \in I\}$ can be shown to obtain the highest possible finite sample breakdown point of (25.48).

The M-functional defined by (25.21) and (25.22) has a breakdown point ϵ^* which satisfies

$$\psi^{-1}\left(\frac{\epsilon^*}{1 - \epsilon^*}\right) = \chi^{-1}\left(\frac{-\epsilon^*}{1 - \epsilon^*}\right) \tag{25.50}$$

(see Huber 1981). For the functions defined by (25.24) and (25.25) the breakdown point is a decreasing function of c. As c tends to zero the breakdown point tends to $1/2$. Indeed, as c tends to zero the location part of the functional tends to the median. For $c = 0.2$ numerical calculations show that the breakdown point is 0.48. The calculation of breakdown points is not always simple. We refer to Huber (1981), Huber (1984) and Gather and Hilker (1997).

The breakdown point is a simple but often effective measure of the robustness of a statistical functional. It does not however take into account the size of the bias. This can be done by trying to quantify the minimum bias over some neighbourhood of the distribution P and if possible to identify a functional which attains it. We formulate this for $P = N(0, 1)$ and consider the Kolmogoroff ball of radius ϵ. We have (Huber 1981)

Theorem 3. *For every $\epsilon < 1/2$ we have*

$$b(MED, P, \epsilon, d_{ko}) \leq b(T_l, P, \epsilon, d_{ko})$$

for any translation functional T_l.

In other words the median minimizes the bias over any Kolmogoroff neighbourhood of the normal distribution. This theorem can be extended to other symmetric distributions and to other situations (Riedel 1989a,b). It is more difficult to obtain such a theorem for scale functionals because of the lack of a property equivalent to symmetry for location. Nevertheless some results in this direction have been obtained and indicate that the length of the shortest half T_{sh} of (25.8) has very good bias properties (Martin and Zamar 1993b).

25.2.5 *Confidence Intervals and Differentiability*

Given a sample $x_1, \ldots, x_n$ with empirical measure P_n we can calculate a location functional $T_l(P_n)$ which in some sense describes the location of the sample. Such a point value is rarely sufficient and in general should be supplemented by a confidence interval, that is a range of values consistent with the data. If T_l is differentiable (25.12) and the data are i.i.d. random variables with distribution P then it follows from (25.3) (see Sect. 25.1.3) that an asymptotic α-confidence interval for $T_l(P)$ is given by

$$\left[T_l(P_n(P)) - z((1+\alpha)/2)\Sigma(P)/\sqrt{n}, T_l(P_n(P)) + z((1+\alpha)/2)\Sigma(P)/\sqrt{n}\right]. \tag{25.51}$$

Here $z(\alpha)$ denotes the α-quantile of the standard normal distribution and

$$\Sigma(P)^2 = \int I(x, T_l, P)^2 \, dP(x). \tag{25.52}$$

At first glance this cannot lead to a confidence interval as P is unknown. If however $\Sigma(P)$ is also Fréchet differentiable at P then we can replace $\Sigma(P)$ by $\Sigma(P_n(P))$ with an error of order $O_P(1/\sqrt{n})$. This leads to the asymptotic α-confidence interval

$$\begin{aligned}\Big[T_l(P_n(P)) - z((1+\alpha)/2)\Sigma(P_n(P))/\sqrt{n}, T_l(P_n(P)) \\ + z((1+\alpha)/2)\Sigma(P_n(P))/\sqrt{n}\Big].\end{aligned} \tag{25.53}$$

A second problem is that (25.53) depends on asymptotic normality and the accuracy of the interval in turn will depend on the rate of convergence to the normal distribution which in turn may depend on P. Both problems can be overcome if T_l is locally uniformly Fréchet differentiable at P. If we consider the M-functionals of Sect. 25.2.3 then they are locally uniformly Fréchet differentiable if the ψ- and χ-functions are sufficiently smooth (see Bednarski et al. 1991, Bednarski 1993, Bednarski and Clarke 1998, and Davies 1998). The influence function $I(\cdot, T_l, P)$ is given by

$$I(x, T_l, P) = T_s(P)\frac{D(P)\tilde{\psi}\left(\frac{x-T_l(P)}{T_s(P)}\right) - B(P)\chi\left(\frac{x-T_l(P)}{T_s(P)}\right)}{A(P)D(P) - B(P)C(P)}, \tag{25.54}$$

where

$$A(P) = \int \tilde{\psi}^{(1)}\left(\frac{x - T_l(P)}{T_s(P)}\right) dP(x) \tag{25.55}$$

$$B(P) = \int \left(\frac{x - T_l(P)}{T_s(P)}\right) \tilde{\psi}^{(1)}\left(\frac{x - T_l(P)}{T_s(P)}\right) dP(x) \tag{25.56}$$

$$C(P) = \int \chi^{(1)}\left(\frac{x - T_l(P)}{T_s(P)}\right) dP(x) \tag{25.57}$$

$$D(P) = \int \left(\frac{x - T_l(P)}{T_s(P)}\right) \chi^{(1)}\left(\frac{x - T_l(P)}{T_s(P)}\right) dP(x). \tag{25.58}$$

Simulations suggest that the covering probabilities of the confidence interval (25.53) are good for sample sizes of 20 or more as long as the distribution P is almost symmetric. For the sample $x_1, \ldots, x_n$ this leads to the interval

$$\left[T_l(P_n) - z((1+\alpha)/2)\Sigma(P_n)/\sqrt{n}, T_l(P_n) + z((1+\alpha)/2)\Sigma(P_n)/\sqrt{n}\right] \tag{25.59}$$

with $\Sigma(P)$ given by (25.52) and $I(x, T_l, P)$ by (25.54). Similar intervals can be obtained for the variations on M-functionals discussed in Sect. 25.2.3.

25.2.6 *Efficiency and Bias*

The precision of the functional T at the distribution P can be quantified by the length $2z((1+\alpha)/2)\Sigma(P)/\sqrt{n}$ of the asymptotic confidence interval (25.51). As the only quantity which depends on T is $\Sigma(P)$ we see that an increase in precision is equivalent to reducing the size of $\Sigma(P)$. The question which naturally arises is then that of determining how small $\Sigma(P)$ can be made. A statistical functional which attains this lower bound is asymptotically optimal and if we denote this lower bound by $\Sigma_{\text{opt}}(P)$, the efficiency of the functional T can be defined as $\Sigma_{\text{opt}}(P)^2/\Sigma(P)^2$. The efficiency depends on P and we must now decide which P or indeed Ps to choose. The arguments given in Sect. 25.1.2 suggest choosing a P which maximizes $\Sigma_{\text{opt}}(P)$ over a class of models. This holds for the normal distribution which maximizes $\Sigma_{\text{opt}}(P)$ over the class of all distributions with a given variance. For this reason and for simplicity and familiarity we shall take the normal distribution as the reference distribution. If a reference distribution is required which also produces outliers then the slash distribution is to be preferred to the Cauchy distribution. We refer to Cohen (1991) and the discussion given there.

If we consider the M-functionals defined by (25.24) and (25.25) the efficiency at the normal distribution is an increasing function of the tuning parameter c. As the breakdown point is a decreasing function of c this would seem to indicate that there is a conflict between efficiency and breakdown point. This is the case for the M-functional defined by (25.24) and (25.25) and is due to the linking of the location and scale parts of the functional. If this is severed by, for example, recalculating a location functional as in (25.26) then there is no longer a conflict between efficiency and breakdown. As however the efficiency of the location functional increases the more it behaves like the mean with a corresponding increase in the bias function of (25.35) and (25.37). The conflict between efficiency and bias is a real one and gives rise to an optimality criterion, namely that of minimizing the bias subject to a lower bound on the efficiency. We refer to Martin and Zamar (1993a).

25.2.7 Outliers in $\mathbb{R}$

One of the main uses of robust functionals is the labelling of so called outliers (see Barnett and Lewis 1994; Hawkins 1980; Atkinson 1994; Gather 1990; Gather et al. 2003; and Simonoff 1984, 1987). In the data of Table 25.1 the laboratories 1 and 3 are clearly outliers which should be flagged. The discussion in Sect. 25.1.1 already indicates that the mean and standard deviation are not appropriate tools for the identification of outliers as they themselves are so strongly influenced by the very outliers they are intended to identify. We now demonstrate this more precisely. One simple rule is to classify all observations more than three standard deviations from the mean as outliers. A simple calculation shows that this rule will fail to identify 10% arbitrarily large outliers with the same sign. More generally if all observations more than λ standard deviations from the mean are classified as outliers then this rule will fail to identify a proportion of $1/(1+\lambda^2)$ outliers with the same sign. This is known as the masking effect (Pearson and Chandra Sekar 1936) where the outliers mask their presence by distorting the mean and, more importantly, the standard deviation to such an extent as to render them useless for the detection of the outliers. One possibility is to choose a small value of λ but clearly if λ is too small then some non-outliers will be declared as outliers. In many cases the main body of the data can be well approximated by a normal distribution so we now investigate the choice of λ for samples of i.i.d. normal random variables. One possibility is to choose λ dependent on the sample size n so that with probability say 0.95 no observation will be flagged as an outlier. This leads to a value of λ of about $\sqrt{2\log(n)}$ (Davies and Gather 1993) and the largest proportion of one-sided outliers which can be detected is approximately $1/(1+2\log(n))$ which tends to zero with n. It follows that there is no choice of λ which can detect say 10% outliers and at the same time not falsely flag non-outliers. In order to achieve this the mean and standard deviation must be replaced by functionals which are less effected by the outliers. In particular these functionals should be locally bounded (25.11). Considerations of asymptotic normality or efficiency are of little relevance

here. Two obvious candidates are the median and MAD and if we use them instead of the mean and standard deviation we are led to the identification rule (Hampel 1985) of the form

$$|x_i - \mathrm{MED}(\boldsymbol{x}_n)| \geq \lambda \mathrm{MAD}(\boldsymbol{x}_n)\,. \tag{25.60}$$

Hampel (1975) proposed setting $\lambda = 5.2$ as a general all purpose value. The concept of an outlier cannot in practice be very precise but in order to compare different identification rules we require a precise definition and a precise measure of performance. To do this we shall restrict attention to the normal model as one which is often reasonable for the main body of data. In other situations such as waiting times the exponential distribution may be more appropriate. The following is based on Davies and Gather (1993). To define an outlier we introduce the concept of an α-outlier. For the normal distribution $N(\mu, \sigma^2)$ and $\alpha \in (0, 1)$ we define the α-outlier region by

$$\mathrm{out}(\alpha, N(\mu, \sigma^2)) = \{x \in \mathbb{R} \colon |x - \mu| > \sigma\, z_{1-\alpha/2}\}\,, \tag{25.61}$$

which is just the union of the lower and the upper $\alpha/2$-tail regions. Here $z_{1-\alpha/2}$ denotes the $1-\alpha/2$-quantile of the standard normal distribution. For the exponential distribution $\mathrm{Exp}(\lambda)$ with parameter λ we set

$$\mathrm{out}(\alpha, \mathrm{Exp}(\lambda)) = \{x \in \mathbb{R} \colon x > -\lambda \ln \alpha\} \tag{25.62}$$

which is the upper α-tail region (Gather and Schultze 1999). The extension to other distributions P is clear. Each point located in the outlier region is called an α-outlier, otherwise it is called an α-inlier. This definition of an outlier refers only to its position in relation to the statistical model for the good data. No assumptions are made concerning the distribution of these outliers or the mechanism by which they are generated.

We can now formulate the task of outlier identification for the normal distribution as follows: For a given sample $\boldsymbol{x}_n = (x_1, \ldots, x_n)$ which contains at least $[n/2] + 1$ i.i.d. observations distributed according to $N(\mu, \sigma^2)$, we have to find all those x_i that are located in $\mathrm{out}(\alpha, N(\mu, \sigma^2))$. The level α can be chosen to be dependent on the sample size. If for some $\tilde{\alpha} \in (0, 1)$ we set

$$\alpha = \alpha_n = 1 - (1 - \tilde{\alpha})^{1/n}\,, \tag{25.63}$$

then the probability of finding at least one observation of a $N(\mu, \sigma^2)$-sample of size n within $\mathrm{out}(\alpha_n, N(\mu, \sigma^2))$ is not larger than $\tilde{\alpha}$. Consider now the general Hampel identifier which classifies all observations x_i in

$$\mathrm{OR}^H(\mathbf{x}_n, \alpha_n) = \{x \in \mathbb{R} \colon |x - \mathrm{Med}(\mathbf{x}_n)| > g_n(\alpha_n)\, \mathrm{MAD}(\mathbf{x}_n)\} \tag{25.64}$$

as outliers. The region $\mathrm{OR}^H(\mathbf{x}_n, \alpha_n)$ may be regarded as an empirical version of the outlier region $\mathrm{out}(\alpha_n, N(\mu, \sigma^2))$. The constant $g_n(\alpha_n)$ standardizes the behaviour of the procedure for i.i.d. normal samples which may be done in several ways. One is to determine the constant so that with probability at least $1 - \tilde{\alpha}$ no observation X_i is

identified as an outlier, that is

$$P\big(X_i \notin \mathrm{OR}(\mathbf{X}_n, \alpha_n),\ i = 1, \ldots, n\big) \ \geq\ 1 - \tilde{\alpha}. \tag{25.65}$$

A second possibility is to require that

$$P\big(\mathrm{OR}(\mathbf{X}_n, \alpha_n) \subset \mathrm{out}(\alpha_n, P)\big) \ \geq\ 1 - \tilde{\alpha}. \tag{25.66}$$

If we use (25.65) and set $\tilde{\alpha} = 0.05$ then for $n = 20, 50$ and 100 simulations give $g_n(\alpha_n) = 5.82,\ 5.53$ and 5.52 respectively. For $n > 10$ the normalizing constants $g_n(\alpha_n)$ can also be approximated according to the equations given in Sect. 5 of Gather (1990).

To describe the worst case behaviour of an outlier identifier we can look at the largest nonidentifiable outlier, which it allows. From Davies and Gather (1993) we report some values of this quantity for the Hampel identifier (HAMP) and contrast them with the corresponding values of a sophisticated high breakdown point outwards testing identifier (ROS), based on the non-robust mean and standard deviation (Rosner 1975; Tietjen and Moore 1972). Both identifiers are standardized by (25.65) with $\tilde{\alpha} = 0.05$. Outliers are then observations with absolute values greater than $3.016 (n = 20)$, $3.284 (n = 50)$ and $3.474 (n = 100)$. For $k = 2$ outliers and $n = 20$ the average sizes of the largest non-detected outlier are 6.68 (HAMP) and 8.77 (ROS), for $k = 5$ outliers and $n = 50$ the corresponding values are 4.64 (HAMP) and 5.91 (ROS) and finally for $k = 15$ outliers and $n = 100$ the values are 5.07 (HAMP) and 9.29 (ROS).

25.3 Location and Scale in $\mathbb{R}^k$

25.3.1 Equivariance and Metrics

In Sect. 25.2.1 we discussed the equivariance of estimators for location and scale with respect to the affine group of transformations on $\mathbb{R}$. This carries over to higher dimensions although here the requirement of affine equivariance lacks immediate plausibility. A change of location and scale for each individual component in $\mathbb{R}^k$ is represented by an affine transformation of the form $A(x) + b$ where A is a diagonal matrix. A general affine transformation forms linear combinations of the individual components which goes beyond arguments based on units of measurement. The use of affine equivariance reduces to the almost empirical question as to whether the data, regarded as a cloud of points in $\mathbb{R}^k$, can be well represented by an ellipsoid. If this is the case as it often is then consideration of linear combinations of different components makes data analytical sense. With this proviso in mind we consider the affine group $\mathcal{A}$ of transformations of $\mathbb{R}^k$ into itself,

$$\boldsymbol{\mathcal{A}} = \{\mathcal{A} : \mathcal{A}(x) = A(x) + b\}\,, \tag{25.67}$$

where A is a non-singular $k \times k$-matrix and b is an arbitrary point in $\mathbb{R}^k$. Let $\mathcal{P}'_k$ denote a family of distributions over $\mathbb{R}^k$ which is closed under affine transformations

$$P \in \mathcal{P}'_k \Rightarrow P^{\mathcal{A}} \in \mathcal{P}'_k, \text{ for all } \mathcal{A} \in \boldsymbol{\mathcal{A}}\,. \tag{25.68}$$

A function $T_l : \mathcal{P}'_k \to \mathbb{R}^k$ is called a location functional if it is well defined and

$$T_l(P^{\mathcal{A}}) = \mathcal{A}(T_l(P)), \text{ for all } \mathcal{A} \in \boldsymbol{\mathcal{A}},\ P \in \mathcal{P}'_k\,. \tag{25.69}$$

A functional $T_s : \mathcal{P}'_k \to \boldsymbol{\Sigma}_k$ where $\boldsymbol{\Sigma}_k$ denotes the set of all strictly positive definite symmetric $k \times k$ matrices is called a scale or scatter functional if

$$T_s(P^{\mathcal{A}}) = AT_l(P)A^{\top}, \text{ for all } \mathcal{A} \in \boldsymbol{\mathcal{A}},\ P \in \mathcal{P}'_k \text{ with } \mathcal{A}(x) = A(x) + b\,. \tag{25.70}$$

The requirement of affine equivariance is a strong one as we now indicate. The most obvious way of defining the median of a k-dimensional data set is to define it by the medians of the individual components. With this definition the median is equivariant with respect to transformations of the form $\Lambda(x) + b$ with Λ a diagonal matrix but it is not equivariant for the affine group. A second possibility is to define the median of a distribution P by

$$\mathrm{MED}(P) = \mathrm{argmin}_{\mu} \int (\|x - \mu\| - \|x\|)\, dP(x)\,.$$

With this definition the median is equivariant with respect to transformations of the form $x \to O(x) + b$ with O an orthogonal matrix but not with respect to the affine group or the group $x \to \Lambda(x) + b$ with Λ a diagonal matrix. The conclusion is that there is no canonical extension of the median to higher dimensions which is equivariant with respect to the affine group.

In Sect. 25.2 use was made of metrics on the space of probability distributions on $\mathbb{R}$. We extend this to $\mathbb{R}^k$ where all metrics we consider are of the form

$$d_{\mathcal{C}}(P, Q) = \sup_{C \in \mathcal{C}} |P(C) - Q(C)| \tag{25.71}$$

where $\mathcal{C}$ is a so called Vapnik–Cervonenkis class (see for example Pollard 1984). The class $\mathcal{C}$ can be chosen to suit the problem. Examples are the class of all lower dimensional hyperplanes

$$\mathcal{H} = \{H : H \text{ lower dimensional hyperplane}\} \tag{25.72}$$

and the class of all ellipsoids

$$\mathcal{E} = \{E : E \text{ an ellipsoid}\}. \tag{25.73}$$

These give rise to the metrics $d_{\mathcal{H}}$ and $d_{\mathcal{E}}$ respectively. Just as in $\mathbb{R}$ metrics $d_{\mathcal{C}}$ of the form (25.71) allow direct comparisons between empirical measures and models. We have

$$d_{\mathcal{C}}(P_n(P), P) = \mathrm{O}(1/\sqrt{n}) \tag{25.74}$$

uniformly in P (see Pollard 1984).

25.3.2 M-estimators of Location and Scale

Given the usefulness of M-estimators for one dimensional data it seems natural to extend the concept to higher dimensions. We follow Maronna (1976). For any positive definite symmetric $k \times k$-matrix Σ we define the metric $d(\cdot, \cdot : \Sigma)$ by

$$d(x, y : \Sigma)^2 = (x - y)^\top \Sigma^{-1}(x - y), \quad x, y \in \mathbb{R}^k.$$

Further, let u_1 and u_2 be two non-negative continuous functions defined on $\mathbb{R}_+$ and be such that $su_i(s), s \in \mathbb{R}_+, i = 1, 2$ are both bounded. For a given probability distribution P on the Borel sets of $\mathbb{R}^k$ we consider in analogy to (25.21) and (25.22) the two equations in μ and Σ

$$\int (x - \mu)u_1(d(x, \mu; \Sigma))\, dP = 0. \tag{25.75}$$

$$\int u_2(d(x, \mu; \Sigma)^2)(x - \mu)(x - \mu)^\top\, dP = 0. \tag{25.76}$$

Assuming that at least one solution (μ, Σ) exists we denote it by $T_M(P) = (\mu, \Sigma)$. The existence of a solution of (25.75) and (25.76) can be shown under weak conditions as follows. If we define

$$\Delta(P) = \max\{P(H) : H \in \mathcal{H}\} \tag{25.77}$$

with $\mathcal{H}$ as in (25.73) then a solution exists if $\Delta(P) < 1 - \delta$ where δ depends only on the functions u_1 and u_2 (Maronna 1976). Unfortunately the problem of uniqueness is much more difficult than in the one-dimensional case. The conditions placed on P in Maronna (1976) are either that it has a density $f_P(x)$ which is a decreasing function of $\|x\|$ or that it is symmetric $P(B) = P(-B)$ for every Borel set B. Such conditions do not hold for real data sets which puts us in an awkward position. Furthermore without existence and uniqueness there can be no results on asymptotic normality and consequently no results on confidence intervals. The situation is unsatisfactory so we now turn to the one class of M-functionals for which existence and uniqueness can be shown. The following is based on Kent and Tyler (1991) and

is the multidimensional generalization of (25.33). The k-dimensional t-distribution with density $f_{k,\nu}(\cdot : \mu, \Sigma)$ is defined by

$$f_{k,\nu}(x : \mu, \Sigma) = \frac{\Gamma(\frac{1}{2}(\nu + k))}{(\nu k)^{k/2}\Gamma\left(\frac{1}{2}\nu\right)}|\Sigma|^{-\frac{1}{2}}\left(1 + \frac{1}{\nu}(x-\mu)^{top}\Sigma^{-1}(x-\mu)\right)^{-\frac{1}{2}(\nu+k)} \tag{25.78}$$

and we consider the minimization problem

$$T_M(p) = (T_l(P), T_s(P)) = \text{argmin}_{\mu,\,\Sigma}\int f_{k,\nu}(x : \mu, \Sigma)\, dP(x) + \frac{1}{2}\log(|\Sigma|) \tag{25.79}$$

where $|\Sigma|$ denotes the determinant of the positive definite matrix Σ. For any distribution P on the Borel sets of $\mathbb{R}^k$ we define $\Delta(P)$ which is the k-dimensional version of (25.23). It can be shown that if $\Delta(P) < 1/2$ then (25.79) has a unique solution. Moreover for data sets there is a simple algorithm which converges to the solution. On differentiating the right hand side of (25.79) it is seen that the solution is an M-estimator as in (25.75) and (25.76). Although this has not been proven explicitly it seems clear that the solution will be locally uniformly Fréchet differentiable, that is, it will satisfy (25.12) where the influence function $I(x, T_M, P)$ can be obtained as in (25.54) and the metric d_{ko} is replaced by the metric $d_{\mathcal{H}}$. This together with (25.74) leads to uniform asymptotic normality and allows the construction of confidence regions. The only weakness of the proposal is the low gross error breakdown point $\epsilon^*(T_M, P, GE)$ defined below which is at most $1/(k+1)$. This upper bound is shared with the M-functionals defined by (25.75) and (25.76) (Maronna 1976). The problem of constructing high breakdown functionals in k dimensions will be discussed below.

25.3.3 Bias and Breakdown

The concepts of bias and breakdown developed in Sect. 25.2.4 carry over to higher dimensions. Given a metric d on the space of distributions on $\mathbb{R}^k$ and a location functional T_l we follow (25.37) and define

$$b(T_l, P, \epsilon, d) = \sup\{\|T_l(Q)\| : d(P, Q) < \epsilon\} \tag{25.80}$$

and

$$b(T_l, P, \epsilon, GE) = \sup\{\|T_l(Q)\| : Q = (1-\epsilon)P + \epsilon G,\ G \in \mathcal{P}\}, \tag{25.81}$$

where by convention $\|T_l(Q)\| = \infty$ if T_l is not defined at Q. The extension to scale functionals is not so obvious as there is no canonical definition of bias. We require a measure of difference between two positive definite symmetric matrices.

For reasons of simplicity and because it is sufficient for our purposes the one we take is $|\log(|\Sigma_1|/|\Sigma_2|)|$. Corresponding to (25.36) we define

$$b(T_s, P, \epsilon, d) = \sup\{|\log(|T_s(Q)|/|T_s(P)|)| : d(P, Q) < \epsilon\} \tag{25.82}$$

and

$$b(T_s, P, \epsilon, GE) = \sup\{|\log(|T_s(Q)|/|T_s(P)|)| : Q = (1-\epsilon)P + \epsilon G,\ G \in \mathcal{P}\}. \tag{25.83}$$

Most work is done using the gross error model (25.81) and (25.83). The breakdown points of T_l are defined by

$$\epsilon^*(T_l, P, d) = \sup\{\epsilon : b(T_l, P, \epsilon, d) < \infty\} \tag{25.84}$$

$$\epsilon^*(T_l, P, GE) = \sup\{\epsilon : b(T_l, P, \epsilon, GE) < \infty\} \tag{25.85}$$

$$\epsilon^*(T_l, P_n, fsbp) = \max\left\{k/n : \left|T_l\left(P_n^k\right)\right| < \infty\right\}, \tag{25.86}$$

where (25.86) corresponds in the obvious manner to (25.41). The breakdown points for the scale functional T_s are defined analogously using the bias functional (25.82). We have

Theorem 4. *For any translation equivariant functional* T_l

$$\epsilon^*(T_l, P, d_{\mathcal{H}}) \le 1/2 \text{ and } \epsilon^*(T_l, P_n, fsbp) \le \lfloor n/2 \rfloor / n \tag{25.87}$$

and for any affine equivariant scale functional

$$\epsilon^*(T_s, P, d_{\mathcal{E}}) \le (1 - \Delta(P))/2 \text{ and } \epsilon^*(T_s, P_n, fsbp) \le (1 - \Delta(P_n))/2\,. \tag{25.88}$$

In Sect. 25.2.4 it was shown that the M-estimators of Sect. 25.2.3 can attain or almost attain the upper bounds of Theorem 1. Unfortunately this is not the case in k dimensions where as we have already mentioned the breakdown points of M-functionals of Sect. 25.3.2 are at most $1/(k+1)$. In recent years much research activity has been directed towards finding high breakdown affinely equivariant location and scale functionals which attain or nearly attain the upper bounds of Theorem 4. This is discussed in the next section.

25.3.4 High Breakdown Location and Scale Functionals in $\mathbb{R}^k$

The first high breakdown affine equivariant location and scale functionals were proposed independently of each other by Stahel (1981) and Donoho (1982). They were defined for empirical data but the construction can be carried over to measures satisfying a certain weak condition. The idea is to project the data points onto lines

through the origin and then to determine which points are outliers with respect to this projection using one-dimensional functions with a high breakdown point. More precisely we set

$$o(x_i, \boldsymbol{\theta}) = \left| x_i^\top \boldsymbol{\theta} - \text{MED}\left(x_1^\top \boldsymbol{\theta}, \ldots, x_n^\top \boldsymbol{\theta}\right) \right| / \text{MAD}\left(x_1^\top \boldsymbol{\theta}, \ldots, x_n^\top \boldsymbol{\theta}\right) \tag{25.89}$$

and

$$o(x_i) = \sup\{o(x_i, \boldsymbol{\theta}) : \|\boldsymbol{\theta}\| = 1\}\,. \tag{25.90}$$

This is a measure for the outlyingness of the point x_i and it may be checked that it is affine invariant. Location and scale functionals may now be obtained by taking for example the mean and the covariance matrix of those $\lfloor n/2 + 1 \rfloor$ observations with the smallest outlyingness measure. Although (25.90) requires a supremum over all values of $\boldsymbol{\theta}$ this can be reduced for empirical distributions as follows. Choose all linearly independent subsets $x_{i_1}, \ldots, x_{i_k}$ of size k and for each such subset determine a $\boldsymbol{\theta}$ which is orthogonal to their span. If the sup in (25.90) is replaced by a maximum over all such $\boldsymbol{\theta}$ then the location and scale functionals remain affine equivariant and retain the high breakdown point. Although this requires the consideration of only a finite number of directions namely at most $\binom{n}{k}$ this number is too large to make it a practicable possibility even for small values of n and k. The problem of calculability has remained with high breakdown methods ever since and it is their main weakness. There are still no high breakdown affine equivariant functionals which can be calculated exactly except for very small data sets. Huber (1995) goes as far as to say that the problem of calculability is the breakdown of high breakdown methods. This is perhaps too pessimistic but the problem remains unsolved.

Rousseeuw (1985) introduced two further high breakdown location and scale functionals as follows. The first, the so called minimum volume ellipsoid (MVE) functional, is a multidimensional version of Tukey's shortest half-sample (25.8) and is defined as follows. We set

$$E = \text{argmin}_{\widetilde{E}}\{|\widetilde{E}| : |\{i : x_i \in \widetilde{E}\}| \geq \lfloor n/2 \rfloor\}\,, \tag{25.91}$$

where $|E|$ denotes the volume of E and $|\{\}|$ denotes the number of elements of the set $\{\}$. In other words E has the smallest volume of any ellipsoid which contains more than half the data points. For a general distribution P we define

$$E(P) = \text{argmin}_{\widetilde{E}} \left\{ |\widetilde{E}| : \int_{\widetilde{E}} dP \geq 1/2 \right\}\,. \tag{25.92}$$

Given E the location functional $T_l(P)$ is defined to be the centre $\mu(E)$ of E and the covariance functional $T_s(P)$ is taken to be $c(k)\Sigma(E)$ where

$$E = \left\{x : (x - \mu(E))^\top \Sigma^{-1}(x - \mu(E)) \leq 1\right\}\,. \tag{25.93}$$

The factor $c(k)$ can be chosen so that $c(k)\Sigma(E) = I_k$ for the standard normal distribution in k dimensions.

The second functional is based on the so called minimum covariance determinant (MCD) and is as follows. We write

$$\mu(B) = \int_B x\, dP(x)/P(B) \tag{25.94}$$

$$\Sigma(B) = \int_B (x-\mu(B))(x-\mu(B))^\top\, dP(x)/P(B) \tag{25.95}$$

and define

$$\mathrm{MCD}(P) = \mathrm{argmin}_B \{|\Sigma(B)| : P(B) \geq 1/2\}, \tag{25.96}$$

where $|\Sigma(B)|$ is defined to be infinite if either of (25.94) or (25.95) does not exist. The location functional is taken to be $\mu(\mathrm{MCD}(B))$ and the scatter functional $c(k)\Sigma(\mathrm{MCD}(B))$ where again $c(k)$ is usually chosen so that $c(k)\Sigma(\mathrm{MCD}(B)) = I_k$ for the standard normal distribution in k-dimensions. It can be shown that both these functionals are affinely equivariant.

A smoothed version of the minimum volume estimator can be obtained by considering the minimization problem

$$\text{minimize } |\Sigma| \text{ subject to } \int \rho\left((x-\mu)^\top \Sigma^{-1}(x-\mu)\right) dP(x) \geq 1/2, \tag{25.97}$$

where $\rho : \mathbb{R}_+ \to [0,\, 1]$ satisfies $\rho(0) = 1, \lim_{x\to\infty} \rho(x) = 0$ and is continuous on the right (see Davies 1987). This gives rise to the class of so called S-functionals. The minimum volume estimator can be obtained by specializing to the case $\rho(x) = \{0 \leq x < 1\}$.

On differentiating (25.97) it can be seen that an S-functional can be regarded as an M-functional but with redescending functions u_1 and u_2 in contrast to the conditions placed on u_1 and u_2 in (25.75) and (25.76) (Lopuhaa 1989). For such functions the defining equations for an M-estimator have many solutions and the minimization problem of (25.97) can be viewed as a choice function. Other choice functions can be made giving rise to different high breakdown M-estimators. We refer to Lopuhaa (1991) and Kent and Tyler (1996). A further class of location and scatter functionals have been developed from Tukey's concept of depth (Tukey 1975). We refer to Donoho and Gasko (1992), Liu at al. (1999) and Zuo and Serfling (2000a,b). Many of the above functionals have breakdown points close to or equal to the upper bound of Theorem 4. For the calculation of breakdown points we refer to Davies (1987), Lopuhaa and Rousseeuw (1991), Donoho and Gasko (1992), Davies (1993) and Tyler (1994).

The problem of determining a functional which minimizes the bias over a neighbourhood was considered in the one-dimensional case in Sect. 25.2.4. The problem is much more difficult in $\mathbb{R}^k$ but some work in this direction has been done (see Adrover 1998). The more tractable problem of determining the size of the bias

function for particular functionals or classes of functionals has also been considered (Maronna et al. 1992; Yohai and Maronna 1990).

All the above functionals can be shown to exist but there are problems concerning the uniqueness of the functional. Just as in the case of Tukey's shortest half (25.8) restrictions must be placed on the distribution P which generally include the existence of a density with given properties (see Davies 1987 and Tatsuoka and Tyler 2000) and which is therefore at odds with the spirit of robust statistics. Moreover even uniqueness and asymptotic normality at some small class of models are not sufficient. Ideally the functional should exist and be uniquely defined and locally uniformly Fréchet differentiable just as in Sect. 25.2.5. It is not easy to construct affinely equivariant location and scatter functionals which satisfy the first two conditions but it has been accomplished by Dietel (1993) using the Stahel–Donoho idea of projections described above. To go further and define functionals which are also locally uniformly Fréchet differentiable with respect to some metric d_C just as in the one-dimensional case considered in Sect. 25.2.5 is a very difficult problem. The only result in this direction is again due to Dietel (1993) who managed to construct functionals which are locally uniformly Lipschitz. The lack of locally uniform Fréchet differentiability means that all derived confidence intervals will exhibit a certain degree of instability. Moreover the problem is compounded by the inability to calculate the functionals themselves. To some extent it is possible to reduce the instability by say using the MCD functional in preference to the MVE functional, by reweighting the observations or by calculating a one-step M-functional as in (25.29) (see Davies 1992a). However the problem remains and for this reason we do not discuss the research which has been carried out on the efficiency of the location and scatter functionals mentioned above. Their main use is in data analysis where they are an invaluable tool for detecting outliers. This will be discussed in the following section.

A scatter matrix plays an important role in many statistical techniques such as principal component analysis and factor analysis. The use of robust scatter functionals in some of these areas has been studied by among others Croux and Haesbroeck (2000), Croux and Dehon (2001) and Willems et al. (2002).

As already mentioned the major weakness of all known high breakdown functionals is their computational complexity. For the MCD functional an exact algorithm of the order of $n^{k(k+3)/2}$ exists and there are reasons for supposing that this cannot be reduced to below n^k (Bernholt and Fischer 2001). This means that in practice for all but very small data sets heuristic algorithms have to be used. We refer to Rousseeuw and Van Driesen (1999) for a heuristic algorithm for the MCD-functional and also to Rousseeuw and Van Driesen (2000).

25.3.5 Outliers in $\mathbb{R}$

Whereas for univariate, bivariate and even trivariate data outliers may often be found by visual inspection, this is not practical in higher dimensions (Barme-Delcroix

and Gather 2000; Caroni and Prescott 1992; Gnanadesikan and Kettenring 1972; Hadi 1994; Hadi and Simonoff 1997). This makes it all the more important to have methods which automatically detect high dimensional outliers. Much of the analysis of the one-dimensional problem given in Sect. 25.2.7 carries over to the k-dimensional problem. In particular outlier identification rules based on the mean and covariance of the data suffer from masking problems and must be replaced by high breakdown functionals (see also Rocke and Woodruff 1996, 1997). We restrict attention to affine equivariant functionals so that an affine transformation of the data will not alter the observations which are identified as outliers. The identification rules we consider are of the form

$$(x_i - T_l(P_n))^\top T_s(P_n)^{-1}(x_i - T_l(P_n)) \geq c(k,n)\,, \tag{25.98}$$

where P_n is the empirical measure, T_l and T_s are affine equivariant location and scatter functionals respectively and $c(k,n)$ is a constant to be determined. This rule is the k-dimensional counterpart of (25.60). In order to specify some reasonable value for $c(k,n)$ and in order to be able to compare different outlier identifiers we require, just as in Sect. 25.2.7, a precise definition of an outlier and a basic model for the majority of the observations. As our basic model we take the k-dimensional normal distribution $\mathcal{N}(\mu, \Sigma)$. The definition of an α_n-outlier corresponds to (25.62) and is

$$\text{out}(\alpha_n, \mu, \Sigma) = \left\{x \in \mathbb{R}^k : (x-\mu)^\top \Sigma^{-1}(x-\mu) > \chi^2_{k;1-\alpha_n}\right\}\,, \tag{25.99}$$

where $\alpha_n = 1 - (1-\widetilde{\alpha})^{1/n}$ for some given value of $\widetilde{\alpha} \in (0,1)$. Clearly for an i.i.d. sample of size n distributed according to $\mathcal{N}(\mu, \Sigma)$ the probability that no observation lies in the outlier region of (25.99) is just $1-\alpha$. Given location and scale functionals T_l and T_s and a sample $\widetilde{x}_n$ we write

$$\text{OR}^H(\widetilde{x}_n, \alpha_n) = \left\{x \in \mathbb{R}^k : (x - T_l(P_n))^\top T_s(P_n)^{-1}(x - T_l(P_n)) \geq c(k,n,\alpha_n)\right\} \tag{25.100}$$

which corresponds to (25.64). The region $\text{OR}^H(\widetilde{x}_n, \alpha_n)$ is the empirical counterpart of $\text{out}(\alpha_n, \mu, \Sigma)$ of (25.99) and any observation lying in $\text{OR}^H(\widetilde{x}_n, \alpha_n)$ will be identified as an outlier. Just as in the one-dimensional case we determine the $c(k,n,\alpha_n)$ by requiring that with probability $1-\widetilde{\alpha}$ no observation is identified as an outlier in i.i.d. $\mathcal{N}(\mu, \Sigma)$ samples of size n. This can be done by simulations with appropriate asymptotic approximations for large n. The simulations will of course be based on the algorithms used to calculate the functionals and will not be based on the exact functionals assuming these to be well defined. For the purpose of outlier identification this will not be of great consequence. We give results for three multivariate outlier identifiers based on the MVE- and MCD-functionals of Rousseeuw (1985) and the S-functional based on Tukey's biweight function as given in Rocke (1996). There are good heuristic algorithms for calculating these functionals at least approximately (Rocke 1996; Rousseeuw and Van Driesen 1999;

Table 25.2 Normalizing constants $c(k, n, \alpha_n)$ for $\mathbf{OR}_{MVE}$, $\mathbf{OR}_{MCD}$, $\mathbf{OR}_{BW}$ for $\alpha = 0.1$

n	k	c_{MVE}	c_{MCD}	c_{BW}
20	2	19.14222	85.58786	21.35944
20	3	23.47072	167.61310	26.87044
20	4	33.72110	388.84680	33.17018
50	2	17.54896	28.51695	16.93195
50	3	20.61580	41.83594	19.78682
50	4	24.65417	64.18462	23.14061

Rousseeuw and van Zoomeren 1990). The following is based on Becker and Gather (2001). Table 25.2 gives the values of $c(k, n, \alpha_n)$ with $\alpha = 0.1$. The results are based on 10000 simulations for each combination of k and n.

Becker and Gather (2001) show by simulations that although none of the above rules fails to detect arbitrarily large outliers it still can happen that very extreme observations are not identified as outliers. To quantify this we consider the identifier OR_{MVE} and the constellation $n = 50, k = 2$ with $m = 5$ observations replaced by other values. The mean norm of the most extreme nonidentifiable outlier is 4.17. The situation clearly becomes worse with an increasing proportion of replaced observations and with the dimension k (see Becker and Gather 1999). If we use the mean of the norm of the most extreme non-identifiable outlier as a criterion then none of the three rules dominates the others although the biweight identifier performs reasonably well in all cases and is our preferred choice.

25.4 Linear Regression

25.4.1 Equivariance and Metrics

The linear regression model may be written in the form

$$Y_i = \boldsymbol{x}_i^\top \beta + \epsilon_i, \quad i = 1, \ldots, n \tag{25.101}$$

where $\boldsymbol{x}_i$, $i = 1 \ldots, n$ and $\beta \in \mathbb{R}^k$. The assumptions of the standard model are that the x_i are fixed and that the ϵ_i are i.i.d. random variables with the default distribution being the normal distribution $N(0, \sigma^2)$. There are of course many other models in the literature including random x_i-values and a covariance structure for the errors ϵ_i. For the purpose of robust regression we consider probability distributions P on $\mathbb{R}^{k+1}$ where the first k components refer to the covariates $\boldsymbol{x}$ and the last component is the corresponding value of y. We restrict attention to the family $\mathcal{P}_{k+1}$ of probability measures given by

$$\mathcal{P}_{k+1} = \{P : P(H \times \mathbb{R}) < 1 \text{ for all lower dimensional subspaces } H \subset \mathbb{R}^k\}. \tag{25.102}$$

The metric we use on $\mathcal{P}_{k+1}$ is $d_{\mathcal{H}}$ with $\mathcal{H}$ given by (25.73).

Consider the regression group G of transformations $g : \mathbb{R}^{k+1} \to \mathbb{R}^{k+1}$ of the form

$$g(\boldsymbol{x}, y) = \left(A(\boldsymbol{x}), sy + \boldsymbol{x}^\top \gamma\right) \tag{25.103}$$

where A is a non-singular $k \times k$-matrix, $s \in \mathbb{R}, s \neq 0$, and $\gamma \in \mathbb{R}^k$. A functional $T : \mathcal{P}_{k+1} \to \mathbb{R}^k \times \mathbb{R}_+$ is called a regression functional if for all $g \in G$ and $P \in \mathcal{P}_{k+1}$

$$T(P^g) = h_g(T(P)), \tag{25.104}$$

where

$$h_g(\beta, \sigma) = \left(s(A^{-1})^\top(\beta + \gamma), s\sigma\right). \tag{25.105}$$

with A and γ as in (25.103). The first k components of $T(P)$ specify the value of $\beta \in \mathbb{R}^k$ and the last component that of σ. The restriction to models $P \in \mathcal{P}_{k+1}$ of (25.102) is that without such a restriction there is no uniquely defined value of β.

25.4.2 M-Estimators for Regression

Given a distribution $P \in \mathcal{P}_{k+1}$ we define an M-functional by $T(P) = (\beta^*, \sigma^*)$ where (β^*, σ^*) is a solution of the equations

$$\int \phi\left(\boldsymbol{x}, \left(y - \boldsymbol{x}^\top \beta\right)/\sigma\right) \boldsymbol{x}\, dP(\boldsymbol{x}, y) = 0, \tag{25.106}$$

$$\int \chi\left(\left(y - \boldsymbol{x}^\top \beta\right)/\sigma\right)\, dP(\boldsymbol{x}, y) = 0 \tag{25.107}$$

for given functions $\phi : \mathbb{R}^{k+1} \to \mathbb{R}$ and $\chi : \mathbb{R} \to \mathbb{R}$. Just as in Sect. 25.3.2 for M-functionals of location and scatter there are problems concerning the existence and uniqueness. Maronna and Yohai (1981) give sufficient conditions for existence which depend only on the properties of ϕ and χ and the values of $\sup_{\boldsymbol{\theta}}\{P(\boldsymbol{\theta}^\top \boldsymbol{x} = 0) : \boldsymbol{\theta} \neq 0\}$ and $\sup_{\alpha,\boldsymbol{\theta}}\{P(\alpha y + \boldsymbol{\theta}^\top \boldsymbol{x} = 0) : |\alpha| + \|\boldsymbol{\theta}\| \neq 0\}$. Uniqueness requires additional strong assumptions such as either symmetry or the existence of a density for the conditional distribution of $y - \boldsymbol{\theta}_0^\top \boldsymbol{x}$ for each fixed $\boldsymbol{x}$. Huber (1981) considers the minimization problem

$$(\beta^*, \sigma^*) = \operatorname{argmin}\left(\int \rho\left(\left(y - \boldsymbol{x}^\top \beta\right)/\sigma\right)\, dP(\boldsymbol{x}, y) + a\right)\sigma, \tag{25.108}$$

where $\rho : \mathbb{R} \to \mathbb{R}_+$ is convex with $\rho(0) = 0$ and $a > 0$. Under appropriate conditions on ρ it can be shown that the solution is unique and that there exists a convergent algorithm to calculate it. On differentiating (25.108) we obtain (25.106) and (25.107) with

$$\phi(\boldsymbol{x}, u) = \rho^{(1)}(u) \text{ and } \chi(u) = u\rho^{(1)}(u) - \rho(u) - a\,. \tag{25.109}$$

Even if the solution of (25.106) and (25.107) exists and is unique it is not necessarily regression equivariant. To make it so we must introduce a scatter functional T_Σ on the marginal distributions P', $P'(B) = P(B \times \mathbb{R})$ of the covariate $\boldsymbol{x}$. Such a functional satisfies $T_\Sigma(P'^A) = AT_\Sigma(P')A^\top$ for any non-singular $k \times k$-matrix A and is required not only for equivariance reasons but also to downweight outlying $\boldsymbol{x}$-values or so called leverage points. For this latter purpose the functional T_Σ must also be robust. We now replace (25.106) by

$$\int \phi\left(\boldsymbol{x}^\top T_\Sigma(P)^{-1}\boldsymbol{x}, \left(y - \boldsymbol{x}^\top\beta\right)/\sigma\right)\boldsymbol{x}\, dP(\boldsymbol{x}, y) = 0\,. \tag{25.110}$$

The resulting functional is now regression equivariant but its analysis is more difficult requiring as it does an analysis of the robustness properties of the scatter functional T_Σ.

Finally we note that in the literature most ϕ functions of (25.106) are of the form

$$\phi(\boldsymbol{x}, u) = \pi(\boldsymbol{x})\psi(u) \tag{25.111}$$

and the resulting functionals are known as GM-functionals. We refer to Hampel et al. (1986).

25.4.3 Bias and Breakdown

Given a regression functional $T_r = (T_b, T_s)$ where T_b refers to the β-components and T_s is the scale part it is usual to define breakdown just by the behaviour of T_b and to neglect T_s. The breakdown point of T_r at the distribution P is defined by

$$\epsilon^*(T_r, P, d_{\mathcal{H}}) = \sup\{\epsilon : b(T_r, P, \epsilon, d_{\mathcal{H}}) < \infty\} \tag{25.112}$$

where

$$b(T_r, P, \epsilon, d_{\mathcal{H}}) = \sup\{\|T_b(Q) - T_b(P)\| : d_{\mathcal{H}}(P, Q) < \epsilon\} \tag{25.113}$$

with corresponding definitions for the gross error neighbourhood $\epsilon^*(T_r, P, GE)$ and for the finite sample breakdown point $\epsilon^*(T_r, P_n, fsbp)$. To state the next theorem we set

$$\Delta(P) = \sup\{P(H \times \mathbb{R}) : H \text{ a plane in } \mathbb{R}^k \text{ of dimension at most } k-1\}\,,$$

which is the regression equivalent of (25.77). We have

Theorem 5. *For any regression equivariant functional*

$$\epsilon^*(T_r, P, d_h) \leq (1-\Delta(P))/2 \text{ and } \epsilon^*(T_r, P_n, fsbp) \leq (1-\Delta(P_n))/2\,. \quad (25.114)$$

If one considers L_1-regression

$$\beta^* = \text{argmin} \sum_{i=1}^{n} \left| y_i - \boldsymbol{x}_i^\top \beta \right| \quad (25.115)$$

it can be shown if one $\boldsymbol{x}_i$ is sufficiently outlying then the residual at this point will be zero and hence the finite sample breakdown point is a disappointing $1/n$. This turns out to apply to most M-functionals of the last section whose breakdown point is at most $1/(k+1)$ irrespective of their exact definition. The literature on this point is unsatisfactory. Although some M-functionals have been shown to have a positive breakdown point this has only been done under the assumption that the scale part T_s is known. As obtaining the correct magnitude of the scale of the errors is in some sense the most difficult problem in robust regression such results are of limited value. They do not however alter the fact that M-functionals have a disappointing breakdown point. We now turn to the problem of constructing high breakdown regression functionals.

25.4.4 High Breakdown Regression Functionals

The first high breakdown regression functional was proposed by Hampel (1975) and is as follows.

$$T_{lms}(P) = \text{argmin}_{(\beta,\sigma)} \left\{ \sigma : \int \left\{ |y - \boldsymbol{x}^\top \beta| \leq \sigma \right\} dP(\boldsymbol{x}, y) \geq 1/2 \right\}. \quad (25.116)$$

The idea goes back to Tukey's shortest half-sample of which it is the regression counter part. It can be shown that it has almost the highest finite sample breakdown point given by Theorem 5. By slightly altering the factor $1/2$ in (25.116) to take into account the dimension k of the $\boldsymbol{x}$-variables it can attain this bound. Rousseeuw (1984) propagated its use and gave it the name by which it is now known, the least median of squares LMS. Rousseeuw calculated the finite sample breakdown point and provided a first heuristic algorithm which could be applied to real data sets. He also defined a second high breakdown functional known as least trimmed squares LTS defined by

$$T_{lts}(P) = \text{argmin}_{(\beta,\sigma)} \left\{ \int \left(y - \boldsymbol{x}^\top \beta \right)^2 \left\{ |y - \boldsymbol{x}^\top \beta| \leq \sigma \right\} dP(\boldsymbol{x}, y) : \right.$$
$$\left. \int \left\{ |y - \boldsymbol{x}^\top \beta| \leq \sigma \right\} dP(\boldsymbol{x}, y) \geq 1/2 \right\}. \quad (25.117)$$

There are now many high breakdown regression functionals such as S-functionals (Rousseeuw and Yohai 1984), MM-functionals (Yohai 1987), τ-functionals (Yohai and Zamar 1988), constrained M-functionals (Mendes and Tyler 1996), rank regression (Chang et al. 1999) and regression depth (Rousseeuw and Hubert 1999). Just as in the location and scale problem in $\mathbb{R}^k$ statistical functionals can have the same breakdown points but very different bias functions. We refer to Martin et al. (1989), Maronna and Yohai (1993) and Berrendero and Zamar (2001). All these high breakdown functionals either attain or by some minor adjustment can be made to attain the breakdown points of Theorem 5 with the exception of depth based methods where the maximal breakdown point is $1/3$ (see Donoho and Gasko 1992).

All the above high breakdown regressional functionals can be shown to exist under weak assumptions but just as in the case of high breakdown location and scatter functionals in $\mathbb{R}^k$ uniqueness can only be shown under very strong conditions which typically involve the existence of a density function for the errors (see Davies 1993). The comments made about high breakdown location and scale functionals in $\mathbb{R}^k$ apply here. Thus even if a regression functional is well defined at some particular model there will be other models arbitrarily close in the metric $d_{\mathcal{H}}$ where a unique solution does not exist. This points to an inherent local instability of high breakdown regression functionals which has been noted in the literature (Ellis 1998; Sheather et al. 1997). Dietel (1993) has constructed regression functionals which are well defined at all models P with $\Delta(P) < 1$ and which are locally uniformly Lipschitz, not however locally uniformly Fréchet differentiable. For this reason all confidence regions and efficiency claims must be treated with a degree of caution. An increase in stability can however be attained by using the LTS-functional instead of the LMS-functional, by reweighting the observations or using some form of one-step M-functional improvement as in (25.29).

Just as with high breakdown location and scatter functionals in $\mathbb{R}^k$ the calculation of high breakdown regression functionals poses considerable difficulties. The first high breakdown regression functional was Hampel's least median of squares and even in the simplest case of a straight line in $\mathbb{R}^2$ the computational cost is of order n^2. The algorithm is by no means simple requiring as it does ideas from computational geometry (see Edelsbrunner and Souvaine 1990). From this and the fact that the computational complexity increases with dimension it follows that one has to fall back on heuristic algorithms. The one recommended for linear regression is that of Rousseeuw and Van Driesen (1999) for the LTS-functional.

25.4.5 Outliers

To apply the concept of α-outlier regions to the linear regression model we have to specify the distribution P_Y of the response and the joint distribution $P_{\boldsymbol{X}}$ of the regressors assuming them to be random. For specificness we consider the model

$$P_{Y|\boldsymbol{X}=\boldsymbol{x}} = N(\boldsymbol{x}^\top \beta, \sigma^2), \tag{25.118}$$

and

$$P_X = \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}). \tag{25.119}$$

Assumption (25.118) states that the conditional distribution of the response given the regressors is normal and assumption (25.119) means that the joint distribution of the regressors is a certain p-variate normal distribution. If both assumptions are fulfilled then the joint distribution of $(Y, \boldsymbol{X})$ is a multivariate normal distribution.

We can define outlier regions under model (25.101) in several reasonable ways. If only (25.118) is assumed then a response-α-outlier region could be defined as

$$\text{out}(\alpha, P_{Y|\boldsymbol{X}=\boldsymbol{x}}) = \left\{y \in \mathbb{R}\colon\ u = |y - \boldsymbol{x}^\top \beta| > \sigma\, z_{1-\alpha/2}\right\}, \tag{25.120}$$

which is appropriate if the regressors are fixed and only outliers in y-direction are to be identified. If the regressors are random, which will be the more frequent case in actuarial or econometric applications, outliers in $\boldsymbol{x}$-direction are important as well. Under assumption (25.119) a regressor-α-outlier region is a special case of the α-outlier region (25.99). This approach leads to a population based version of the concept of leverage points. These are the points in a sample $(y_i, \boldsymbol{x}_i)$, $i = 1, \ldots, n$, from model (25.101) "for which $\boldsymbol{x}_i$ is far away from the bulk of the $\boldsymbol{x}_i$ in the data" (Rousseeuw and van Zoomeren 1990).

For the identification of regressor-outliers (leverage points) the same identification rules can be applied as in the multivariate normal situation. For the detection of response-outliers by resistant one-step identifiers, one needs robust estimators of the regression coefficients and the scale σ. Examples of high breakdown estimators that can be used in this context are the Least Trimmed Squares estimator and the corresponding scale estimator (Rousseeuw 1984; Rousseeuw and Leroy 1987), S-estimators (Rousseeuw and Yohai 1984), MM-estimators (Yohai 1987) or the REWLS-estimators (Gervini and Yohai 2002).

25.5 Analysis of Variance

25.5.1 One-way Table

The one-way analysis of variance is concerned with the comparison of the locations of k samples x_{ij}, $j = 1, \ldots, n_i$, $i = 1, \ldots, k$. The term "analysis of variance" goes back to the pioneering work of Fisher (1935) who decomposed the variance of the combined samples as follows

$$\sum_{ij}(x_{ij} - \bar{\boldsymbol{x}})^2 = \sum_i \sum_j (x_{ij} - \bar{\boldsymbol{x}}_i)^2 + \sum_i n_i (\bar{\boldsymbol{x}}_i - \bar{\boldsymbol{x}})^2 . \tag{25.121}$$

The first term of (25.121) is the total sum of squares, the second is the sum of squares within samples and the third is the sum of squares between samples. If the data are modelled as i.i.d. normal random variables with a common variance σ^2 but with the

ith sample mean μ_i then it is possible to derive a test for the null hypothesis that the means are equal. The single hypothesis of equal means is rarely of interest in itself. All pairwise comparisons

$$\mu_i = \mu_l, \quad 1 \le i < l \le k\,,$$

as well as contrasts $\sum_i c_i \mu_i = 0$ may also be of interest and give rise to the problem of multiple testing and the associated difficulties. The use of the L_2-norm as in (25.121) is widespread perhaps because of the elegant mathematics. The peculiarities of data analysis must however have priority over mathematical theory and as real data sets may contain outliers, be skewed to some extent and have different scales it becomes clear that an L_2-norm and Gaussian based theory is of limited applicability. We sketch a robustified approach to the one-way table (see Davies 2004).

As a first step gross outliers are eliminated from each sample using a simplified version of the outlier identification rule based on the median and MAD of the sample. Using the robust location and scale functionals T_l and T_s an α_k confidence or approximation interval I_i for location for the ith sample is calculated. To control the error rate for Gaussian and other samples we set $\alpha_k = \alpha^{1/k}$ with for example $\alpha = 0.95$. This choice guarantees that for Gaussian samples

$$P(\mu_i \in I_i, i = 1, \ldots, k) = \alpha\,. \tag{25.122}$$

Simulations show that this holds accurately for other symmetric distributions such as the slash, Cauchy and the double exponential. All questions relating to the locations of the samples are now reduced to questions concerning the intervals. For example, the samples i and l can be approximated by the same location value if and only if $I_i \cap I_l \neq \emptyset$. Similarly if the samples are in some order derived from a covariable it may be of interest as to whether the locations can be taken to be non-decreasing. This will be the case if and only if there exist $a_i, i = 1, \ldots, k$ with $a_1 \le a_2 \le \ldots \le a_k$ and $a_i \in I_i$ for each i. Because of (25.122) all such questions when stated in terms of the μ_i can be tested simultaneously and on Gaussian test beds the error rate will be $1-\alpha$ regardless of the number of tests. Another advantage of the method is that it allows a graphical representation. Every analysis should include a plot of the boxplots for the k data sets. This can be augmented by the corresponding plot of the intervals I_i which will often look like the boxplots but if the sample sizes differ greatly this will influence the lengths of the intervals but not the form of the boxplots.

25.5.2 Two-Way Table

Given IJ samples

$$\left(x_{ijk}\right)_{k=1}^{n_{ij}}, \quad i = 1, \ldots, I,\ j = 1, \ldots, J$$

the two-way analysis of variance in its simplest version looks for a decomposition of the data of the form

$$x_{ijk} = m + a_i + b_j + c_{ij} + r_{ijk} \tag{25.123}$$

with the the following interpretation. The overall effect is represented by m, the row and column effects by the a_i and b_j respectively and the interactions by the c_{ij}. The residuals r_{ijk} take care of the rest. As it stands the decomposition (25.123) is not unique but can be made so by imposing side conditions on the a_i, b_j and the c_{ij}. Typically these are of the form

$$\sum_i a_i = \sum_j b_j = \sum_i c_{ij} = \sum_j c_{ij} = 0\,, \tag{25.124}$$

where the latter two hold for all j and i respectively. The conditions (25.124) are almost always stated as technical conditions required to make the decomposition (25.123) identifiable. The impression is given that they are neutral with respect to any form of data analysis. But this is not the case as demonstrated by Tukey (1993) and as can be seen by considering the restrictions on the interactions c_{ij}. The minimum number of interactions for which the restrictions hold is four which, in particular, excludes the case of a single interaction in one cell. The restrictions on the row and column effects can also be criticized but we take this no further than mentioning that the restrictions

$$\mathrm{MED}(a_1, \ldots, a_I) = \mathrm{MED}(b_1, \ldots, b_J) = 0 \tag{25.125}$$

may be more appropriate. The following robustification of the two-way table is based on Terbeck and Davies (1998). The idea is to look for a decomposition which minimizes the number of non-zero interactions. We consider firstly the case of one observation per cell, $n_{ij} = 1$, for all i and j, and look for a decomposition

$$x_{ij} = m + a_i + b_j + c_{ij} \tag{25.126}$$

with the smallest number of c_{ij} which are non-zero. We denote the positions of the c_{ij} by a $I \times J$-matrix C with $C(i, j) = 1$ if and only if $c_{ij} \neq 0$, the remaining entries being zero. It can be shown that for certain matrices C the non-zero interactions c_{ij} can be recovered whatever their values and, moreover, they are the unique non-zero residuals of the L_1-minimization problem

$$\min_{a_i, b_j} \sum_{ij} |x_{ij} - a_i - b_j|\,. \tag{25.127}$$

We call matrices C for which this holds unconditionally identifiable. They can be characterized and two such matrices are

$$\begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix} \quad \begin{pmatrix} 1 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{25.128}$$

as well as matrices obtained from any permutations of rows and columns. The above considerations apply to exact models without noise. It can be shown however that the results hold true if noise is added in the sense that for unconditionally identifiable matrices sufficiently large (compared to the noise) interactions c_{ij} can be identified as the large residuals from an L_1-fit. Three further comments are in order. Firstly Tukey's median polish can often identify interactions in the two-way-table. This is because it attempts to approximate the L_1-solution. At each step the L_1-norm is reduced or at least not increased but unfortunately the median polish may not converge and, even if it does, it may not reach the L_1 solution. Secondly L_1 solutions in the presence of noise are not unique. This can be overcome by approximating the moduls function $|x|$ by a strictly convex function almost linear in the tails. Thirdly, if there is more than one observation per cell it is recommended that they are replaced by the median and the method applied to the medians. Finally we point out that an interaction can also be an outlier. There is no a priori way of distinguishing the two.

References

Adrover, J.: Minimax bias-robust estimation of the dispersion matrix of multivariate distributions. Ann. Stat. **26**, 2301–2320 (1998)

Andrews, D.F., Bickel, P.J., Hampel, F.R., Rogers, W.H., Tukey, J.W.: Robust Estimates of Location: Survey and Advances. Princeton University Press, Princeton, NJ (1972)

Atkinson, A.C.: Fast very robust methods for the detection of multiple outliers. J. Am. Stat. Assoc. **89**, 1329–1339 (1994)

Barme-Delcroix, M.-F., Gather, U.: An isobar-surfaces approach to multidimensional outlier-proneness. Technical Report 20, Sonderforschungsbereich 475, University of Dortmund, Dortmund, Germany (2000)

Barnett, V., Lewis, T.: *Outliers in Statistical Data*. (3rd edn.), Wiley, New York, (1994)

Bartlett, M.S.: The effect of non-normality on the t-distribution. Proc. Camb. Phil. Soc. **31**, 223–231 (1935)

Becker, C., Gather, U.: The masking breakdown point of multivariate outlier identification rules. J. Am. Stat. Assoc. 94:947–955 (1999)

Becker, C., Gather, U.: The largest nonidentifiable outlier: a comparison of multivariate simultaneous outlier identification rules. Comput. Stat. Data Anal. **36**, 119–127 (2001)

Bednarski, T.: Fréchet differentiability and robust estimation. In: Mandl, P., Husková, M. (eds) Asymptotic Statistics: Proceedings of the Fifth Prague Symposium, Springer Lecture Notes, pp. 49–58. Springer (1993)

Bednarski, T., Clarke, B.R.: On locally uniform expansions of regular functionals. Discussiones Mathematicae Algebra Stoch. Meth. **18**, 155–165 (1998)

Bednarski, T., Clarke, B.R., Kolkiewicz, W.: Statistical expansions and locally uniform Fréchet differentiability. J. Aust. Math. Soc. **50**, 88–97 (1991)

Bernholt, T., Fischer, P.: The complexity of computing the mcd-estimator. Technical Report 45, Sonderforschungsbereich 475, University of Dortmund, Dortmund, Germany (2001)
Berrendero, J.R., Zamar, R.H.: Maximum bias curves for robust regression with non-elliptical regressors. Ann. Stat. **29**, 224–251 (2001)
Box, G.E.P.: Non-normality and test on variance. Biometrika **40**, 318–335 (1953)
Box, G.E.P., Andersen, S.L.: Permutation theory in the derivation of robust criteria and the study of departures from assumption. J. Roy. Stat. Soc. B **17**, 1–34 (1955)
Caroni, C., Prescott, P.: Sequential application of wilk's multivariate outlier test. Appl. Stat. **41**, 355–364 (1992)
Chang, H., McKean, J.W., Narjano, J.D., Sheather, S.J.: High-breakdown rank regression. J. Am. Stat. Assoc. 94(445):205–219 (1999)
Clarke, B.R.: Uniqueness and Fréchet differentiability of functional solutions to maximum likelihood type equations. Ann. Stat. **11**, 1196–1205 (1983)
Cohen, M.: The background of configural polysampling: a historical perspective. In: Morgenthaler, S., Tukey, J.W. (eds) Configural Polysampling: A Route to Practical Robustness, Chap. 2. Wiley, New York (1991)
Croux, C., Dehon, C.: Robust linear discriminant analysis using S-estimators. Can. J. Stat. **29**, 473–492 (2001)
Croux, C., Haesbroeck, G.: Principal components analysis based on robust estimators of the covariance or correlation matrix: influence functions and efficiencies. Biometrika **87**, 603–618 (2000)
Croux, C., Rousseeuw, P.J.: Time-efficient algorithms for two highly robust estimators of scale. In: Dodge, Y., Whittaker, J.C. (eds) Computational Statistics, vol. 1, pp. 411–428. Physica, Heidelberg (1992)
Davies, P.L.: Asymptotic behaviour of S-estimates of multivariate location parameters and dispersion matrices. Ann. Stat. **15**, 1269–1292 (1987)
Davies, P.L.: The asymptotics of Rousseeuw's minimum volume ellipsoid. Ann. Stat. **20**, 1828–1843 (1992a)
Davies, P.L.: Aspects of robust linear regression. Ann. Stat. **21**, 1843–1899 (1993)
Davies, P.L.: Data features. Stat. Neerl. **49**, 185–245 (1995)
Davies, P.L.: On locally uniformly linearizable high breakdown location and scale functionals. Ann. Stat. **26**, 1103–1125 (1998)
Davies, P.L.: The one-way table. J. Stat. Plann. Infer. **122**, 3–13 (2004)
Davies, P.L., Gather, U.: The identification of multiple outliers (with discussion). J. Am. Stat. Assoc. **88**, 782–801 (1993)
Dietel, G.: Global location and dispersion functionals. PhD thesis, University of Essen (1993)
Donoho, D.L.: Breakdown properties of multivariate location estimators. PhD thesis, Department of Statistics, Harvard University, Harvard, Mass (1982)
Donoho, D.L., Gasko, M.: Breakdown properties of location estimates based on halfspace depth and project outlyingness. Ann. Stat. **20**, 1803–1827 (1992)
Donoho, D.L., Huber, P.J.: The notion of breakdown point. In: Bickel, P.J., Doksum, K.A., Hodges, J.L. Jr. (eds) A Festschrift for Erich L. Lehmann, pp. 157–184, Wadsworth, Belmont, California (1983)
Eddington, A.S.: Stellar Movements and the Structure of the Universe. Macmillan, New York (1914)
Edelsbrunner, H., Souvaine, D.: Computing median-of-squares regression lines and guided topological sweep. J. Am. Stat. Assoc. **85**, 115–119 (1990)
Ellis, S.P.: Instability of least squares, least absolute deviation and least median of squares linear regression. Stat. Sci. **13**(4), 337–350 (1998)
Fernholz, L.T.: Von Mises Calculus for Statistical Functionals. Number 19 in Lecture Notes in Statistics. Springer, New York (1983)
Fisher, R.A.: A mathematical examination of the methods of determining the accuracy of an observation by the mean error and the mean square error. Mon. Not. Roy. Astron. Soc. **80**, 758–770 (1920)

Fisher, R.A.: The Design of Experiments, Oliver and Boyd, Edinburgh and London (1935)
Gather, U.: Modelling the occurrence of multiple outliers. All. Stat. Arch. **74**, 413–428 (1990)
Gather, U., Hilker, T.: A note on tyler's modification of the MAD for the stahel-donoho estimator. Ann. Stat. **25**, 2024–2026 (1997)
Gather, U., Kuhnt, S., Pawlitschko, J.: Concepts of outlyingness for various data structures. In: Misra, J.C. (eds.) Industrial Mathematics and Statistics. pp. 545–585. Narosa Publishing House, New Delhi (2003)
Gather, U., Schultze, V.: Robust estimation of scale of an exponential distribution. Stat. Neerl. **53**, 327–341 (1999)
Gayen, A.K.: The distribution of the variance ratio in random samples of any size drawn from non-normal universe. Biometrika **37**, 236–255 (1950)
Geary, R.C.: The distribution of 'student's' ratio for non-normal samples. J. Roy. Stat. Soc. Suppl. **3**, 178–184 (1936)
Geary, R.C.: Testing for normality. Biometrika **34**, 209–242 (1947)
Gervini, D., Yohai, V.J.: A class of robust and fully efficient regression estimators. Ann. Stat. **30**(2), 583–616 (2002)
Gnanadesikan, R., Kettenring, J.R.: Robust estimates, residuals, and outlier detection with multiresponse data. Biometrics **28**, 81–124 (1972)
Hadi, A.S.: A modification of a method for the detection of outliers in multivariate samples. J. Roy. Stat. Soc. B **56**, 393–396 (1994)
Hadi, A.S., Simonoff, J.S.: Procedures for the identification of multiple outliers in linear models. J. Am. Stat. Assoc. **88**, 1264–1272 (1997)
Hampel, F.R.: Contributions to the theory of robust estimation. PhD thesis, University of California, Berkeley (1968)
Hampel, F.R.: Beyond location parameters: Robust concepts and methods (with discussion). In: Proceedings of the 40th Session of the ISI, vol. 46, Book 1, pp. 375–391 (1975)
Hampel, F.R.: The breakdown points of the mean combined with some rejection rules. Technometrics **27**, 95–107 (1985)
Hampel, F.R., Ronchetti, E.M., Rousseeuw, P.J., Stahel, W.A.: Robust Statistics: The Approach Based on Influence Functions, Wiley, New York (1986)
Hawkins, D.M.: Identification of outliers, Chapman and Hall, London (1980)
Huber, P.J.: Robust estimation of a location parameter. Ann. Math. Stat. **35**, 73–101 (1964)
Huber, P.J.: Robust statistical procedures. In: Regional Conference Series in Applied Mathematics No. 27, Society for Industrial and Applied Mathematics, Philadelphia, Penn (1977)
Huber, P.J.: Robust Statistics, Wiley, New York (1981)
Huber, P.J.: Finite sample breakdown points of m- and p-estimators. Ann. Stat. **12**, 119–126 (1984)
Huber, P.J.: Robustness: Where are we now? Student **1**, 75–86 (1995)
Kent, J.T., Tyler, D.E.: Redescending M-estimates of multivariate location and scatter. Ann. Stat. **19**, 2102–2119 (1991)
Kent, J.T., Tyler, D.E.: Constrained M-estimation for multivariate location and scatter. Ann. Stat. **24**, 1346–137 (1996)
Liu, R.Y., Parelius, J.M., Singh, K.: Multivariate analysis by data depth: descriptive statistics, graphics and inference. Ann. Stat. **27**, 783–840 (1999)
Lopuhaä, H.P.: On the relation between S-estimators and M-estimators of multivariate location and covariance. Ann. Stat. **19**, 229–248 (1989)
Lopuhaä, H.P.: Multivariate τ-estimators for location and scatter. Can. J. Stat. **19**, 307–321 (1991)
Lopuhaä, H.P., Rousseeuw, P.J.: Breakdown properties of affine equivariant estimators of multivariate location and covariance matrices. Ann. Stat. **19**, 229–248 (1991)
Marazzi, A.: Algorithms, Routines, and S- Functions for Robust Statistics. Chapman and Hall, New York (1992)
Maronna, R.A.: Robust M-estimators of multivariate location and scatter. Ann. Stat. **4**(1), 51–67 (1976)
Maronna, R.A., Stahel, W.A., Yohai, V.J.: Bias-robust estimators of multivariate scatter based on projections. J. Multivariate Anal. **42**, 141–161 (1992)

Maronna, R.A., Yohai, V.J.: Asymptotic behavior of general M-estimates for regression and scale with random carriers. Zeitschrift für Wahrscheinlichkeitstheorie und verwandte Gebiete **58**, 7–20 (1981)
Maronna, R.A., Yohai, V.J.: Bias-robust estimates of regression based on projections. Ann. Stat. **21**(2), 965–990 (1993)
Martin, R.D., Yohai, V.J., Zamar, R.H.: Min-max bias robust regression. Ann. Stat. **17**(4), 1608–1630 (1989)
Martin, R.D., Zamar, R.H.: Bias robust estimation of scale. Ann. Stat. **21**(2), 991–1017 (1993a)
Martin, R.D., Zamar, R.H.: Efficiency constrained bias robust estimation of location. Ann. Stat. **21**(1), 338-354 (1993b)
Mendes, B., Tyler, D.E.: Constrained M-estimates for regression. In: Rieder, H. (eds.) Robust Statistics; Data Analysis and Computer Intensive Methods, number 109 in Lecture Notes in Statistics, pp. 299–320. Springer, New York (1996)
Neyman, J., Pearson, E.S.: On the problem of the most efficient tests of statistical hypotheses. Phil. Trans. Roy. Soc. (London) A **231**, 289–337 (1933)
Pearson, E.S.: The distribution of frequency constants in small samples from non-normal symmetrical and skew populations. Biometrika **21**, 259–286 (1929)
Pearson, E.S.: The analysis of variance in cases of non-normal variation. Biometrika **23**, 114–133 (1931)
Pearson, E.S., Chandra Sekar, S.: The efficiency of statistical tools and a criterion for the rejection of outlying observations. Biometrika **28**, 308–320 (1936)
Pollard, D.: Convergence of Stochastic Processes. Springer, New York (1984)
Riedel, M.: On the bias-robustness in the location model i. Statistics **2**, 223–233 (1989a)
Riedel, M.: On the bias-robustness in the location model ii. Statistics **2**, 235–246 (1989b)
Rieder, H.: Robust Asymptotic Statistics. Springer, Berlin (1994)
Rocke, D.M.: Robustness properties of S-estimators of multivariate location and shape in high dimension. Ann. Stat. **24**, 1327–1345 (1996)
Rocke, D.M., Woodruff, D.L.: Identification of outliers in multivariate data. J. Am. Stat. Assoc. **91**(435), 1047–1061 (1996)
Rocke, D.M., Woodruff, D.L.: Robust estimation of multivariate location and shape. J. Stat. Plann. Infer. **91**, 245–255 (1997)
Rosner, B.: On the detection of many outliers. Technometrics **17**, 221–227 (1975)
Rousseeuw, P.J.: Least median of squares regression. J. Am. Stat. Assoc. **79**, 871–880 (1984)
Rousseeuw, P.J.: Multivariate estimation with high breakdown point. In: Grossmann, W., Pflug, C.G., Vincze, I., Wertz, W. (eds) Mathematical Statistics and Applications (Proceedings of the 4th Pannonian Symposium on Mathematical Statistics), vol. B, Reidel, Dordrecht (1985)
Rousseeuw, P.J., Croux, C.: Explicit scale estimators with high breakdown point. In: Dodge, Y. (eds.) L_1-Statistical Analysis and Related Methods, pp. 77–92, North Holland, Amsterdam (1992)
Rousseeuw, P.J., Croux, C.: Alternatives to the median absolute deviation. J. Am. Stat. Assoc. **88**, 1273–1283 (1993)
Rousseeuw, P.J., Croux, C.: The bias of k-step M-estimators. Stat. Probab. Lett. **20**, 411–420 (1994)
Rousseeuw, P.J., Hubert, M.: Regression depth. J. Am. Stat. Assoc. **94**, 388–402 (1999)
Rousseeuw, P.J., Leroy, A.M.: Robust Regression and Outlier Detection, Wiley, New York (1987)
Rousseeuw, P.J., Van Driesen, K.: A fast algorithm for the minimum covariance determinant estimator. Technometrics **41**, 212–223 (1999)
Rousseeuw, P.J., Van Driesen, K.: An algorithm for positive-breakdown methods based on concentration steps. In: Gaul, W., Opitz, O., Schader, M. (eds.) Data Analysis: Scientific modelling and Practical Application, pp. 335–346. Springer, New York (2000)
Rousseeuw, P.J., van Zoomeren, B.C.: Unmasking multivariate outliers and leverage points. J. Am. Stat. Assoc. **85**, 633–639 (1990)
Rousseeuw, P.J., Yohai, V.J.: Robust regression by means of S-estimators. In: Franke, J.E.A. (eds.) Robust and Nonlinear Time Series Analysis, pp. 256–272, New York. Springer (1984)

Scholz, F.W.: *Comparison of Optimal Location Estimators*. PhD thesis, Department of Statistics, University of California, Berkley (1971)
Sheather, S.J., McKean, J.W., Hettmansperger, T.P.: Finite sample stability properties of the least median of squares estimator. J. Stat. Comput. Simul. **58**(4), 371–383 (1997)
Simonoff, J.S.: A comparison of robust methods and detection of outlier techniques when estimating a location parameter. Comm. Stat. A **13**, 813–842 (1984)
Simonoff, J.S.: The breakdown and influence properties of outlier rejection-plus-mean procedures. Comm. Stat. A **16**, 1749–1760 (1987)
Stahel, W.A.: Breakdown of covariance estimators. Research Report 31, Fachgruppe für Statistik, ETH, Zurich (1981)
Staudte, R.G., Sheather, S.J.: *Robust Estimation and Testing*, Wiley, New York (1990)
Tatsuoka, K.S., Tyler, D.E.: On the uniqueness of S-functionals and M-functionals under non-elliptic distributions. Ann. Stat. **28**(4), 1219–1243 (2000)
Terbeck, W., Davies, P.L.: Interactions and outliers in the two-way analysis of variance. Ann. Stat. **26**, 1279–1305 (1998)
Tietjen, G.L., Moore, R.H.: Some grubbs-type statistics for the detection of several outliers. Technometrics **14**, 583–597 (1972)
Tukey, J.W.: A survey of sampling from contaminated distributions. In: Olkin, I. (eds.) Contributions to Probability and Statistics. Stanford University Press, Stanford, California (1960)
Tukey, J.W.: Mathematics and picturing data. In: Proceedings of International Congress of Mathematicians, Vancouver, vol. 2, pp. 523–531 (1975)
Tukey, J.W.: Exploratory analysis of variance as providing examples of strategic choices. In: Morgenthaler, S., Ronchetti, E., Stahel, W.A. (eds.) New Directions in Statistical Data Analysis and Robustness, Birkhäuser, Basel (1993)
Tyler, D.E.: Finite sample breakdown points of projection based multivariate location and scatter statistics. Ann. Stat. **22**, 1024–1044 (1994)
von Mises, R.: Sur les fonctions statistiques. In: Conférence de la Réunion Internationale des Mathï£¡maticiens. Gauthier-Villars (1937)
Willems, S., Pison, G., Rousseeuw, P.J., Van Aelst, S.: A robust Hotelling test. Metrika **55**, 125–138 (2002)
Yohai, V.J.: High breakdown point and high efficiency robust estimates for regression. Ann. Stat. **15**, 642–656 (1987)
Yohai, V.J., Maronna, R.A.: The maximum bias of robust covariances. Comm. Stat. Theor. Meth. **19**, 3925–3933 (1990)
Yohai, V.J., Zamar, R.H.: High breakdown point estimates of regression by means of the minimization of an efficient scale. J. Am. Stat. Assoc. **83**, 406–413 (1988)
Zuo, Y.: Some quantitative relationships between two types of finite sample breakdown points. Stat. Probab. Lett. **51**, 369–375 (2001)
Zuo, Y., Serfling, R.: General notions of statistical depth function. Ann. Stat. **28**, 461–482 (2000a)
Zuo, Y., Serfling, R.: Structural properties and convergence results for contours of sample statistical depth functions. Ann. Stat. **28**, 483–499 (2000b)

Chapter 26
Bayesian Computational Methods

Christian P. Robert

26.1 Introduction

If, in the mid 1980s, one had asked the average statistician about the difficulties of using Bayesian Statistics, the most likely answer would have been "Well, there is this problem of selecting a prior distribution and then, even if one agrees on the prior, the whole Bayesian inference is simply impossible to implement in practice!" The same question asked in the Twenty first Century does not produce the same reply, but rather a much less aggressive complaint about the lack of generic software (besides winBUGS), along with the renewed worry of subjectively selecting a prior! The last 20 years have indeed witnessed a tremendous change in the way Bayesian Statistics are perceived, both by mathematical statisticians and by applied statisticians and the impetus behind this change has been a prodigious leap-forward in the computational abilities. The availability of very powerful approximation methods has correlatively freed Bayesian modelling, in terms of both model scope *and* prior modelling. This opening has induced many more scientists from outside the statistics community to opt for a Bayesian perspective as they can now handle those tools on their own. As discussed below, a most successful illustration of this gained freedom can be seen in Bayesian model choice, which was only emerging at the beginning of the MCMC era, for lack of appropriate computational tools.

In this chapter, we will first present the most standard computational challenges met in Bayesian Statistics (Sect. 26.2), and then relate these problems with computational solutions. Of course, this chapter is only a terse introduction to the problems and solutions related to Bayesian computations. For more complete references, see Robert and Casella (2004), Marin and Robert (2007a), Robert and Casella (2004) and Liu (2001), among others. We also restrain from providing an introduction to

C.P. Robert (✉)
Université Paris-Dauphine, CEREMADE, and CREST-INSEE, Paris, France
e-mail: Christian.Robert@ceremade.dauphine.fr

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_26,

Bayesian Statistics *per se* and for comprehensive coverage, address the reader to Marin and Robert (2007a) and Robert (2007), (again) among others.

26.2 Bayesian Computational Challenges

Bayesian Statistics being a complete inferential methodology, its scope encompasses the whole range of standard statistician inference (and design), from point estimation to testing, to model selection, and to non-parametrics. In principle, once a prior distribution has been chosen on the proper space, the whole inferential machinery is set and the computation of estimators is usually automatically derived from this setup. Obviously, the practical or numerical derivation of these procedures may be exceedingly difficult or even impossible, as we will see in a few selected examples. Before, we proceed with an incomplete typology of the categories and difficulties met by Bayesian inference. First, let us point out that computational difficulties may originate from one or several of the following items:

(1) Use of a complex parameter space, as for instance in constrained parameter sets like those resulting from imposing stationarity constraints in dynamic models.
(2) Use of a complex sampling model with an intractable likelihood, as for instance in missing data and graphical models.
(3) Use of a huge dataset.
(4) Use of a complex prior distribution (which may be the posterior distribution associated with an earlier sample).
(5) Use of a complex inferential procedure.

26.2.1 Bayesian Point Estimation

In a formalised representation of Bayesian inference, the statistician is given (or selects) a triplet:

- A sampling distribution, $f(x|\boldsymbol{\theta})$, usually associated with an observation (or a sample) x.
- A prior distribution $\pi(\boldsymbol{\theta})$, defined on the parameter space Θ.
- A loss function $\mathrm{L}(\boldsymbol{\theta}, d)$ that compares the decisions (or estimations) d for the true value $\boldsymbol{\theta}$ of the parameter.

Using (f, π, L) and an observation x, the Bayesian inference is *always* given as the solution to the minimisation programme

$$\min_d \int_\Theta L(\boldsymbol{\theta}, d)\, f(x|\boldsymbol{\theta})\, \pi(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta}\,,$$

equivalent to the minimisation programme

$$\min_d \int_\Theta L(\boldsymbol{\theta}, d)\, \pi(\boldsymbol{\theta}|x)\, \mathrm{d}\boldsymbol{\theta}\,.$$

The corresponding procedure is thus associated, for every x, to the solution of the above programme (see, e.g. Robert 2007, Chap. 2).

There are therefore two levels of computational difficulties with this resolution: first the above integral must be computed. Second, it must be minimised in d. For the most standard losses, like the traditional squared error loss,

$$\mathrm{L}(\boldsymbol{\theta}, d) = |\boldsymbol{\theta} - d\,|^2\,,$$

the solution to the minimisation problem is universally[1] known. For instance, for the squared error loss, it is the posterior mean,

$$\int_\Theta \boldsymbol{\theta}\, \pi(\boldsymbol{\theta}|x)\, \mathrm{d}\boldsymbol{\theta} = \int_\Theta \boldsymbol{\theta}\; f(x|\boldsymbol{\theta})\, \pi(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta} \Big/ \int_\Theta f(x|\boldsymbol{\theta})\, \pi(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta}\,,$$

which still requires the computation of both integrals and thus whose complexity depends on the complexity of Θ, $f(x|\boldsymbol{\theta})$, and $\pi(\boldsymbol{\theta})$.

Example 1. For a normal distribution $\mathcal{N}(\boldsymbol{\theta}, 1)$, the use of a so-called conjugate prior (see, e.g., Robert 2007, Chap. 3)

$$\boldsymbol{\theta} \sim \mathcal{N}(\mu, \epsilon)\,,$$

leads to a closed form expression for the mean, since

$$\begin{aligned}
&\int_\Theta \boldsymbol{\theta}\; f(x|\boldsymbol{\theta})\, \pi(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta} \Big/ \int_\Theta f(x|\boldsymbol{\theta})\, \pi(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta} \\
&\quad = \int_{\mathbb{R}} \boldsymbol{\theta} \exp \frac{1}{2} \left\{-\boldsymbol{\theta}^2(1+\epsilon^{-2}) + 2\boldsymbol{\theta}(x+\epsilon^{-2}\mu)\right\} \mathrm{d}\boldsymbol{\theta} \\
&\qquad \times \Big/ \int_{\mathbb{R}} \exp \frac{1}{2} \left\{-\boldsymbol{\theta}^2(1+\epsilon^{-2}) + 2\boldsymbol{\theta}(x+\epsilon^{-2}\mu)\right\} \mathrm{d}\boldsymbol{\theta} = \frac{x+\epsilon^{-2}\mu}{1+\epsilon^{-2}}\,.
\end{aligned}$$

On the other hand, if we use instead a more involved prior distribution like a poly-t distribution (Bauwens and Richard 1985),

$$\pi(\boldsymbol{\theta}) = \prod_{i=1}^{k} \left[\alpha_i + (\boldsymbol{\theta} - \beta_i)^2\right]^{-\nu_i} \qquad \alpha, \nu > 0$$

[1] In this chapter, the denomination *universal* is used in the sense of *uniformly over all distributions.*

the above integrals cannot be computed in closed form anymore. This is *not* a toy example in that the problem may occur after a sequence of k Student's t observations, or with a sequence of normal observations whose variance is unknown.

The above example is one-dimensional, but, obviously, bigger challenges await the Bayesian statistician when she wants to tackle high-dimensional problems.

Example 2. In a generalised linear model, a conditional distribution of $y \in \mathbb{R}$ given $x \in \mathbb{R}^p$ is defined via a density from an exponential family

$$y|x \sim \exp\{y \cdot \boldsymbol{\theta}(x) - \psi(\boldsymbol{\theta}(x))\}$$

whose natural parameter $\boldsymbol{\theta}(x)$ depends on the conditioning variable x,

$$\boldsymbol{\theta}(x) = g(\beta^{\mathrm{T}} x)\,, \qquad \beta \in \mathbb{R}^p$$

that is, linearly modulo the transform g. Obviously, in practical applications like Econometrics, p can be quite large. Inference on β (which is the true parameter of the model) proceeds through the posterior distribution (where $\mathbf{x} = (x_1, \ldots, x_T)$ and $\mathbf{y} = (y_1, \ldots, y_T)$)

$$\begin{aligned}\pi(\beta|\mathbf{x}, \mathbf{y}) &\propto \prod_{t=1}^{T} \exp\{y_t \cdot \boldsymbol{\theta}(x_t) - \psi(\boldsymbol{\theta}(x_t))\}\ \pi(\beta) \\ &= \exp\left\{\sum_{t=1}^{T} y_t \cdot \boldsymbol{\theta}(x_t) - \sum_{t=1}^{T} \psi(\boldsymbol{\theta}(x_t))\right\}\ \pi(\beta)\,,\end{aligned}$$

which rarely is available in closed form. In addition, in some cases ψ may be costly simply to compute and in others T may be large or even very large. Take for instance the case of the dataset processed by Abowd et al. (1999), which covers twenty years of employment histories for over a million workers, with x including indicator variables for over one hundred thousand companies.

Complexity starts sharply increasing if we introduce in addition random effects to the model, that is writing $\boldsymbol{\theta}(x)$ as $g(\beta^{\mathrm{T}} x + \epsilon(x))$, where $\epsilon(x)$ is a random perturbation indexed by x. For instance, in the above employment dataset, it may correspond to a worker effect or to a company effect. The difficulty is that those random variables can very rarely be integrated out into a closed-form marginal distribution. They must therefore be included within the model parameter, which then increases its dimension severalfold.

A related, although conceptually different, inferential issue concentrates upon *prediction*, that is, the approximation of a distribution related with the parameter of interest, say $g(y|\boldsymbol{\theta})$, based on the observation of $x \sim f(x|\boldsymbol{\theta})$. The *predictive distribution* is then defined as

$$\pi(y|x) = \int_{\Theta} g(y|\boldsymbol{\theta})\pi(\boldsymbol{\theta}|x)\mathrm{d}\boldsymbol{\theta}\,.$$

A first difference with the standard point estimation perspective is obviously that the parameter $\boldsymbol{\theta}$ vanishes through the integration. A second and more profound difference is that this parameter is not necessarily well-defined anymore. As will become clearer in a following Section, this is a paramount feature in setups where the model is not well-defined and where the statistician hesitates between several (or even an infinity of) models. It is also a case where the standard notion of identifiability is irrelevant, which paradoxically is a "plus" from the computational point of view, as seen below in, e.g., Example 13.

Example 3. Recall that an $AR(p)$ model is given as the *auto-regressive* representation of a time series,

$$x_t = \sum_{i=1}^{p} \boldsymbol{\theta}_i x_{t-i} + \sigma\varepsilon_t\,.$$

It is often the case that the order p of the AR model is not fixed *a priori*, but has to be determined from the data itself. Several models are then competing for the "best" fit of the data, but if the prediction of the next value x_{t+1} is the most important part of the inference, the order p chosen for the best fit is not really relevant. Therefore, all models can be considered in parallel and aggregated through the predictive distribution

$$\pi(x_{t+1}|x_t,\ldots,x_1) \propto \int f(x_{t+1}|x_t,\ldots,x_{t-p+1})\pi(\boldsymbol{\theta}, p|x_t,\ldots,x_1)\mathrm{d}p\,\mathrm{d}\boldsymbol{\theta}\,,$$

which thus amounts to integrating over the parameters of all models, simultaneously:

$$\sum_{p=0}^{\infty} \int f(x_{t+1}|x_t,\ldots,x_{t-p+1})\pi(\boldsymbol{\theta}|p,x_t,\ldots,x_1)\,\mathrm{d}\boldsymbol{\theta}\,\pi(p|x_t,\ldots,x_1)\,.$$

Note the multiple layers of complexity in this case:

(1) If the prior distribution on p has an infinite support, the integral simultaneously considers an infinity of models, with parameters of unbounded dimensions.
(2) The parameter $\boldsymbol{\theta}$ varies from model $AR(p)$ to model $AR(p+1)$, so must be evaluated differently from one model to another. In particular, if the stationarity constraint usually imposed in these models is taken into account, the constraint on $(\boldsymbol{\theta}_1,\ldots,\boldsymbol{\theta}_p)$ varies[2] between model $AR(p)$ and model $AR(p+1)$.

[2]To impose the stationarity constraint when the order of the $AR(p)$ model varies, it is necessary to reparameterise this model in terms of either the partial autocorrelations or of the roots of the associated lag polynomial. (See, e.g., Robert 2007, Sect. 4.5.)

(3) Prediction is usually used sequentially: every tick/second/hour/day, the next value is predicted based on the past values $x_t, \ldots, x_1$). Therefore when t moves to $t+1$, the entire posterior distribution $\pi(\boldsymbol{\theta}, p|x_t, \ldots, x_1)$ must be re-evaluated again, possibly with a very tight time constraint as for instance in financial or radar tracking applications.

We will discuss this important problem in deeper details after the testing section, as part of the model selection problematic.

26.2.2 Testing Hypotheses

A domain where both the philosophy and the implementation of Bayesian inference are at complete odds with the classical approach is the area of testing of hypotheses. At a primary level, this is obvious when opposing the Bayesian evaluation of an hypothesis $H_0 : \boldsymbol{\theta} \in \Theta_0$

$$\Pr^{\pi}(\boldsymbol{\theta} \in \Theta_0|x)$$

with a Neyman–Pearson p-value

$$\sup_{\boldsymbol{\theta}\in\Theta_0} \Pr_{\boldsymbol{\theta}}(T(X) \geq T(x))$$

where T is an appropriate statistic, with observed value $T(x)$. The first quantity involves an integral over the *parameter* space, while the second provides an evaluation over the *observational* space. At a secondary level, the two answers may also strongly disagree even when the number of observations goes to infinity, although there exist cases and priors for which they agree to the order $\mathrm{O}(n^{-1})$ or even $\mathrm{O}(n^{-3/2})$. (See Robert 2007, Sect. 3.5.5 and Chap. 5, for more details.)

From a computational point of view, most Bayesian evaluations of the relevance of an hypothesis – also called the *evidence* – given a sample x involve marginal distributions

$$\int_{\Theta_i} f(x|\boldsymbol{\theta}_i)\pi_i(\boldsymbol{\theta}_i)\,\mathrm{d}\boldsymbol{\theta}_i \tag{26.1}$$

where Θ_i and π_i denote the parameter space and the corresponding prior, respectively, under hypothesis H_i $(i = 0, 1)$. For instance, the *Bayes factor* is defined as the ratio of the posterior probabilities of the null and the alternative hypotheses over the ratio of the prior probabilities of the null and the alternative hypotheses, i.e.,

$$B_{01}^{\pi}(x) = \frac{\Pr(\boldsymbol{\theta} \in \Theta_0 \mid x)}{\Pr(\boldsymbol{\theta} \in \Theta_1 \mid x)} \Big/ \frac{\pi(\boldsymbol{\theta} \in \Theta_0)}{\pi(\boldsymbol{\theta} \in \Theta_1)}.$$

This quantity is instrumental in the computation of the posterior probability

$$\Pr(\boldsymbol{\theta} \in \Theta_0 \mid x) = \frac{1}{1 + B_{10}^{\pi}(x)}$$

under equal prior probabilities for both Θ_0 and Θ_1. It is also the central tool in practical (as opposed to decisional) Bayesian testing (Jeffreys 1961) and can be seen as the Bayesian equivalent of the likelihood ratio.

The first ratio in $B_{01}^{\pi}(x)$ is then the ratio of integrals of the form (26.1) and it is rather common to face difficulties in the computation of *both* integrals.[3]

Example 4 (Continuation of Example 2). In the case of the generalised linear model, a standard testing situation is to decide whether or not a factor, x_1 say, is influential on the dependent variable y. This is often translated as testing whether or not the corresponding component of β, β_1, is *equal* to 0, i.e. $\Theta_0 = \{\beta; \beta_1 = 0\}$. If we denote by β_{-1} the *other* components of β, the Bayes factor for this hypothesis will be

$$\int_{\mathbb{R}^p} \exp\left\{\sum_{t=1}^{T} y_t \cdot g(\beta^{\mathrm{T}} x_t) - \sum_{t=1}^{T} \psi(g(\beta^{\mathrm{T}} x_t))\right\} \pi(\beta)\,\mathrm{d}\beta \Big/$$

$$\int_{\mathbb{R}^{p-1}} \exp\left\{\sum_{t=1}^{T} y_t \cdot g(\beta_{-1}^{\mathrm{T}}(x_t)_{-1}) - \sum_{t=1}^{T} \psi(\beta_{-1}^{\mathrm{T}}(x_t)_{-1})\right\} \pi_{-1}(\beta_{-1})\,\mathrm{d}\beta_{-1}\,,$$

when π_{-1} is the prior constructed for the null hypothesis and when the prior weights of H_0 and of the alternative are both equal to $1/2$. Obviously, besides the normal conjugate case, both integrals cannot be computed in a closed form.

In a related manner, *confidence regions* are also mostly intractable, being defined through the solution to an implicit equation. Indeed, the Bayesian confidence region for a parameter $\boldsymbol{\theta}$ is defined as the *highest posterior region*,

$$\{\boldsymbol{\theta}; \pi(\boldsymbol{\theta}|x) \geq k(x)\} \tag{26.2}$$

where $k(x)$ is determined by the coverage constraint

$$\Pr^{\pi}(\pi(\boldsymbol{\theta}|x) \geq k(x)|x) = \alpha\,,$$

α being the confidence level. While the normalising constant is not necessary to construct a confidence region, the resolution of the implicit equation (26.2) is rarely straightforward! However, simulation-based equivalents generally produce

[3] In this presentation of Bayes factors, we completely bypass the methodological difficulty of defining $\pi(\boldsymbol{\theta} \in \Theta_0)$ when Θ_0 is of measure 0 for the original prior π and refer the reader to Robert (2007, Sect. 5.2.3) and Marin and Robert (2007, Sect. 2.3.2) for proper coverage of this issue.

acceptable approximations in a straightforward manner when both the prior and the likelihood can be numerically computed.

Example 5. Consider a binomial observation $x \sim \mathcal{B}(n, \theta)$ with a conjugate prior distribution, $\theta \sim \mathcal{B}e(\gamma_1, \gamma_2)$. In this case, the posterior distribution is available in closed form,

$$\theta|x \sim \mathcal{B}e(\gamma_1 + x, \gamma_2 + n - x)\,.$$

However, the determination of the θ's such that

$$\theta^{\gamma_1+x-1}(1-\theta)^{\gamma_2+n-x-1} \geq k(x)$$

with

$$\Pr^{\pi}\left(\theta^{\gamma_1+x-1}(1-\theta)^{\gamma_2+n-x-1} \geq k(x)|x\right) = \alpha$$

is not possible analytically. It actually implies two levels of numerical difficulties:

1. Find the solution(s) to $\theta^{\gamma_1+x-1}(1-\theta)^{\gamma_2+n-x-1} = k$.
2. Find the k corresponding to the right coverage.

and each value of k examined in step 2. requires a new resolution of step 1. However, k can also be interpreted as the $(1-\alpha)$ quantile of the random variable $\theta^{\gamma_1+x-1}(1-\theta)^{\gamma_2+n-x-1}$, hence derived from a large sample of θ's in a Monte Carlo perspective.

The setting is usually much more complex when θ is a multidimensional parameter, because the interest is usually in getting marginal confidence sets. Example 2 is an illustration of this setting: deriving a confidence region on one component, β_1 say, first involves computing the marginal posterior distribution of this component. As in Example 4, the integral

$$\int_{\mathbb{R}^{p-1}} \exp\left\{\sum_{t=1}^{T} y_t \cdot g(\beta^{\mathrm{T}} x_t) - \sum_{t=1}^{T} \psi(\beta^{\mathrm{T}} x_t)\right\} \pi_{-1}(\beta_{-1})\,\mathrm{d}\beta_{-1}\,,$$

which is proportional to $\pi(\beta_1|x)$, is most often intractable. Fortunately, the simulation approximations mentioned above are also available to bypass this integral computation.

26.2.3 Model Choice

Although they are essentially identical from a conceptual viewpoint, we do distinguish here between *model choice* and testing, partly because the former leads to further computational difficulties, and partly because it encompasses a larger scope of inferential goals than mere testing. Note first that model choice has been the subject of considerable effort in the past decades, and has seen many advances, including the coverage of problems of higher complexity and the introduction of

new concepts. We stress that such advances mostly owe to the introduction of new computational methods.

As discussed in further details in Robert (2007, Chap. 7), the inferential action related with model choice does take place on a wider scale than simple testing: it covers and compares models, rather than parameters, which makes the sampling distribution $f(x)$ "more unknown" than simply depending on an undetermined parameter. In some respect, it is thus closer to estimation than to regular testing. In any case, it requires a more precise evaluation of the consequences of choosing the "wrong" model or, equivalently of deciding which model is the most appropriate for the data at hand. It is thus both broader and less definitive than deciding whether $H_0 : \boldsymbol{\theta}_1 = 0$ is true. At last, the larger inferential scope mentioned in the first point means that we are leaving for a while the well-charted domain of solid parametric models.

From a computational point of view, model choice involves more complex structures that, almost systematically, require advanced tools, like simulation methods which can handle collections of parameter spaces (also called *spaces of varying dimensions*), specially designed for model comparison.

Example 6. A mixture of distributions is the representation of a distribution (density) as the weighted sum of standard distributions (densities). For instance, a mixture of Poisson distributions, denoted as

$$\sum_{i=1}^{k} p_i \mathcal{P}(\lambda_i)$$

has the following density:

$$\Pr(X = k) = \sum_{i=1}^{k} p_i \, \frac{\lambda_i^k}{k!} \, e^{-\lambda_i} \, .$$

This representation of distributions is multi-faceted and can be used in populations with known heterogeneities (in which case a component of the mixture corresponds to an homogeneous part of the population) as well as a non-parametric modelling of unknown populations. This means that, in some cases, k is known and, in others, it is both unknown and part of the inferential problem.

First, consider the setting where several (parametric) models are in competition,

$$\mathfrak{M}_i : x \sim f_i(x|\boldsymbol{\theta}_i), \qquad \boldsymbol{\theta}_i \in \Theta_i, \quad i \in I \, ,$$

the index set I being possibly infinite. From a Bayesian point of view, a prior distribution must be constructed for each model $\mathfrak{M}_i$ as if it were the only and true model under consideration since, in most perspectives except *model averaging*, *only one* of these models will be selected and used as the only and true model. The parameter space associated with the above set of models can be written as

$$\hat{} = \bigcup_{i \in I} \{i\} \times \Theta_i \,, \tag{26.3}$$

the model indicator $\mu \in I$ being now part of the parameters. So, if the modeller allocates probabilities p_i to the indicator values, that is, to the models $\mathfrak{M}_i$ $(i \in I)$, and if she then defines priors $\pi_i(\boldsymbol{\theta}_i)$ on the parameter subspaces Θ_i, things fold over by virtue of Bayes's theorem, since one can compute

$$p(\mathfrak{M}_i|x) = \Pr(\mu = i|x) = \frac{p_i \displaystyle\int_{\Theta_i} f_i(x|\boldsymbol{\theta}_i)\pi_i(\boldsymbol{\theta}_i)\mathrm{d}\boldsymbol{\theta}_i}{\displaystyle\sum_j p_j \int_{\Theta_j} f_j(x|\boldsymbol{\theta}_j)\pi_j(\boldsymbol{\theta}_j)\mathrm{d}\boldsymbol{\theta}_j} \,.$$

While a common solution based on this prior modelling is simply to take the (marginal) MAP estimator of μ, that is, to determine the model with the largest $p(\mathfrak{M}_i|x)$, or even to use directly the average

$$\sum_j p_j \int_{\Theta_j} f_j(y|\boldsymbol{\theta}_j)\pi_j(\boldsymbol{\theta}_j|x)\mathrm{d}\boldsymbol{\theta}_j = \sum_j p(\mathfrak{M}_j|x)\, m_j(y)$$

as a predictive density in y in *model averaging*, a deeper-decision theoretic evaluation is often necessary.

Example 7 (Continuation of Example 3). In the setting of the $AR(p)$ models, when the order p of the dependence is unknown, model averaging as presented in Example 3 is not always a relevant solution when the statistician wants to estimate this order p for different purposes. Estimation is then a more appropriate perspective than testing, even though care must be taken because of the discrete nature of p. (For instance, the posterior expectation of p is not an appropriate estimator!)

As stressed earlier in this Section, the computation of predictive densities, marginals, Bayes factors, and other quantities related to the model choice procedures is generally very involved, with specificities that call for tailor-made solutions:

- The computation of integrals is increased by a factor corresponding to the number of models under consideration.
- Some parameter spaces are infinite-dimensional, as in non-parametric settings and that may cause measure-theoretic complications.
- The computation of posterior or predictive quantities involves integration over different parameter spaces and thus increases the computational burden, since there is no time savings from one subspace to the next.
- In some settings, the size of the collection of models is very large or even infinite and some models cannot be explored. For instance, in Example 4, the collection of all submodels is of size 2^p and some pruning method must be found in variable selection to avoid exploring the whole tree of all submodels.

26.3 Monte Carlo Methods

The natural approach to these computational problems is to use computer simulation and Monte Carlo techniques, rather than numerical methods, simply because there is much more to gain from exploiting the probabilistic properties of the integrands rather than their analytical properties. In addition, the dimension of most problems considered in current Bayesian Statistics is such that very involved numerical methods should be used to provide a satisfactory approximation in such integration or optimisation problems. Indeed, down-the-shelf numerical methods cannot handle integrals in moderate dimensions and more advanced numerical integration methods require analytical studies on the distribution of interest.

26.3.1 Preamble: Monte Carlo Importance Sampling

Given the statistical nature of the problem, the approximation of an integral like

$$\mathfrak{I} = \int_{\Theta} h(\boldsymbol{\theta})\, f(x|\boldsymbol{\theta})\pi(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta},$$

should indeed take advantage of the special nature of $\mathfrak{I}$, namely, the fact that π is a probability density[4] or, instead, that $f(x|\boldsymbol{\theta})\pi(\boldsymbol{\theta})$ is proportional to a density. As detailed in Chap. II.2 this volume, or in Robert and Casella (2004, Chap. 3) and Robert and Casella (2010), the *Monte Carlo method* was introduced by Metropolis and Ulam (1949) for this purpose. For instance, if it is possible to generate (via a computer) random variables $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_m$ from $\pi(\boldsymbol{\theta})$, the average

$$\frac{1}{m} \sum_{i=1}^{m} h(\boldsymbol{\theta}_i)\, f(x|\boldsymbol{\theta}_i)$$

converges (almost surely) to $\mathfrak{I}$ when m goes to $+\infty$, according to the Law of Large Numbers. Obviously, if an i.i.d. sample of $\boldsymbol{\theta}_i$'s from the posterior distribution $\pi(\boldsymbol{\theta}|x)$ can be produced, the average

$$\frac{1}{m} \sum_{i=1}^{m} h(\boldsymbol{\theta}_i) \tag{26.4}$$

converges to

$$\mathbb{E}^{\pi}[h(\boldsymbol{\theta})|x] = \frac{\int_{\Theta} h(\boldsymbol{\theta})\, f(x|\boldsymbol{\theta})\pi(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta}}{\int_{\Theta} f(x|\boldsymbol{\theta})\pi(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta}}$$

[4] The prior distribution can be used for importance sampling only if it is a proper prior and not a σ-finite measure.

and it usually is more interesting to use this approximation, rather than

$$\sum_{i=1}^{m} h(\boldsymbol{\theta}_i) f(x|\boldsymbol{\theta}_i) \Big/ \sum_{i=1}^{m} f(x|\boldsymbol{\theta}_i)$$

when the $\boldsymbol{\theta}_i$'s are generated from $\pi(\boldsymbol{\theta})$, especially when $\pi(\boldsymbol{\theta})$ is flat compared with $\pi(\boldsymbol{\theta}|x)$.

In addition, if the posterior variance $\text{var}(h(\boldsymbol{\theta})|x)$ is finite, the Central Limit Theorem applies to the empirical average (26.4), which is then asymptotically normal with variance $\text{var}(h(\boldsymbol{\theta})|x)/m$. Confidence regions can then be built from this normal approximation and, most importantly, the magnitude of the error remains of order $1/\sqrt{m}$, whatever the dimension of the problem, in opposition with numerical methods.[5] (See also Robert and Casella 2004, 2009, Chap. 4, for more details on the convergence assessment based on the CLT.)

The Monte Carlo method actually applies in a much wider generality than the above simulation from π. For instance, because $\mathfrak{I}$ can be represented in an infinity of ways as an expectation, there is no need to simulate from the distributions $\pi(\cdot|x)$ or π to get a good approximation of $\mathfrak{I}$. Indeed, if g is a probability density with $\text{supp}(g)$ including the support of $|h(\boldsymbol{\theta})| f(x|\boldsymbol{\theta})\pi(\boldsymbol{\theta})$, the integral $\mathfrak{I}$ can also be represented as an expectation against g, namely

$$\int \frac{h(\boldsymbol{\theta}) f(x|\boldsymbol{\theta})\pi(\boldsymbol{\theta})}{g(\boldsymbol{\theta})} g(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta}.$$

This representation leads to the *Monte Carlo method with importance function* g: generate $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_m$ according to g and approximate $\mathfrak{I}$ through

$$\frac{1}{m} \sum_{i=1}^{m} h(\boldsymbol{\theta}_i)\omega_i(\boldsymbol{\theta}_i),$$

with the weights $\omega(\boldsymbol{\theta}_i) = f(x|\boldsymbol{\theta}_i)\pi(\boldsymbol{\theta}_i)/g(\boldsymbol{\theta}_i)$. Again, by the Law of Large Numbers, this approximation almost surely converges to $\mathfrak{I}$. And this estimator is unbiased. In addition, an approximation to $\mathbb{E}^{\pi}[h(\boldsymbol{\theta})|x]$ is given by

$$\frac{\sum_{i=1}^{m} h(\boldsymbol{\theta}_i)\omega(\boldsymbol{\theta}_i)}{\sum_{i=1}^{m} \omega(\boldsymbol{\theta}_i)}. \tag{26.5}$$

since the numerator and denominator converge to

[5] The constant order of the Monte Carlo error does not imply that the computational effort remains the same as the dimension increases, most obviously, but rather that the decrease (with m) in variation has the rate $1/\sqrt{m}$.

$$\int_{\Theta} h(\boldsymbol{\theta})\, f(x|\boldsymbol{\theta})\pi(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta} \qquad \text{and} \qquad \int_{\Theta} f(x|\boldsymbol{\theta})\pi(\boldsymbol{\theta})\, \mathrm{d}\boldsymbol{\theta},$$

respectively, if $\mathrm{supp}(g)$ includes $\mathrm{supp}(f(x|\cdot)\pi)$. Notice that the ratio (26.5) does not depend on the normalising constants in either $h(\boldsymbol{\theta})$, $f(x|\boldsymbol{\theta})$ or $\pi(\boldsymbol{\theta})$. The approximation (26.5) can therefore be used in settings when some of these normalising constants are unknown. Notice also that the *same* sample of $\boldsymbol{\theta}_i$'s can be used for the approximation of both the numerator and denominator integrals: even though using an estimator in the denominator creates a bias, (26.5) does converge to $\mathbb{E}^{\pi}[h(\boldsymbol{\theta})|x]$.

While this convergence is guaranteed for all densities g with wide enough support, the choice of the importance function is crucial. First, simulation from g must be easily implemented. Moreover, the function $g(\boldsymbol{\theta})$ must be close enough to the function $h(\boldsymbol{\theta})\pi(\boldsymbol{\theta}|x)$, in order to reduce the variability of (26.5) as much as possible; otherwise, most of the weights $\omega(\boldsymbol{\theta}_i)$ will be quite small and a few will be overly influential. In fact, if $\mathbb{E}^{h}[h^2(\boldsymbol{\theta})\omega^2(\boldsymbol{\theta})]$ is not finite, the variance of the estimator (26.5) is infinite (see Robert and Casella 2004, Chap. 3). Obviously, the dependence on g of the importance function h can be avoided by proposing generic choices such as the posterior distribution $\pi(\boldsymbol{\theta}|x)$.

26.3.2 First Illustrations

In either point estimation or simple testing situations, the computational problem is often expressed as a ratio of integrals. Let us start with a toy example to set up the way Monte Carlo methods proceed and highlight the difficulties of applying a generic approach to the problem.

Example 8. Consider a t-distribution $\mathcal{T}(\nu, \boldsymbol{\theta}, 1)$ sample $(x_1, \ldots, x_n)$ with ν known. Assume in addition a flat prior $\pi(\boldsymbol{\theta}) = 1$ as in a non-informative environment. While the posterior distribution on $\boldsymbol{\theta}$ can be easily plotted, up to a normalising constant (Fig. 26.1), because we are in dimension 1, direct simulation and computation from this posterior is impossible.

If the inferential problem is to decide about the value of $\boldsymbol{\theta}$, the posterior expectation is

$$\mathbb{E}^{\pi}[\boldsymbol{\theta}|x_1, \ldots, x_n] = \int \boldsymbol{\theta} \prod_{i=1}^{n} \left[\nu + (x_i - \boldsymbol{\theta})^2\right]^{-(\nu+1)/2} \mathrm{d}\boldsymbol{\theta}$$
$$\Big/ \int \prod_{i=1}^{n} \left[\nu + (x_i - \boldsymbol{\theta})^2\right]^{-(\nu+1)/2} \mathrm{d}\boldsymbol{\theta}\,.$$

This ratio of integrals is not directly computable. Since $(\nu + (x_i - \boldsymbol{\theta})^2)^{-(\nu+1)/2}$ is proportional to a t-distribution $\mathcal{T}(\nu, x_i, 1)$ density, a solution to the approximation

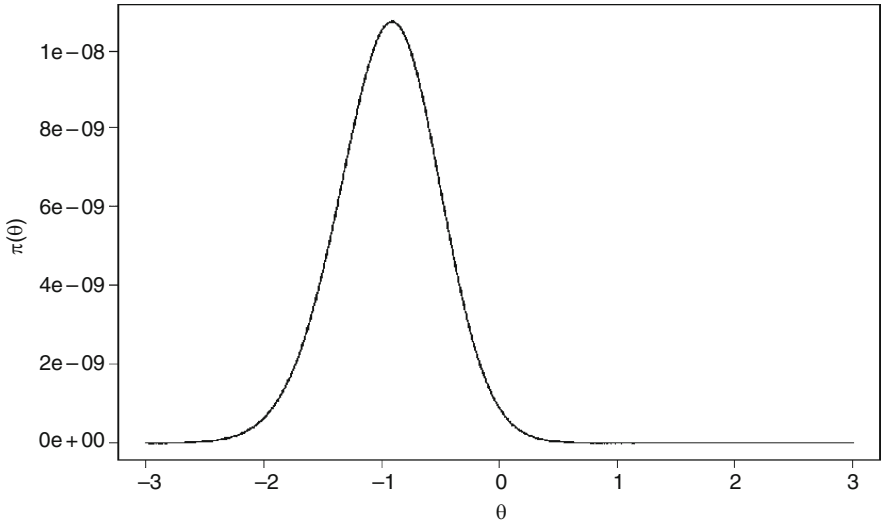


Fig. 26.1 Posterior density of $\boldsymbol{\theta}$ in the setting of Example 8 for $n = 10$, with a simulated sample from $\mathcal{T}(3, 0, 1)$

of the integrals is to use *one* of the i's to "be" the density in both integrals. For instance, if we generate $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_m$ from the $\mathcal{T}(\nu, x_1, 1)$ distribution, the equivalent of (26.5) is

$$\delta_m^\pi = \sum_{j=1}^{m} \boldsymbol{\theta}_j \prod_{i=2}^{n} \left[\nu + (x_i - \boldsymbol{\theta}_j)^2\right]^{-(\nu+1)/2} \Bigg/ \sum_{j=1}^{m} \prod_{i=2}^{n} \left[\nu + (x_i - \boldsymbol{\theta}_j)^2\right]^{-(\nu+1)/2} \tag{26.6}$$

since the first term in the product has been "used" for the simulation and the normalisation constants have vanished in the ratio. Figure 26.2 is an illustration of the speed of convergence of this estimator to the true posterior expectation: it provides the evolution of δ_m^π as a function of m both on average and on range (provided by repeated simulations of δ_m^π). As can be seen from the graph, the average is almost constant from the start, as it should, because of unbiasedness, while the range decreases very slowly, as it should, because of extreme value theory. The graph provides in addition the 90% empirical confidence interval built on these simulations.[6] The empirical confidence intervals are decreasing in $1/\sqrt{n}$, as expected from the theory. (This is further established by regressing the log-

[6]The empirical (Monte Carlo) confidence interval is not to be confused with the asymptotic confidence interval derived from the normal approximation. As discussed in Robert and Casella

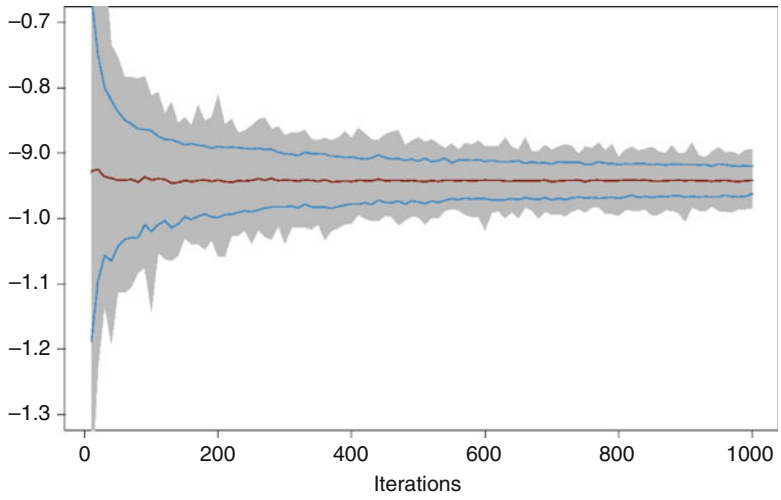


Fig. 26.2 Evolution of a sequence of 500 estimators (26.6) over 1,000 iterations: range *(in gray)*, 0.05 and 0.95 quantiles, and average, obtained on the same sample as in Fig. 26.1 when simulating from the t distribution with location x_1

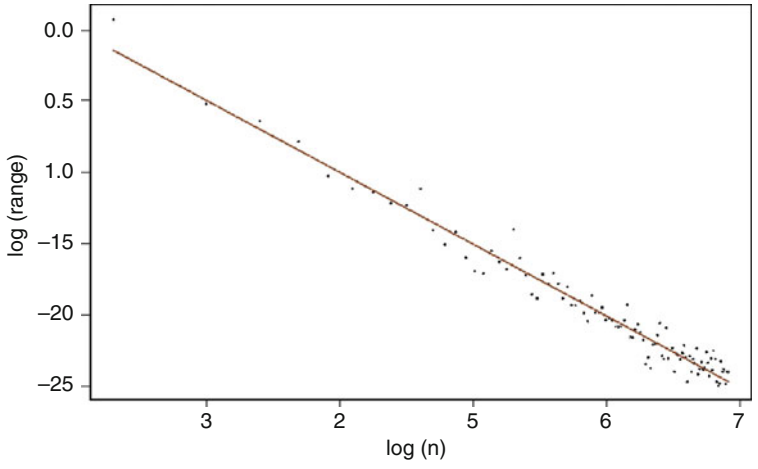


Fig. 26.3 Regression of the log-ranges *(left)* and the log-lengths of the confidence intervals *(right)* on $\log(n)$, for the output in Fig. 26.2

lengths of the confidence intervals on $\log(n)$, with slope equal to -0.5, as shown by Fig. 26.3.)

Now, there is a clear arbitrariness in the choice of x_1 in the sample $(x_1, \ldots, x_n)$ for the proposal $\mathcal{T}(\nu, x_1, 1)$. While any of the x_i's has the same theoretical validity

(2004, Chap. 4), these two intervals may differ considerably in width, with the interval derived from the CLT being much more optimistic!

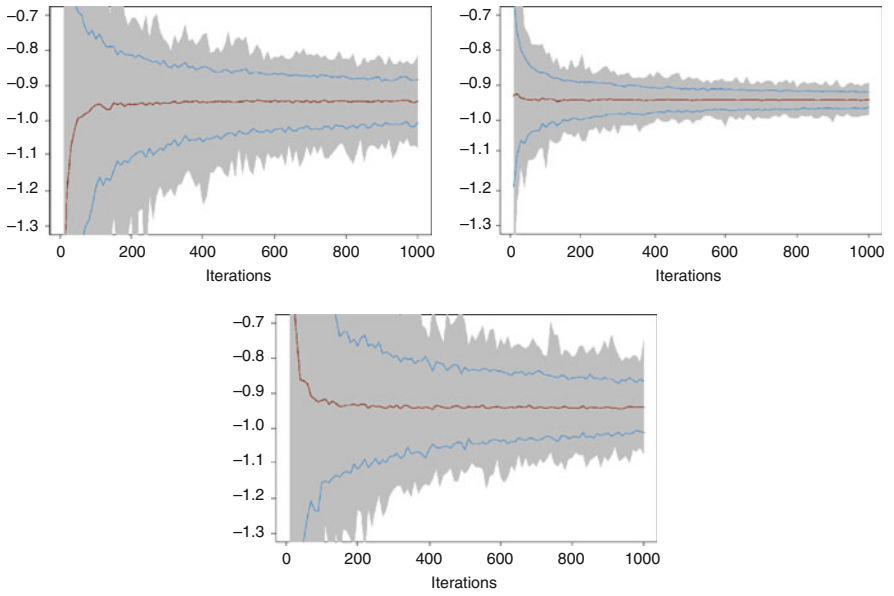


Fig. 26.4 Repetition of the experiment described in Fig. 26.2 for three different choices of x_i: $\min x_i$, $x_{(5)}$ and $\max x_i$ *(from left to right)*

to "represent" the integral and the integrating density, the choice of x_i's closer to the posterior mode (the true value of $\boldsymbol{\theta}$ is 0) induces less variability in the estimates, as shown by a further simulation experiment through Fig. 26.4. It is fairly clear from this comparison that the choice of extremal values like $x_{(1)} = -3.21$ and even more $x_{(10)} = 1.72$ is detrimental to the quality of the approximation, compared with the median $x_{(5)} = -0.86$. The range of the estimators is much wider for both extremes, but the influence of this choice is also visible for the average which does not converge so quickly.[7]

This example thus shows that Monte Carlo methods, while widely available, may easily face inefficiency problems when the simulated values are not sufficiently attuned to the distribution of interest. It also demonstrates that, fundamentally, there is no difference between importance sampling and regular Monte Carlo, in that the integral $\mathfrak{I}$ can naturally be represented in many ways.

Although we do not wish to spend too much space on this issue, let us note that the choice of the importance function gets paramount when the support of the function of interest is *not* the whole space. For instance, a tail probability, associated

[7] An alternative to the simulation from one $\mathcal{T}(\nu, x_i, 1)$ distribution that does not require an extensive study on the most appropriate x_i is to use a mixture of the $\mathcal{T}(\nu, x_i, 1)$ distributions. As seen in Sect. 26.5.2, the weights of this mixture can even be optimised automatically.

with $h(\boldsymbol{\theta}) = \mathbb{I}_{\boldsymbol{\theta} \geq \boldsymbol{\theta}_0}$ say, should be estimated with an importance function whose support is $[\boldsymbol{\theta}_0, \infty)$. (See Robert and Casella 2004, Chap. 3, for details.)

Example 9 (Continuation of Example 8). If, instead, we wish to consider the probability that $\boldsymbol{\theta} \geq 0$, using the t-distribution $\mathcal{T}(\nu, x_i, 1)$ is not a good idea because negative values of $\boldsymbol{\theta}$ are somehow simulated "for nothing". A better proposal (in terms of variance) is to use the "folded" t-distribution $\mathcal{T}(\nu, x_i, 1)$, with density proportional to

$$\psi_i(\boldsymbol{\theta}) = \left[\nu + (x_i - \boldsymbol{\theta})^2\right]^{-(\nu+1)/2} + \left[\nu + (x_i + \boldsymbol{\theta})^2\right]^{-(\nu+1)/2} ,$$

on $\mathbb{R}_+$, which can be simulated by taking the absolute value of a $\mathcal{T}(\nu, x_i, 1)$ rv. All simulated values are then positive and the estimator of the probability is

$$\rho_m^{\pi} = \sum_{j=1}^{m} \prod_{i \neq k} \left[\nu + (x_i - |\boldsymbol{\theta}_j|)^2\right]^{-(\nu+1)/2} / \psi_k(|\boldsymbol{\theta}_j|) \times \Big/ \sum_{j=1}^{m} \prod_{i \neq k} \left[\nu + (x_i - \boldsymbol{\theta}_j)^2\right]^{-(\nu+1)/2} \tag{26.7}$$

where the $\boldsymbol{\theta}_j$'s are iid $\mathcal{T}(\nu, x_k, 1)$. Note that this is a very special occurrence where the *same* sample can be used in both the numerator and the denominator. In fact, in most cases, two different samples have to be used, if only because the support of the importance distribution for the numerator is *not* the whole space, unless, of course, all normalising constants are known. Figure 26.5 reproduces earlier figures for this problem, when using $x_{(5)}$ as the parameter of the t distribution.

The above example is one-dimensional (in the parameter) and the problems exhibited there can be found severalfold in multidimensional settings. Indeed, while Monte Carlo methods do not suffer from the "curse of dimension" in the sense that the error of the corresponding estimators is always decreasing in $1/\sqrt{n}$, notwithstanding the dimension, it gets increasingly difficult to come up with satisfactory importance sampling distributions as the dimension gets higher and higher. As we will see in Sect. 26.5, the intuition built on MCMC methods has to be exploited to derive satisfactory importance functions.

Example 10 (Continuation of Example 2). A particular case of generalised linear model is the *probit model*,

$$\Pr_{\boldsymbol{\theta}}(Y = 1|x) = 1 - \Pr_{\boldsymbol{\theta}}(Y = 0|x) = \Phi(x^{\mathrm{T}}\boldsymbol{\theta}) \qquad \boldsymbol{\theta} \in \mathbb{R}^p ,$$

where Φ denotes the normal $\mathcal{N}(0, 1)$ cdf. Under a flat prior $\pi(\boldsymbol{\theta}) = 1$, for a sample $(x_1, y_1), \ldots, (x_n, y_n)$, the corresponding posterior distribution is proportional to

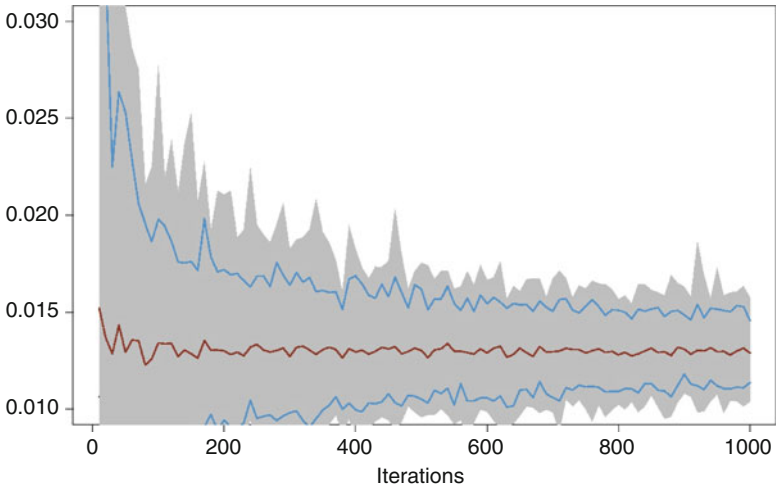


Fig. 26.5 Evolution of a sequence of 100 estimators (26.7) over 1,000 iterations *(same legend as Fig. 26.2)*

$$\prod_{i=1}^{n} \Phi(x_i^{\mathrm{T}}\boldsymbol{\theta})^{y_i}\, \Phi(-x_i^{\mathrm{T}}\boldsymbol{\theta})^{1-y_i}\,. \tag{26.8}$$

Direct simulation from this distribution is obviously impossible since the computation of $\Phi(z)$ is a difficulty in itself. If we pick an importance function for this problem, the adequation with the posterior distribution will need to be better and better as the dimension p increases. Otherwise, the repartition of the weights will get increasingly asymmetric: very few weights will be different from 0.

Figure 26.6 illustrates this degeneracy of the importance sampling approach as the dimension increases. We simulate parameters β's and datasets (x_i, y_i) $(i = 1, \ldots, 245)$ for dimensions p ranging from 1 to 10, then represented the histograms of the largest weight for $p = 1, 2, 5, 10$. The x_i's were simulated from a $\mathcal{N}_p(0, I_p)$ distribution, while the importance sampling distribution was a $\mathcal{N}_p(0, I_p/p)$ distribution.

26.3.3 Approximations of the Bayes Factor

As explained in Sects. 26.2.2 and 26.2.3, the first computational difficulty associated with Bayesian testing is the derivation of the *Bayes factor*, of the form

$$B_{12}^{\pi} = \frac{\displaystyle\int_{\Theta_1} f_1(x|\boldsymbol{\theta}_1)\pi_1(\boldsymbol{\theta}_1)\mathrm{d}\boldsymbol{\theta}_1}{\displaystyle\int_{\Theta_2} f_2(x|\boldsymbol{\theta}_2)\pi_2(\boldsymbol{\theta}_2)\mathrm{d}\boldsymbol{\theta}_2} = \frac{m_1(x)}{m_2(x)}\,,$$

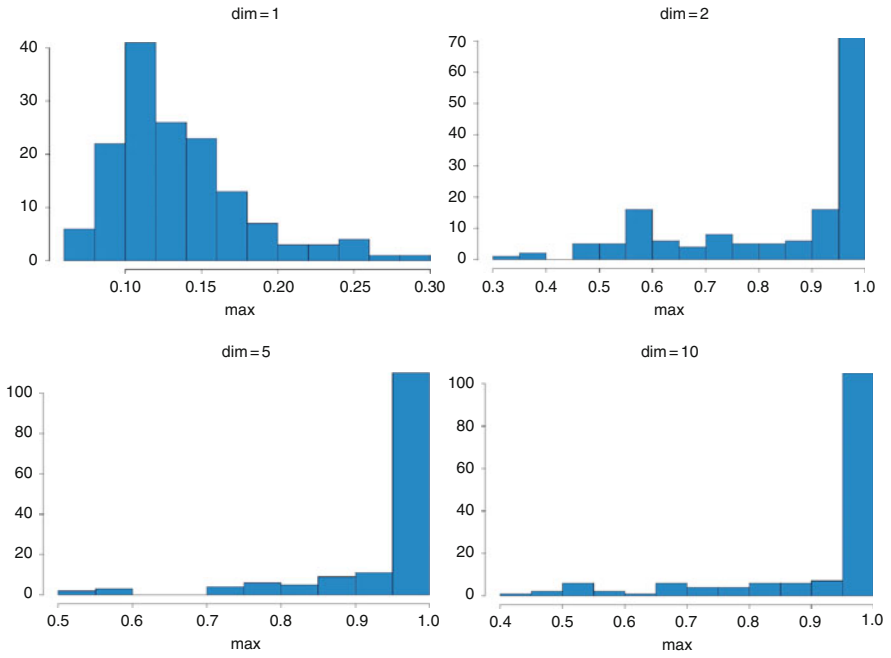


Fig. 26.6 Comparison of the distribution of the largest importance weight based upon 150 replications of an importance sampling experiment with 245 observations and dimensions $p = 1, 2, 5, 10$

where, for simplicity's sake, we have adopted a model choice perspective (that is, $\boldsymbol{\theta}_1$ and $\boldsymbol{\theta}_2$ may live in completely different spaces).

Specific Monte Carlo methods for the estimation of ratios of normalising constants, or, equivalently, of Bayes factors, have been developed in the past five years. See Chen et al. (2000, Chap. 5) for a complete exposition, as well as Chopin and Robert (2010) and Marin and Robert (2007b) for recent reassessments. In particular, the importance sampling technique is rather well-adapted to the computation of those Bayes factors: Given a importance distribution, with density proportional to g, and a sample $\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(T)}$ simulated from g, the marginal density for model $\mathfrak{M}_i$, $m_i(x)$, is approximated by

$$\widehat{m}_i(x) = \sum_{t=1}^{T} f_i(x|\boldsymbol{\theta}^{(t)}) \frac{\pi_i(\boldsymbol{\theta}^{(t)})}{g(\boldsymbol{\theta}^{(t)})} \Bigg/ \sum_{t=1}^{T} \frac{\pi_i(\boldsymbol{\theta}^{(t)})}{g(\boldsymbol{\theta}^{(t)})} ,$$

where the denominator takes care of the (possibly) missing normalising constants. (Notice that, if g is a density, the expectation of $\pi(\boldsymbol{\theta}^{(t)})/g(\boldsymbol{\theta}^{(t)})$ is 1 and the denominator should be replaced by T to decrease the variance of the estimator of $m_i(x)$.)

A compelling incentive, among others, for using importance sampling in the setting of model choice is that the sample $(\boldsymbol{\theta}^{(1)}, \ldots, \boldsymbol{\theta}^{(T)})$ can be recycled for all models $\mathfrak{M}_i$ sharing the same parameters (in the sense that the models $\mathfrak{M}_i$ are parameterised in the same way, e.g. by their first moments).

Example 11 (Continuation of Example 4). In the case the β's are simulated from a product instrumental distribution

$$g(\beta) = \prod_{i=1}^{p} g_i(\beta_i)\,,$$

the sample of β's produced for the general model of Example 2, $\mathfrak{M}_1$ say, can also be used for the restricted model, $\mathfrak{M}_2$, where $\beta_1 = 0$, simply by deleting the first component and keeping the following components, with the corresponding importance density being

$$g_{-1}(\beta) = \prod_{i=2}^{p} g_i(\beta_i)\,.$$

Once the β's have been simulated,[8] the Bayes factor B_{12}^{π} can be approximated by $\widehat{m}_1(x)/\widehat{m}_2(x)$.

However, the variance of $\widehat{m}(x)$ may be infinite, depending on the choice of g. A possible choice is $g(\boldsymbol{\theta}) = \pi(\boldsymbol{\theta})$, with wider tails than $\pi(\boldsymbol{\theta}|x)$, but this is often inefficient if the data is informative because the prior and the posterior distributions will be quite different and most of the simulated values $\boldsymbol{\theta}^{(t)}$ fall outside the modal region of the likelihood. (This is of course impossible when π is improper.) For the choice $g(\boldsymbol{\theta}) = f(x|\boldsymbol{\theta})\pi(\boldsymbol{\theta})$,

$$\widehat{m}(x) = 1 \Big/ \frac{1}{T}\sum_{t=1}^{T} \frac{1}{f(x|\boldsymbol{\theta}^{(t)})}\,, \tag{26.9}$$

is the *harmonic mean* of the likelihoods, but the corresponding variance is infinite when the likelihood has thinner tails than the prior (which is often the case). Having an infinite variance means that the numerical value produced by the estimate cannot be trusted at all, as it may be away from the true marginal by several orders of magnitude. As discussed in Chopin and Robert (2010) and Marin and Robert (2007b), it is actually possible to use a generalised harmonic mean estimate

$$\widehat{m}(x) = 1 \Big/ \frac{1}{T}\sum_{t=1}^{T} \frac{\varphi(\boldsymbol{\theta}^{(t)})}{\pi(\boldsymbol{\theta}^{(t)})\, f(x|\boldsymbol{\theta}^{(t)})}\,,$$

[8] We stress the point that this is mostly an academic exercise as, in regular settings, it is rarely the case that independent components are used for the importance function.

when φ is a probability density with tails that are thinner than the posterior tails. For instance, a density with support an approximate HPD region is quite appropriate.

Explicitly oriented towards the computation of ratios of normalising constants, *bridge sampling* was introduced in Meng and Wong (1996): if both models $\mathfrak{M}_1$ and $\mathfrak{M}_2$ cover the same parameter space Θ, if $\pi_1(\boldsymbol{\theta}|x) = c_1\tilde{\pi}_1(\boldsymbol{\theta}|x)$ and $\pi_2(\boldsymbol{\theta}|x) = c_2\tilde{\pi}_2(\boldsymbol{\theta}|x)$, where c_1 and c_2 are unknown normalising constants, then the equality

$$\frac{c_2}{c_1} = \frac{\mathbb{E}^{\pi_2}[\tilde{\pi}_1(\boldsymbol{\theta}|x)\, h(\boldsymbol{\theta})]}{\mathbb{E}^{\pi_1}[\tilde{\pi}_2(\boldsymbol{\theta}|x)\, h(\boldsymbol{\theta})]}$$

holds for any *bridge function* $h(\boldsymbol{\theta})$ such that both expectations are finite. The *bridge sampling* estimator is then

$$B_{12}^S = \frac{\dfrac{1}{n_1}\displaystyle\sum_{i=1}^{n_1} \tilde{\pi}_2(\boldsymbol{\theta}_{1i}|x)\, h(\boldsymbol{\theta}_{1i})}{\dfrac{1}{n_2}\displaystyle\sum_{i=1}^{n_2} \tilde{\pi}_1(\boldsymbol{\theta}_{2i}|x)\, h(\boldsymbol{\theta}_{2i})},$$

where the $\boldsymbol{\theta}_{ji}$'s are simulated from $\pi_j(\boldsymbol{\theta}|x)$ $(j = 1, 2,\ i = 1, \ldots, n_j)$.

For instance, if

$$h(\boldsymbol{\theta}) = 1/\left[\tilde{\pi}_1(\boldsymbol{\theta}|x)\tilde{\pi}_2(\boldsymbol{\theta}_{1i}|x)\right],$$

then B_{12}^S is a ratio of harmonic means, generalising (26.9). Meng and Wong (1996) have derived an (asymptotically) optimal bridge function

$$h^*(\boldsymbol{\theta}) = \frac{n_1 + n_2}{n_1\pi_1(\boldsymbol{\theta}|x) + n_2\pi_2(\boldsymbol{\theta}|x)}.$$

This choice is not of direct use, since the normalising constants of $\pi_1(\boldsymbol{\theta}|x)$ and $\pi_2(\boldsymbol{\theta}|x)$ are unknown (otherwise, we should not need to resort to such techniques!). Nonetheless, it shows that a good bridge function should cover the support of both posteriors, with equal weights if $n_1 = n_2$.

Example 12 (Continuation of Example 2). For *generalised linear models*, the mean (conditionally on the covariates) satisfies

$$\mathbb{E}[y|\boldsymbol{\theta}] = \nabla\psi(\boldsymbol{\theta}) = \Psi(x^t\beta),$$

where Ψ is the *link function*. The choice of the *link function* Ψ usually is quite open. For instance, when the y's take values in $\{0, 1\}$, three common choices of Ψ are (McCullagh and Nelder 1989)

$$\Psi_1(t) = \exp(t)/(1+\exp(t)), \quad \Psi_2(t) = \Phi(t), \quad \text{and} \quad \Psi_3(t) = 1-\exp(-\exp(t)),$$

corresponding to the *logit*, *probit* and *log–log* link functions (where Φ denotes the c.d.f. of the $\mathcal{N}(0,1)$ distribution). If the prior distribution π on the β's is a normal $\mathcal{N}_p(\xi, \tau^2 I_p)$, and if the bridge function is $h(\beta) = 1/\pi(\beta)$, the bridge sampling estimate is then $(1 \le i < j \le 3)$

$$B_{ij}^S = \frac{\dfrac{1}{n}\displaystyle\sum_{t=1}^{n} f_j(\beta_{it}|x)}{\dfrac{1}{n}\displaystyle\sum_{t=1}^{n} f_i(\beta_{jt}|x)},$$

where the β_{it} are generated from $\pi_i(\beta_i|x) \propto f_i(\beta_i|x)\pi(\beta_i)$, that is, from the true posteriors for each link function.

The drawback in using bridge sampling is that the extension of the method to cases where the two models $\mathfrak{M}_1$ and $\mathfrak{M}_2$ do not cover the same parameter space, for instance because the two corresponding parameter spaces are of different dimensions, requires the introduction of a pseudo-posterior distribution (Chen et al. 2000; Marin and Robert 2007b) that complete the smallest parameter space so that dimensions match.[9] The impact of those completion pseudo-posteriors on the quality of the Bayes factor approximation is such that they cannot be suggested for a general usage in Bayes factor computation.

A completely different route to the approximation of a marginal likelihood is Chib's (1995), which happens to be a direct application of Bayes' theorem: given $x \sim f_k(x|\boldsymbol{\theta}_k)$ and $\boldsymbol{\theta}_k \sim \pi_k(\boldsymbol{\theta}_k)$ $(k = 1, 2)$, we have that

$$m_k(x) = \frac{f_k(x|\boldsymbol{\theta})\,\pi_k(\boldsymbol{\theta})}{\pi_k(\boldsymbol{\theta}|x)},$$

for every value of $\boldsymbol{\theta}$ (since both the lhs and the rhs of this equation are constant in $\boldsymbol{\theta}$). Therefore, if an arbitrary value of $\boldsymbol{\theta}$, say $\boldsymbol{\theta}_k^*$, is selected – e.g., the MAP estimate – and if a good approximation to $\pi_k(\boldsymbol{\theta}|y)$ can be constructed, denoted $\hat{\pi}(\boldsymbol{\theta}|y)$, Chib's (1995) approximation to the marginal likelihood $m_k(x)$ is

$$\widehat{m}_k(x) = \frac{f_k(y|\boldsymbol{\theta}_k^*)\,\pi_k(\boldsymbol{\theta}_k^*)}{\hat{\pi}_k(\boldsymbol{\theta}_k^*|y)}. \tag{26.10}$$

As a first solution, $\hat{\pi}(\boldsymbol{\theta}|y)$ may be a Gaussian approximation based on the MLE, but this is unlikely to be accurate in a general setting. Chib's (1995) solution is to use a nonparametric approximation based on a preliminary Monte Carlo Markov chain (Sect. 26.4) sample, even though the accuracy may also suffer in large dimensions.

[9] Sect. 26.4.3 covers in greater details the setting of varying dimension problems, with the same theme that completion distributions and parameters are necessary but influential for the performances of the approximation.

In the special setting of latent variables models (like the mixtures of distributions discussed in Example 6), this approximation is particularly attractive as there exists a natural approximation to $\pi_k(\boldsymbol{\theta}|y)$, based on the Rao–Blackwell (Gelfand and Smith 1990) estimate

$$\hat{\pi_k}(\boldsymbol{\theta}_k^*|y) = \frac{1}{T}\sum_{t=1}^{T} \pi_k(\boldsymbol{\theta}_k^*|y, z_k^{(t)}),$$

where the $z_k^{(t)}$'s are the latent variables simulated by the Monte Carlo Markov chain algorithm. The estimate $\hat{\pi_k}(\boldsymbol{\theta}_k^*|y)$ is a parametric unbiased approximation of $\pi_k(\boldsymbol{\theta}_k^*|y)$ that converges to the true density with rate $O(\sqrt{T})$. This Rao–Blackwell approximation obviously requires the full conditional density $\pi_k(\boldsymbol{\theta}_k^*|y, z)$ to be available in closed form (constant included) but this is the case for many standard models. Figure 26.7 reproduces an illustration from Marin and Robert (2007b) that compares the variability of Chib's (1995) method against nearly optimal harmonic and importance sampling approximations in the setting of a probit posterior distribution. While the former solution varies more than the two

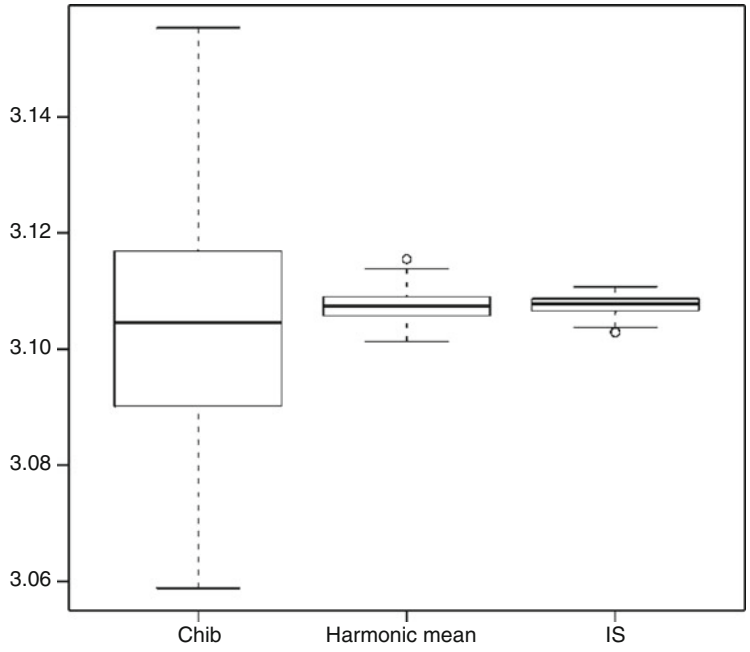


Fig. 26.7 In the setting of the probit modelling of R Pima Indian dataset, using three covariates and testing for the significance of the diabetes pedigree function, boxplots of 100 Chib's, harmonic mean and importance estimates of $B_{01}(y)$, based on simulations from the prior distributions, for 2×10^4 simulations (*Source:* Marin and Robert 2007b)

latter solutions, its generic features make Chib's (1995) method a reference for this problem.

While amenable to an importance sampling technique of sorts, the alternative approach of nested sampling (Skilling 2006) for computing evidence is discussed in Chopin and Robert (2010) and Robert and Wraith (2009). Similarly, the specific approach based on the Savage-Dickey (Dickey 1971; Verdinelli and Wasserman 1995) representation of the Bayes factor associated with the null hypothesis $H_0 : \boldsymbol{\theta} = \boldsymbol{\theta}_0$, as

$$B_{01}(x) = \frac{\pi_1(\boldsymbol{\theta}_0|x)}{\pi_1(\boldsymbol{\theta}_0)},$$

can be related to the family of bridge sampling techniques, even though the initial theoretical foundations of the method are limited, as explained in Robert and Marin (2009). This paper engineered a general framework that produces a converging and unbiased approximation of the Bayes factor, unrelated with the approach of Verdinelli and Wasserman (1995), that builds upon a mixed pseudo-posterior naturally derived from the priors under both the null and the alternative hypotheses.

As can be seen from the previous developments, such methods require a rather careful tuning to be of any use. Therefore, they are rather difficult to employ outside settings where pairs of models are opposed. In other words, they cannot be directly used in general model choice settings where the parameter space (and in particular the parameter dimension) varies across models, as in, for instance, Example 7. To address the computational issues corresponding to these cases requires more advanced techniques introduced in the next Section.

26.4 Markov Chain Monte Carlo Methods

As described precisely in Chap. II.3 and in Robert and Casella (2004), MCMC methods try to overcome the limitation of regular Monte Carlo methods through the use of a Markov chain with stationary distribution the posterior distribution. There exist rather generic ways of producing such chains, including Metropolis–Hastings and Gibbs algorithms. Besides the fact that stationarity of the target distribution is enough to justify a simulation method by Markov chain generation, the idea at the core of MCMC algorithms is that local exploration, when properly weighted, can lead to a valid representation of the distribution of interest, as for instance, when using the Metropolis–Hastings algorithm.

26.4.1 Metropolis–Hastings as Universal Simulator

The Metropolis–Hastings, presented in Robert and Casella (2004) and Chap. II.3, offers a straightforward solution to the problem of simulating from the posterior distribution $\pi(\boldsymbol{\theta}|x) \propto f(x|\boldsymbol{\theta})\,\pi(\boldsymbol{\theta})$: starting from an arbitrary point $\boldsymbol{\theta}_0$, the corresponding Markov chain explores the surface of this posterior distribution by a

similarly arbitrary random walk proposal $q(\boldsymbol{\theta}|\boldsymbol{\theta}')$ that progressively visits the whole range of the possible values of $\boldsymbol{\theta}$.

– Metropolis–Hastings Algorithm –

At iteration t

1 Generate $\xi \sim q(\xi|\boldsymbol{\theta}^{(t)})$, $u_t \sim \mathcal{U}([0,1])$

2 Take

$$\boldsymbol{\theta}^{(t+1)} = \begin{cases} \xi_t & \text{if } u_t \leq \dfrac{\pi(\xi_t|x)}{\pi(\boldsymbol{\theta}^{(t)}|x)} \dfrac{q(\boldsymbol{\theta}^{(t)}|\xi_t)}{q(\xi_t|\boldsymbol{\theta}^{(t)})} \\ \boldsymbol{\theta}^{(t)} & \text{otherwise} \end{cases}$$

Example 13 (Continuation of Example 10). In the case $p = 1$, the probit model defined in Example 10 can also be over-parameterised as

$$\Pr(Y_i = 1|x_i) = 1 - \Pr(Y_i = 0|x_i) = \Phi(x_i\beta/\sigma)\,,$$

since it only depends on β/σ. The Bayesian processing of non-identified models poses no serious difficulty as long as the posterior distribution is well defined. This is the case for a proper prior like

$$\pi(\beta, \sigma^2) \propto \sigma^{-4} \exp\{-1/\sigma^2\} \exp\{-\beta^2/50)$$

that corresponds to a normal distribution on β and a gamma distribution on σ^{-2}. While the posterior distribution on (β, σ) is not a standard distribution, it is available up to a normalising constant. Therefore, it can be directly processed via an MCMC algorithm. In this case, we chose a Gibbs sampler that simulates β and σ^2 alternatively, from

$$\pi(\beta|\mathbf{x}, \mathbf{y}, \sigma) \propto \prod_{y_i=1} \Phi(x_i\beta/\sigma) \prod_{y_i=0} \Phi(-x_i\beta/\sigma) \times \pi(\beta)$$

and

$$\pi(\sigma^2|\mathbf{x}, \mathbf{y}, \beta) \propto \prod_{y_i=1} \Phi(x_i\beta/\sigma) \prod_{y_i=0} \Phi(-x_i\beta/\sigma) \times \pi(\sigma^2)$$

respectively. Since both of these conditional distributions are also non-standard, we replace the direct simulation by a one-dimensional Metropolis–Hastings step, using normal $\mathcal{N}(\beta^{(t)}, 1)$ and log-normal $\mathcal{LN}(\log \sigma^{(t)}, .04)$ random walk proposals, respectively. For a simulated dataset of $1,000$ points, the contour plot of the log-posterior distribution is given in Fig. 26.8, along with the last 1,000 points of a corresponding MCMC sample after 100,000 iterations. This graph shows a very satisfactory repartition of the simulated parameters over the likelihood surface, with higher concentrations near the largest posterior regions. For another simulation, Fig. 26.9 details the first 500 steps, when started at $(\beta, \sigma^2) = (0.1, 4.0)$. Although each step contains both a β *and* a σ proposal, some moves are either horizontal or vertical: this corresponds to cases when either the β or the σ proposals have been

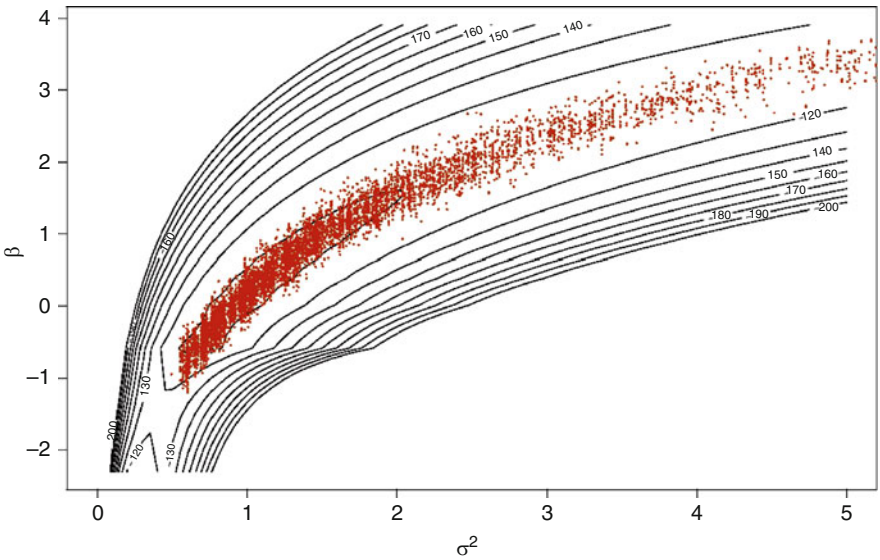


Fig. 26.8 Contour plot of the log-posterior distribution for a probit sample of 1,000 observations, along with 1,000 points of an MCMC sample *(Source: Robert and Casella 2004)*

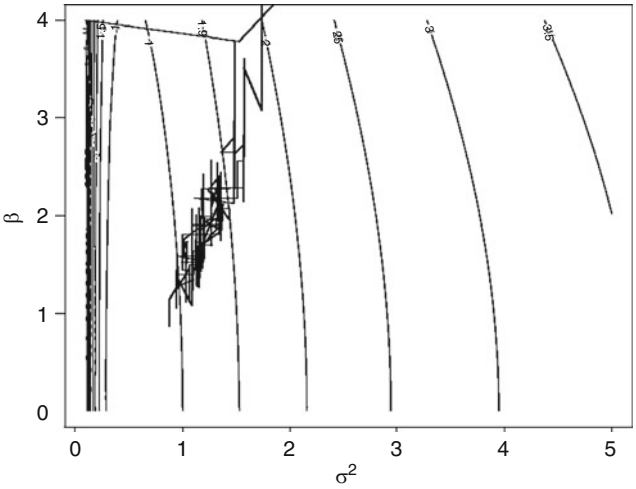


Fig. 26.9 First 500 steps of the Metropolis–Hastings algorithm on the probit log-posterior surface, when started at $(\beta, \sigma^2) = (0.1, 4.0)$

rejected. Note also the fairly rapid convergence to a modal zone of the posterior distribution in this case.

Obviously, this is only a toy example and more realistic probit models do not fare so well with down-the-shelf random walk Metropolis–Hastings algorithms, as

shown for instance in Nobile (1998) (see also Robert and Casella 2004, Sect. 10.3.2, Marin and Robert 2007a, Sect. 4.3, and Marin and Robert 2007b).[10]

The difficulty inherent to random walk Metropolis–Hastings algorithms is the scaling of the proposal distribution: it must be adapted to the shape of the target distribution so that, in a reasonable number of steps, the whole support of this distribution can be visited. If the scale of the proposal is too small, this will not happen as the algorithm stays "too local" and, if there are several modes on the posterior, the algorithm may get trapped within one modal region because it cannot reach other modal regions with jumps of too small a magnitude. The larger the dimension p is, the harder it is to set up the right scale, though, because:

(a) The curse of dimension implies that there are more and more empty regions in the space, that is, regions with zero posterior probability.
(b) The knowledge and intuition about the modal regions get weaker and weaker.
(c) The proper scaling involves a symmetric (p, p) matrix Ξ in the proposal $g((\boldsymbol{\theta} - \boldsymbol{\theta}')^{\mathrm{T}} \Xi (\boldsymbol{\theta} - \boldsymbol{\theta}'))$. Even when the matrix Ξ is diagonal, it gets harder to scale as the dimension increases (unless one resorts to a Gibbs like implementation, where each direction is scaled separately).

Note also that the on-line scaling of the algorithm against the empirical acceptance rate is inherently flawed in that (a) the process is no longer Markovian and (b) the attraction of a modal region may give a false sense of convergence and lead to a choice of too small a scale, simply because other modes will not be visited during the scaling experiment.

26.4.2 Gibbs Sampling and Latent Variable Models

The Gibbs sampler is a definitely attractive algorithm for Bayesian problems because it naturally fits the hierarchical structures so often found in such problems. "Natural" being a rather vague notion from a simulation point of view, it routinely happens that other algorithms fare better than the Gibbs sampler. Nonetheless, Gibbs sampler is often worth a try (possibly with other Metropolis–Hastings refinements at a later stage) in well-structured objects like Bayesian hierarchical models and more general graphical models.

A very relevant illustration is made of latent variable models, where the observational model is itself defined as a mixture model,

$$f(x|\boldsymbol{\theta}) = \int_{\mathscr{Z}} f(x|z, \boldsymbol{\theta})\, g(z|\boldsymbol{\theta})\, \mathrm{d}z.$$

Such models were instrumental in promoting the Gibbs sampler in the sense that they have the potential to make Gibbs sampling sound natural very easily. (See

[10] Even in the simple case of the probit model, MCMC algorithms do not always converge very quickly, as shown in Robert and Casella (2004, Chap. 14).

also Chap. II.3.) For instance, Tanner and Wong (1987) wrote a precursor article to Gelfand and Smith (1990) that designed specific two-stage Gibbs samplers for a variety of latent variable models. And many of the first applications of Gibbs sampling in the early 90's were actually for models of that kind. The usual implementation of the Gibbs sampler in this case is to simulate the missing variables Z conditional on the parameters and reciprocally, as follows:

– Latent Variable Gibbs Algorithm –

At iteration t

1 Generate $z^{(t+1)} \sim g(z|\boldsymbol{\theta}^{(t)}, x)$
2 Generate $\boldsymbol{\theta}^{(t+1)} \sim \pi(\boldsymbol{\theta}|x, z^{(t+1)})$

While we could have used the probit case as an illustration (Example 10), as done in Chap. II.3, we choose to pick the case of mixtures (Example 6) as a better setting.

Example 14 (Continuation of Example 6). The natural missing data structure of a mixture of distribution is historical. In one of the first mixtures to be ever studied by Bertillon, in 1863, a bimodal structure on the height of conscripts in south eastern France (Doubs) can be explained by the mixing of two populations of military conscripts, one from the plains and one from the mountains (or hills). Therefore, in the analysis of data from distributions of the form

$$\sum_{i=1}^{k} p_i f(x|\boldsymbol{\theta}_i)\,,$$

a common missing data representation is to associate with each observation x_j a missing multinomial variable $z_j \sim \mathcal{M}_k(1; p_1, \ldots, p_k)$ such that $x_j|z_j = i \sim f(x|\boldsymbol{\theta}_i)$. In heterogeneous populations made of several homogeneous subgroups or subpopulations, it makes sense to interpret z_j as the index of the population of origin of x_j, which has been lost in the observational process.

However, mixtures are also customarily used for density approximations, as a limited dimension proxy to non-parametric approaches. In such cases, the components of the mixture and even the number k of components in the mixture are often meaningless for the problem to be analysed. But this distinction between natural and artificial completion (by the z_j's) is lost to the MCMC sampler, whose goal is simply to provide a Markov chain that converges to the posterior as stationary distribution. Completion is thus, from a simulation point of view, a mean to generate such a chain.

The most standard Gibbs sampler for mixture models (Diebolt and Robert 1994) is thus based on the successive simulation of the z_j's and of the $\boldsymbol{\theta}_i$'s, conditional on one another and on the data:

1. Generate $z_j|\boldsymbol{\theta}, x_j$ $(j = 1, \ldots, n)$
2. Generate $\boldsymbol{\theta}_i|\mathbf{x}, \mathbf{z}$ $(i = 1, \ldots, k)$

Given that the density f is most often from an exponential family, the simulation of the $\boldsymbol{\theta}_i$'s is generally straightforward.

As an illustration, consider the (academic) case of a normal mixture with two components, with equal known variance and fixed weights,

$$p\,\mathcal{N}(\mu_1, \sigma^2) + (1-p)\,\mathcal{N}(\mu_2, \sigma^2)\,. \tag{26.11}$$

Assume in addition a normal $\mathcal{N}(0, 10\sigma^2)$ prior on both means μ_1 and μ_2. It is easy to see that μ_1 and μ_2 are independent, given $(\mathbf{z}, \mathbf{x})$, and the respective conditional distributions are

$$\mathcal{N}\left(\sum_{z_i=j} x_i / \left(.1 + n_j\right), \sigma^2 / \left(.1 + n_j\right)\right),$$

where n_j denotes the number of z_i's equal to j. Even more easily, it comes that the conditional posterior distribution of $\mathbf{z}$ given (μ_1, μ_2) is a product of binomials, with

$$\Pr(Z_i = 1 | x_i, \mu_1, \mu_2) = \frac{p \exp\{-(x_i - \mu_1)^2/2\sigma^2\}}{p \exp\{-(x_i - \mu_1)^2/2\sigma^2\} + (1-p)\exp\{-(x_i - \mu_2)^2/2\sigma^2\}}\,.$$

Figure 26.10 illustrates the behaviour of the Gibbs sampler in that setting, with a simulated dataset of 100 points from the $.7\mathcal{N}(0,1) + .3\mathcal{N}(2.7,1)$ distribution. The representation of the MCMC sample after 5,000 iterations is quite in agreement with the posterior surface, represented via a grid on the (μ_1, μ_2) space and some

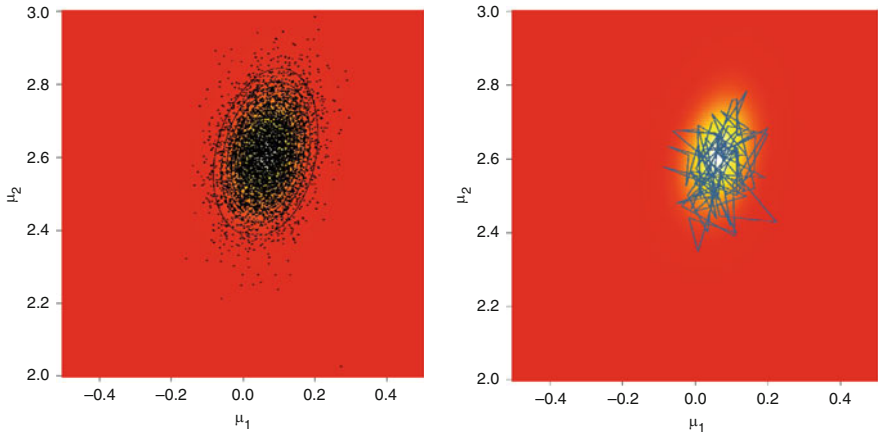


Fig. 26.10 Gibbs sample of 5,000 points for the mixture posterior *(left)* and path of the last 100 consecutive steps *(right)* against the posterior surface *(Source: Robert and Casella 2004)*

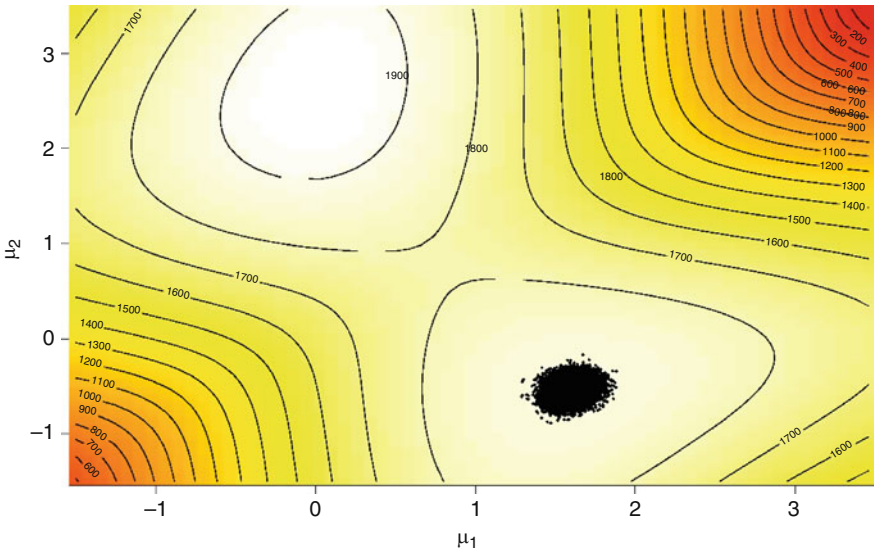


Fig. 26.11 Posterior surface and corresponding Gibbs sample for the two mean mixture model, when initialised close to the second and lower mode, based on 10,000 iterations *(Source: Robert and Casella 2004)*

contours. The sequence of consecutive steps represented on the left graph also shows that the mixing behaviour is satisfactory, since the jumps are scaled in terms of the modal region of the posterior.

This experiment gives a wrong sense of safety, though, because it does not point out the fairly large dependence of the Gibbs sampler to the initial conditions, already signalled in Diebolt and Robert (1994) under the name of *trapping states*. Indeed, the conditioning of (μ_1, μ_2) on $\mathbf{z}$ implies that the new simulations of the means will remain very close to the previous values, especially if there are many observations, and thus that the new allocations $\mathbf{z}$ will not differ much from the previous allocations. In other words, to see a significant modification of the allocations (and thus of the means) would require a very very large number of iterations. Figure 26.11 illustrates this phenomenon for the same sample as in Fig. 26.10, for a wider scale: there always exists a second mode in the posterior distribution, which is much lower than the first mode located around $(0, 2.7)$. Nonetheless, a Gibbs sampler initialised close to the second and lower mode will not be able to leave the vicinity of this (irrelevant) mode, even after a large number of iterations. The reason is as given above: to jump to the other mode, a majority of z_j's would need to change simultaneously and the probability of such a jump is too close to 0 to let the event occur.[11]

[11]It is quite interesting to see that the mixture Gibbs sampler suffers from the same pathology as the EM algorithm, although this is not surprising given that it is based on the same completion scheme.

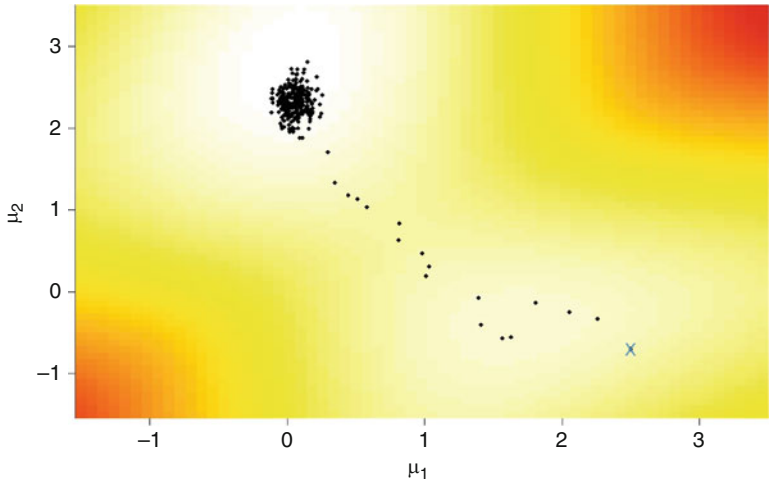


Fig. 26.12 Track of a 1,000 iteration random walk Metropolis–Hastings chain on the posterior surface, the starting point being indicated by a cross. *(The scale of the random walk is 0.2)*

The literature on this specific issue – of exploring all the modes of the posterior distribution – has grown around the denomination of "label switching problem". For instance, Celeux et al. (2000) have pointed out the deficiency of most MCMC samplers in this respect, while Frühwirth-Schnatter (2001, 2004, 2006), Berkhof et al. (2003), Jasra et al. (2005), and Geweke (2007) have proposed different devices to overcome this difficulty, in particular when testing for the number of components. See Lee et al. (2009) for a survey on recent developments.

This example illustrates quite convincingly that, while the completion is natural from a model point of view (since it is a part of the definition of the model), it does not necessarily transfer its utility for the simulation of the posterior. Actually, when the missing variable model allows for a closed form likelihood, as is the case for mixtures, probit models (Examples 10 and 13) and even hidden Markov models (see Cappé et al. 2005), the whole range of the MCMC technology can be used as well. The appeal of alternatives like random walk Metropolis–Hastings schemes is that they remain in a smaller dimension space, since they avoid the completion step(s), and that they are not restricted in the range of their moves.[12]

Example 15 (Continuation of Example 14). Given that the likelihood of a sample $(x_1, \ldots, x_n)$ from the mixture distribution (26.11) can be computed in $O(2n)$ time, a regular random walk Metropolis–Hastings algorithm can be used in this setup. Figure 26.12 shows how quickly this algorithm escapes the attraction of the poor mode, as opposed to the Gibbs sampler of Fig. 26.11: within a few

[12]This wealth of possible alternatives to the completion Gibbs sampler is a mixed blessing in that their range, for instance the scale of the random walk proposals, needs to be scaled properly to avoid inefficiencies.

iterations of the algorithm, the chain drifts over the poor mode and converges almost deterministically to the proper region of the posterior surface. The random walk is based on $\mathcal{N}(\mu_i^{(t)}, 0.04)$ proposals, although other scales would work as well but would require more iterations to reach the proper model regions. For instance, a scale of 0.005 in the Normal proposal above needs close to 5,000 iterations to attain the main mode.

The secret of a successful MCMC implementation in such latent variable models is to maintain the distinction between latency in models and latency in simulation (the later being often called use of *auxiliary variables*). When latent variables can be used with adequate mixing of the resulting chain and when the likelihood cannot be computed in a closed form (as in hidden semi-Markov models, Cappé et al. 2004), a Gibbs sampler is a rather simple solution that is often easy to simulate from. Adding well-mixing random walk Metropolis–Hastings steps in the simulation scheme cannot hurt the overall mixing of the chain (Robert and Casella 2004, Chap. 13), especially when several scales can be used at once (see Sect. 26.5). Note also that the technique of tempering that flattens the target distribution in order to facilitate its exploration by a Markov chain is available for most latent variable models, if sometimes in rudimentary versions. See Chopin (2007) and Marin and Robert (2007, Sect. 6.6) for an illustration in the setups of hidden Markov models and of mixtures (Example 15), respectively. A final word is that the latent variable completion can be conducted in an infinity of ways and that several of these should be tried or used in parallel to increase the chances of success.

26.4.3 Reversible Jump Algorithms for Variable Dimension Models

As described in Sect. 26.2.3, model choice is computationally different from testing in that it considers at once a (much) wider range of models $\mathfrak{M}_i$ and parameter spaces Θ_i. Although early approaches could only go through a pedestrian pairwise comparison, a more adequate perspective is to envision the model index μ as part of the parameter to be estimated, as in (26.3). The (computational) difficulty is that we are then dealing with a possibly infinite space[13] that is the collection of unrelated sets: how can we then simulate from the corresponding distribution?[14]

[13]The difficulty with the infinite part of the problem is easily solved in that the setting is identical to simulation problems in (countable or uncountable) infinite spaces. When running simulations in those spaces, some values are never visited by the simulated Markov chain and the chances a value is visited is related with the probability of this value under the target distribution.

[14]Early proposals to solve the varying dimension problem involved saturation schemes where all the parameters for all models were updated deterministically (Carlin and Chib 1995), but they do not apply for an infinite collection of models and they need to be precisely calibrated to achieve a sufficient amount of moves between models.

The MCMC solution proposed by Green (1995) is called *reversible jump MCMC*, because it is based on a *reversibility* constraint on the transitions between the sets Θ_i. In fact, the only real difficulty compared with previous developments is to validate moves (or *jumps*) between the Θ_i's, since proposals restricted to a given Θ_i follow from the usual (fixed-dimensional) theory. Furthermore, *reversibility* can be processed at a local level: since the model indicator μ is a integer-valued random variable, we can impose reversibility for each pair (k_1, k_2) of possible values of μ. The idea at the core of reversible jump MCMC is then to supplement each of the spaces Θ_{k_1} and Θ_{k_2} with adequate artificial spaces in order to create a *bijection* between them. For instance, if $\dim(\Theta_{k_1}) > \dim(\Theta_{k_2})$ and if the move from Θ_{k_1} to Θ_{k_2} can be represented by a *deterministic* transformation of $\boldsymbol{\theta}^{(k_1)}$

$$\boldsymbol{\theta}^{(k_2)} = T_{k_1 \to k_2}(\boldsymbol{\theta}^{(k_1)}) \,,$$

Green (1995) imposes a *dimension matching* condition which is that the opposite move from Θ_{k_2} to Θ_{k_1} is concentrated on the curve

$$\left\{ \boldsymbol{\theta}^{(k_1)} \,:\, \boldsymbol{\theta}^{(k_2)} = T_{k_1 \to k_2}(\boldsymbol{\theta}^{(k_1)}) \right\} \,.$$

In the general case, if $\boldsymbol{\theta}^{(k_1)}$ is completed by a simulation $u_{k_1} \sim g_{k_1}(u_{k_1})$ into $(\boldsymbol{\theta}^{(k_1)}, u_{k_1})$ and $\boldsymbol{\theta}^{(k_2)}$ by $u_{k_2} \sim g_{k_2}(u_{k_2})$ into $(\boldsymbol{\theta}^{(k_2)}, u_{k_2})$ so that the mapping between $(\boldsymbol{\theta}^{(k_1)}, u_{k_1})$ and $(\boldsymbol{\theta}^{(k_2)}, u_{k_2})$ is a bijection,

$$(\boldsymbol{\theta}^{(k_2)}, u_{k_2}) = T_{k_1 \to k_2}(\boldsymbol{\theta}^{(k_1)}, u_{k_1}), \tag{26.12}$$

the probability of acceptance for the move from model $\mathfrak{M}_{k_1}$ to model $\mathfrak{M}_{k_2}$ is then

$$\min\left(\frac{\pi(k_2, \boldsymbol{\theta}^{(k_2)})}{\pi(k_1, \boldsymbol{\theta}^{(k_1)})} \, \frac{\pi_{21} g_{k_2}(u_{k_2})}{\pi_{12} g_{k_1}(u_{k_1})} \left| \frac{\partial T_{k_1 \to k_2}(\boldsymbol{\theta}^{(k_1)}, u_{k_1})}{\partial(\boldsymbol{\theta}^{(k_1)}, u_{k_1})} \right|, 1 \right) \,,$$

involving

- The Jacobian of the transform $T_{k_1 \to k_2}$,
- The probability π_{ij} of choosing a jump to $\mathcal{M}_{k_j}$ while in $\mathcal{M}_{k_i}$, and
- g_i, the density of u_i.

The acceptance probability for the reverse move is based on the inverse ratio if the move from $\mathfrak{M}_{k_2}$ to $\mathfrak{M}_{k_1}$ also satisfies (26.12) with $u_{k_2} \sim g_{k_2}(u_{k_2})$.[15]

The pseudo-code representation of Green's algorithm is thus as follows:

[15]For a simple proof that the acceptance probability guarantees that the stationary distribution is $\pi(k, \boldsymbol{\theta}^{(k)})$, see Robert and Casella (2004, Sect. 11.2.2).

– Green's Algorithm –

At iteration t, if $x^{(t)} = (m, \boldsymbol{\theta}^{(m)})$,

1. Select model $\mathfrak{M}_n$ with probability π_{mn}
2. Generate $u_{mn} \sim \varphi_{mn}(u)$
3. Set $(\boldsymbol{\theta}^{(n)}, v_{nm}) = T_{m\to n}(\boldsymbol{\theta}^{(m)}, u_{mn})$
4. Take $x^{(t+1)} = (n, \boldsymbol{\theta}^{(n)})$ with probability

$$\min\left(\frac{\pi(n,\boldsymbol{\theta}^{(n)})}{\pi(m,\boldsymbol{\theta}^{(m)})}\,\frac{\pi_{nm}\varphi_{nm}(v_{nm})}{\pi_{mn}\varphi_{mn}(u_{mn})}\left|\frac{\partial T_{m\to n}(\boldsymbol{\theta}^{(m)},u_{mn})}{\partial(\boldsymbol{\theta}^{(m)},u_{mn})}\right|,1\right),$$

and take $x^{(t+1)} = x^{(t)}$ otherwise.

Even more than previous methods, the implementation of this algorithm requires a high degree of skillfulness in picking the right proposals and the appropriate scales. Indeed, it may be argued that the ultimate dependence of the method on the pairwise completion schemes prevents an extension of its use by a broader audience. This "art of reversible jump MCMC" is illustrated on the two following examples, extracted from Robert and Casella (2004, Sect. 14.2.3).

Example 16 (Continuation of Example 6). If we consider for model $\mathfrak{M}_k$ the k component normal mixture distribution,

$$\sum_{j=1}^{k} p_{jk}\mathcal{N}(\mu_{jk},\sigma_{jk}^2)\,,$$

moves between models involve changing the number of components in the mixture and thus adding new components or removing older components or yet again changing several components. As in Richardson and Green (1997), we can restrict the moves when in model $\mathfrak{M}_k$ to only models $\mathfrak{M}_{k+1}$ and $\mathfrak{M}_{k-1}$. The simplest solution is to use a birth-and-death process: The *birth step* consists in adding a new normal component in the mixture generated from the prior and the *death step* is the opposite, removing one of the k components at random. In this case, the corresponding birth acceptance probability is

$$\min\left(\frac{\pi_{(k+1)k}}{\pi_{k(k+1)}}\,\frac{(k+1)!}{k!}\,\frac{\pi_{k+1}(\boldsymbol{\theta}_{k+1})}{\pi_k(\boldsymbol{\theta}_k)\,(k+1)\varphi_{k(k+1)}(u_{k(k+1)})},1\right)$$

$$=\min\left(\frac{\pi_{(k+1)k}}{\pi_{k(k+1)}}\,\frac{\varrho(k+1)}{\varrho(k)}\,\frac{\ell_{k+1}(\boldsymbol{\theta}_{k+1})\,(1-p_{k+1})^{k-1}}{\ell_k(\boldsymbol{\theta}_k)},1\right),$$

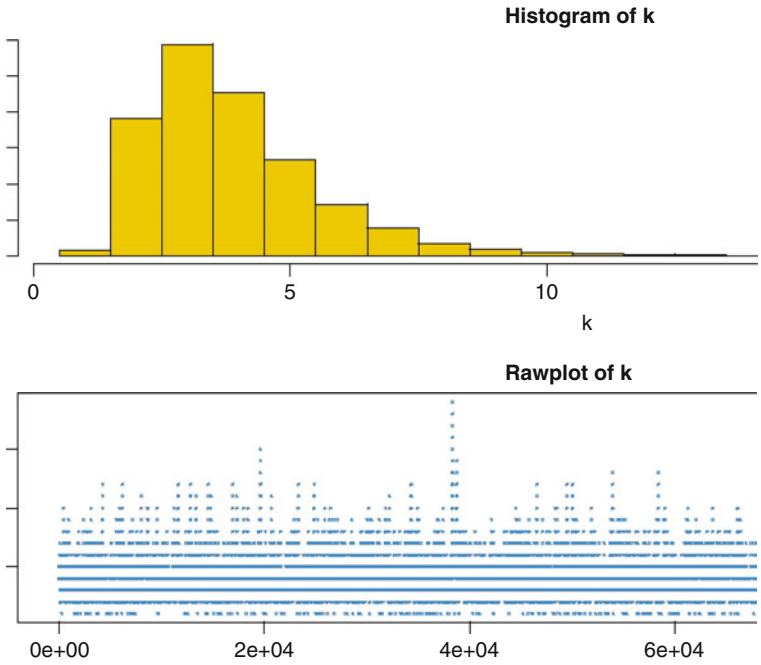


Fig. 26.13 Histogram and raw plot of 100,000 k's produced by a reversible jump MCMC algorithm for the Galaxy dataset

where ℓ_k denotes the likelihood of the k component mixture model $\mathfrak{M}_k$ and $\varrho(k)$ is the prior probability of model $\mathfrak{M}_k$.[16]

While this proposal can work well in some setting, as in Richardson and Green (1997) when the prior is calibrated against the data, it can also be inefficient, that is, leading to a high rejection rate, if the prior is vague, since the birth proposals are not tuned properly. A second proposal, central to the solution of Richardson and Green (1997), is to devise more local jumps between models, called *split* and *combine* moves, since a new component is created by splitting an existing component into two, under some moment preservation conditions, and the reverse move consists in combining two existing components into one, with symmetric constraints that ensure reversibility. (See, e.g., Robert and Casella 2004, for details.)

Figures 26.13–26.15 illustrate the implementation of this algorithm for the so-called Galaxy dataset used by Richardson and Green (1997) (see also Roeder 1992), which contains 82 observations on the speed of galaxies. On Fig. 26.13, the MCMC output on the number of components k is represented as a histogram on k, and

[16] In the birth acceptance probability, the factorials $k!$ and $(k+1)!$ appear as the numbers of ways of ordering the k and $k+1$ components of the mixtures. The ratio cancels with $1/(k+1)$, which is the probability of selecting a particular component for the death step.

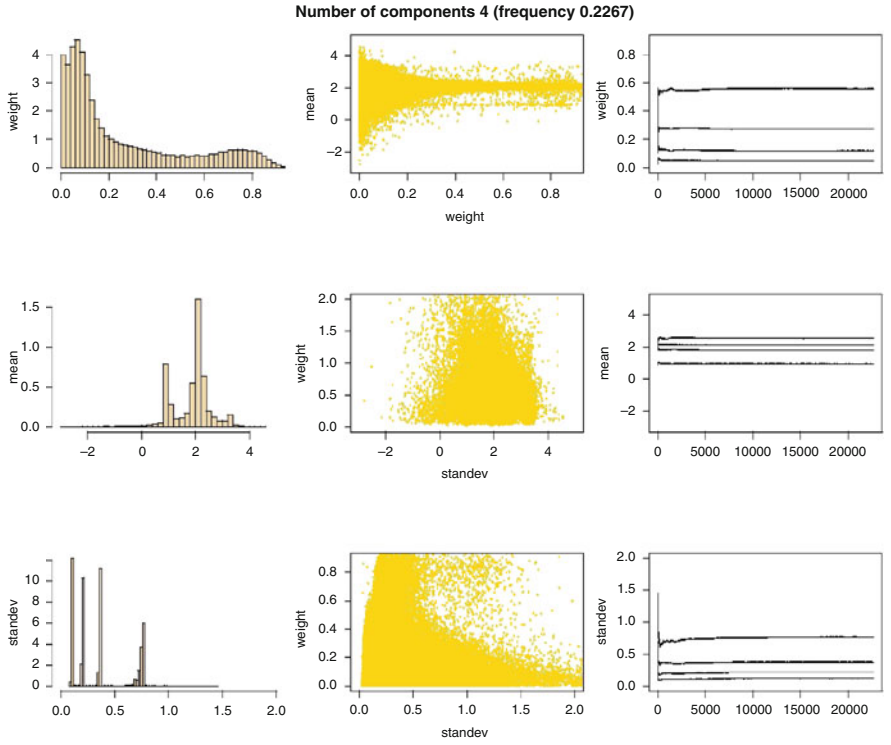


Fig. 26.14 Reversible jump MCMC output on the parameters of the model $\mathcal{M}_3$ for the Galaxy dataset, obtained by conditioning on $k = 3$. *The left column gives the histogram of the weights, means, and variances; the middle column the scatterplot of the pairs weights-means, means-variances, and variances-weights; the right column plots the cumulated averages (over iterations) for the weights, means, and variances*

the corresponding sequence of k's. The prior used on k is a uniform distribution on $\{1, \ldots, 20\}$: as shown by the lower plot, most values of k are explored by the reversible jump algorithm, but the upper bound does not appear to be restrictive since the $k^{(t)}$'s hardly ever reach this upper limit. Figure 26.14 illustrates the fact that conditioning the output on the most likely value of k (3 here) is possible. The nine graphs in this Figure show the joint variation of the three types of parameters, as well as the stability of the Markov chain over the 1,000,000 iterations: the cumulated averages are quite stable, almost from the start.

The density plotted on top of the histogram in Fig. 26.15 is another good illustration of the inferential possibilities offered by reversible jump algorithms, as a case of *model averaging*: this density is obtained as the average over iterations t of

$$\sum_{j=1}^{k^{(t)}} p_{jk}^{(t)} \mathcal{N}(\mu_{jk}^{(t)}, (\sigma_{jk}^{(t)})^2)\,,$$

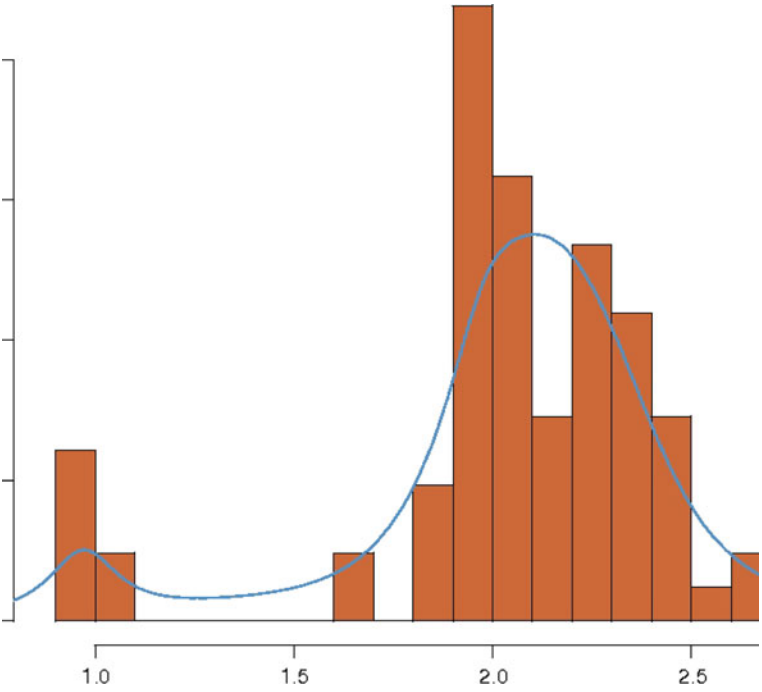


Fig. 26.15 Fit of the dataset by the averaged density, $\mathbb{E}[f(y|\boldsymbol{\theta})|\mathbf{x}]$

which approximates the posterior expectation $\mathbb{E}[f(y|\boldsymbol{\theta})|\mathbf{x}]$, where $\mathbf{x}$ denotes the data $x_1, \ldots, x_{82}$.

Example 17 (Continuation of Example 3). For the $AR(p)$ model of Example 3, the best way to include the stationarity constraints is to use the lag-polynomial representation

$$\prod_{i=1}^{p} (1 - \lambda_i B)\ X_t = \epsilon_t\,, \quad \epsilon_t \sim \mathcal{N}(0, \sigma^2)\,,$$

of model $\mathfrak{M}_p$, and to constrain the inverse roots, λ_i, to stay within the unit circle if complex and within $[-1, 1]$ if real (see, e.g. Robert 2007, Sect. 4.5.2). The associated uniform priors for the real and complex roots λ_j is

$$\pi_p(\boldsymbol{\lambda}) = \frac{1}{\lfloor p/2 \rfloor + 1} \prod_{\lambda_i \in \mathbb{R}} \frac{1}{2} \mathbb{I}_{|\lambda_i|<1} \prod_{\lambda_i \notin \mathbb{R}} \frac{1}{\pi} \mathbb{I}_{|\lambda_i|<1}\,,$$

where $\lfloor p/2 \rfloor + 1$ is the number of different values of r_p. This factor must be included within the posterior distribution when using reversible jump since it does not vanish

in the acceptance probability of a move between models $\mathfrak{M}_p$ and $\mathfrak{M}_q$. Otherwise, this results in a modification of the prior probability of each model.

Once again, a simple choice is to use a birth-and-death scheme where the birth moves either create a real or two conjugate complex roots. As in the birth-and-death proposal for Example 16, the acceptance probability simplifies quite dramatically since it is for instance

$$\min\left(\frac{\pi_{(p+1)p}}{\pi_{p(p+1)}}\,\frac{(r_p+1)!}{r_p!}\,\frac{\lfloor p/2\rfloor+1}{\lfloor (p+1)/2\rfloor+1}\,\frac{\ell_{p+1}(\boldsymbol{\theta}_{p+1})}{\ell_p(\boldsymbol{\theta}_p)},1\right)$$

in the case of a move from $\mathfrak{M}_p$ to $\mathfrak{M}_{p+1}$. (As for the above mixture example, the factorials are related to the possible choices of the created and the deleted roots.)

Figure 26.16 presents some views of the corresponding reversible jump MCMC algorithm. Besides the ability of the algorithm to explore a range of values of k, it also shows that Bayesian inference using these tools is much richer, since it can, for instance, condition on or average over the order k, mix the parameters of different models and run various tests on these parameters. A last remark on this graph is that both the order and the value of the parameters are well estimated, with a characteristic trimodality on the histograms of the $\boldsymbol{\theta}_i$'s, even when conditioning on k different from 3, the value used for the simulation.

In conclusion, while reversible jump MCMC is a generic and powerful method for handling model comparison when faced with a multitude of models, its sensitivity to the tuning "parameters" that determine the pairwise jumps makes us favour the alternative of computing Bayes factors. When the number of models under comparisons is sufficiently small to allow for the computation of all marginal likelihoods in parallel, this computational solution is more efficient. It is indeed quite rare that the structure of the posterior under a model $\mathfrak{M}_1$ provides information about the structure of the posterior under another model $\mathfrak{M}_2$, unless both models are quite close. Intrinsically, reversible jump MCMC is a random walk over the collection of models and, as such, looses in efficiency in its exploration of this collection. (In particular, it almost never makes sense to implement reversible jump MCMC when comparing a small number of models.)

26.5 More Monte Carlo Methods

While MCMC algorithms considerably expanded the range of applications of Bayesian analysis, they are not, by any means, the end of the story! Further developments are taking place, either at the fringe of the MCMC realm or far away from it. We indicate below a few of the directions in Bayesian computational Statistics, omitting many more that also are of interest...

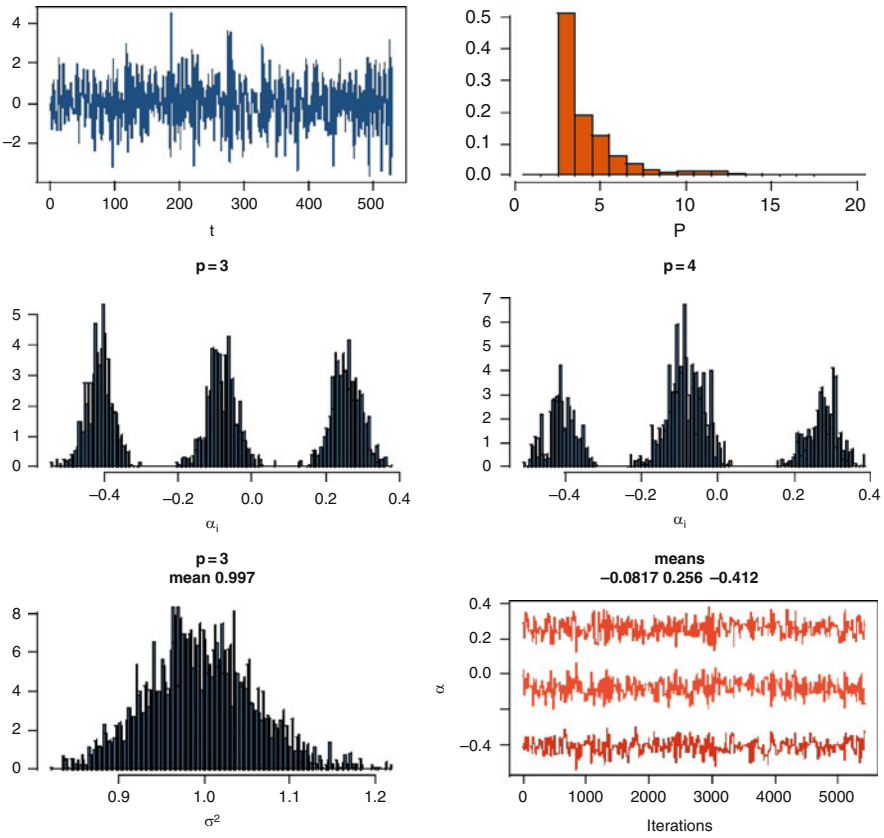


Fig. 26.16 Output of a reversible jump algorithm based on an $AR(3)$ simulated dataset of 530 points *(upper left)* with true parameters $\boldsymbol{\theta}_i$ $(-0.1, 0.3, -0.4)$ and $\sigma = 1$. The first histogram is associated with k, the following histograms are associated with the $\boldsymbol{\theta}_i$'s, for different values of k, and of σ^2. The final graph is a scatterplot of the complex roots (for iterations where there were complex roots). The one before last graph plots the evolution over the iterations of $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \boldsymbol{\theta}_3$ *(Source: Robert 2003)*

26.5.1 Adaptivity for MCMC Algorithms

Given the range of situations where MCMC applies, it is unrealistic to hope for a *generic* MCMC sampler that would function in every possible setting. The more generic proposals like random-walk Metropolis–Hastings algorithms are known to fail in large dimension and disconnected supports, because they take too long to explore the space of interest (Neal 2003). The reason for this impossibility theorem is that, in realistic problems, the complexity of the distribution to simulation is the very reason why MCMC is used! So it is difficult to ask for a prior opinion about this distribution, its support or the parameters of the proposal distribution used in the MCMC algorithm: intuition is close to void in most of these problems.

However, the performances of off-the-shelve algorithms like the random-walk Metropolis–Hastings scheme bring information about the distribution of interest and, as such, should be incorporated in the design of better and more powerful algorithms. The problem is that we usually miss the time to train the algorithm on these previous performances and are looking for the Holy Grail of automated MCMC procedures! While it is natural to think that the information brought by the first steps of an MCMC algorithm should be used in later steps, there is a severe catch: using the whole past of the "chain" implies that this is not a Markov chain any longer. Therefore, usual convergence theorems do not apply and the validity of the corresponding algorithms is questionable. Further, it may be that, in practice, such algorithms do degenerate to point masses because of a too rapid decrease in the variation of their proposal.

Example 18 (Continuation of Example 8). For the t-distribution sample, we could fit a normal proposal from the empirical mean and variance of the previous values of the chain,

$$\mu_t = \frac{1}{t}\sum_{i=1}^{t} \boldsymbol{\theta}^{(i)} \quad \text{and} \quad \sigma_t^2 = \frac{1}{t}\sum_{i=1}^{t} (\boldsymbol{\theta}^{(i)} - \mu_t)^2 \,.$$

This leads to a Metropolis–Hastings algorithm with acceptance probability

$$\prod_{j=2}^{n} \left[\frac{\nu + (x_j - \boldsymbol{\theta}^{(t)})^2}{\nu + (x_j - \xi)^2} \right]^{-(\nu+1)/2} \frac{\exp -(\mu_t - \boldsymbol{\theta}^{(t)})^2/2\sigma_t^2}{\exp -(\mu_t - \xi)^2/2\sigma_t^2} \,,$$

where ξ is the proposed value from $\mathcal{N}(\mu_t, \sigma_t^2)$. The invalidity of this scheme (because of the dependence on the whole sequence of $\boldsymbol{\theta}^{(i)}$'s till iteration t) is illustrated in Fig. 26.17: when the range of the initial values is too small, the sequence of $\boldsymbol{\theta}^{(i)}$'s cannot converge to the target distribution and concentrates on too small a support. But the problem is deeper, because even when the range of the simulated values is correct, the (long-term) dependence on past values modifies the distribution of the sequence. Figure 26.18 shows that, for an initial variance of 2.5, there is a bias in the histogram, even after 25,000 iterations and stabilisation of the empirical mean and variance.

Even though the Markov chain is converging *in distribution* to the target distribution (when using a proper, i.e. time-homogeneous updating scheme), using past simulations to create a non-parametric approximation to the target distribution does not work either. Figure 26.19 shows for instance the output of an adaptive scheme in the setting of Example 18 when the proposal distribution is the Gaussian kernel based on earlier simulations. A very large number of iterations is not sufficient to reach an acceptable approximation of the target distribution.

The overall message is thus that, without further guidance, one should not *constantly* adapt the proposal distribution on the past performances of the simulated

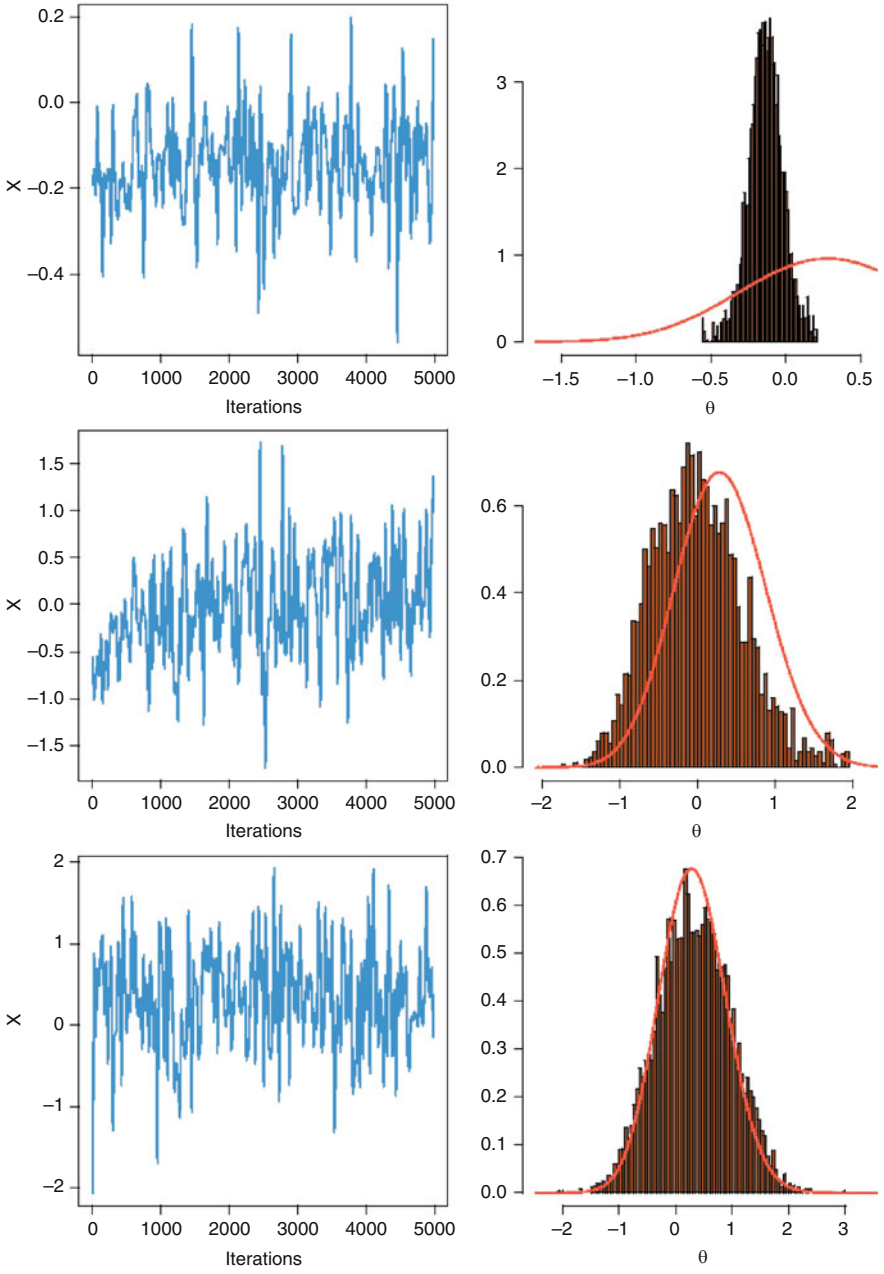


Fig. 26.17 Output of the adaptive scheme for the t-distribution posterior with a sample of 10 $x_j \sim \mathcal{T}_3$ and initial variances of *(top)* 0.1, *(middle)* 0.5, and *(bottom)* 2.5. The left column plots the sequence of $\boldsymbol{\theta}^{(i)}$'s while the right column compares its histogram against the true posterior distribution *(with a different scale for the upper graph)*

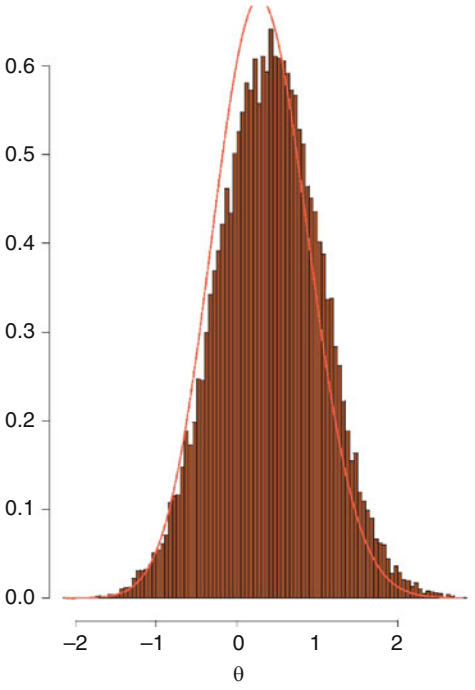


Fig. 26.18 Comparison of the distribution of an adaptive scheme sample of 25,000 points with initial variance of 2.5 and of the target distribution

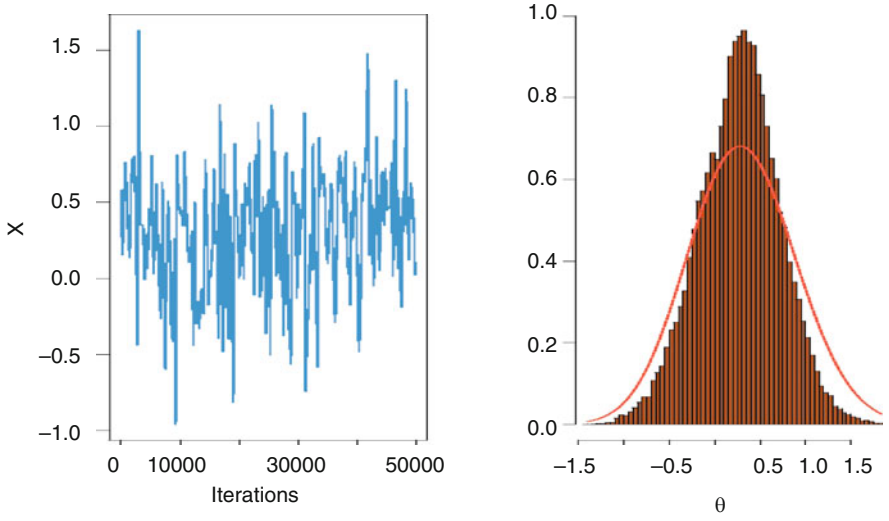


Fig. 26.19 Sample produced by 50,000 iterations of a nonparametric adaptive MCMC scheme and comparison of its distribution with the target distribution

chain. Either the adaptation must cease after a period of *burnin* (not to be taken into account for the computations of expectations and quantities related to the target distribution). Else, the adaptive scheme must be theoretically assess on its own right. This later path is not easy and only a few examples can be found (so far) in the literature. See, e.g., Gilks et al. (1998) who use regeneration to create block independence and preserve Markovianity on the paths rather than on the values, Haario et al. (1999, 2001) who derive a proper adaptation scheme in the spirit of Example 18 by using a ridge-like correction to the empirical variance, and Andrieu and Robert (2001) who propose a more general framework of valid adaptivity based on stochastic optimisation and the Robbin-Monro algorithm. A more generic perspective is found in Roberts and Rosenthal (2009), who tune the random walk scale in each direction of the parameter space toward an optimal acceptance rate of 0.44, a rate suggested in Gelman et al. (1996). Roberts and Rosenthal (2009) provide validated constraints on the tuning scheme to the extent of offering an R package called `amcmc` and described in Rosenthal (2007). More precisely, for each component of the simulated parameter, a factor δ_i corresponding to the logarithm of the random walk standard deviation is updated every 50 iterations by adding or subtracting a factor ϵ_t depending on whether or not the average acceptance rate on that batch of 50 iterations and for this component was above or below 0.44. If ϵ_t decreases to zero as $\min(0.01, 1/\sqrt{t})$, the conditions for convergence are satisfied. (See Robert and Casella (2009, Sect. 8.5.2, for more details.)

26.5.2 Population Monte Carlo

To reach acceptable adaptive algorithms, while avoiding an extended study of their theoretical properties, a better alternative is to leave the setup of Markov chain simulations and to consider instead *sequential* or *population* Monte Carlo methods (Cappé et al. 2004; Iba 2000) that have much more in common with importance sampling than with MCMC. They are inspired from *particle systems* that were introduced to handle rapidly changing target distributions like those found in signal processing and imaging (Del Moral et al. 2006; Doucet et al. 2001; Gordon et al. 1993; Shephard and Pitt 1997) but primarily handle fixed but complex target distributions by building a sequence of increasingly better proposal distributions.[17] Each iteration of the population Monte Carlo (PMC) algorithm thus produces a sample approximately simulated from the target distribution but the iterative structure allows for adaptivity toward the target distribution. Since the validation is based on importance sampling principles, dependence on the past samples can be arbitrary *and* the approximation to the target is valid (unbiased) at *each iteration* and does not require convergence times nor stopping rules.

[17]The "sequential" denomination in the sequential Monte Carlo methods thus refers to the algorithmic part, not to the statistical part.

If t indexes the iteration and i the sample point, consider proposal distributions q_{it} that simulate the $x_i^{(t)}$'s and associate to each $x_i^{(t)}$ an importance weight

$$\varrho_i^{(t)} = \pi(x_i^{(t)}) \big/ q_{it}(x_i^{(t)}) \,, \qquad i = 1, \ldots, n \,.$$

(The proposal distribution thus depends both on the iteration and on the particle index.) Approximations of the form

$$\mathfrak{I}_t = \frac{1}{n} \sum_{i=1}^{n} \varrho_i^{(t)} \, h(x_i^{(t)})$$

are then unbiased estimators of $\mathbb{E}^\pi[h(X)]$, even when the importance distribution q_{it} depends on the entire past of the experiment. Indeed, if ζ denotes the vector of past random variates that contribute to q_{it}, and $g(\zeta)$ its *arbitrary* distribution, we have

$$\int\int \frac{\pi(x)}{q_{it}(x|\zeta)} \, h(x) q_{it}(x) \mathrm{d}x \, g(\zeta) \mathrm{d}\zeta = \int\int h(x)\pi(x) \mathrm{d}x \, g(\zeta) \mathrm{d}\zeta = \mathbb{E}^\pi[h(X)] \,.$$

Furthermore, assuming that the variances

$$\mathrm{var}\left(\varrho_i^{(t)} h(x_i^{(t)})\right)$$

exist for every $1 \leq i \leq n$, we have

$$\mathrm{var}\,(\mathfrak{I}_t) = \frac{1}{n^2} \sum_{i=1}^{n} \mathrm{var}\left(\varrho_i^{(t)} h(x_i^{(t)})\right) \,,$$

due to the cancelling effect of the weights $\varrho_i^{(t)}$.

Since, usually, the density π is unscaled, we use instead

$$\varrho_i^{(t)} \propto \frac{\pi(x_i^{(t)})}{q_{it}(x_i^{(t)})} \,, \qquad i = 1, \ldots, n \,,$$

scaled so that the $\varrho_i^{(t)}$'s sum up to 1. In this case, the unbiasedness is lost, although it approximately holds. In fact, the estimation of the normalising constant of π improves with each iteration t, since the overall average

$$\varpi_t = \frac{1}{tn} \sum_{\tau=1}^{t} \sum_{i=1}^{n} \frac{\pi(x_i^{(\tau)})}{q_{i\tau}(x_i^{(\tau)})}$$

is convergent. Therefore, as t increases, ϖ_t contributes less and less to the variability of $\mathfrak{I}_t$.

However, Douc et al. (2007a) have shown that this use of the importance weights $\varrho_i^{(t)}$ in Cappé et al. (2004) was not adaptive enough and they proposed a Rao–Blackwellised substitute[18]

$$\varrho_i^{(t)} \propto \frac{\pi(x_i^{(t)})}{\sum_j q_{jt}(x_i^{(t)})}\,, \qquad i = 1, \ldots, n\,,$$

that guaranteed an (asymptotic in n) improvement of the proposal at each iteration t, under specific forms of the q_{it}'s (Cappé et al. 2008; Douc et al. 2007a,b)

Since the above establishes that an simulation scheme based on sample dependent proposals is fundamentally a specific kind of importance sampling, the following algorithm is validated by the same principles as regular importance sampling:

– Population Monte Carlo Algorithm –

For $t = 1, \ldots, T$

1. For $i = 1, \ldots, n$,
   (i) Select the generating distribution $q_{it}(\cdot)$
   (ii) Generate $x_i^{(t)} \sim q_{it}(x)$
   (iii) Compute $\varrho_i^{(t)} = \pi(x_i^{(t)}) / \sum_j q_{jt}(x_i^{(t)})$
2. Normalise the $\varrho_i^{(t)}$'s to sum up to 1
3. Resample n values from the $x_i^{(t)}$'s with replacement, using the weights $\varrho_i^{(t)}$, to create the sample $(x_1^{(t)}, \ldots, x_n^{(t)})$

Step (i) is singled out because it is the central property of the PMC algorithm, namely that adaptivity can be extended to the individual level and that the q_{it}'s can be picked based on the performances of the previous $q_{i(t-1)}$'s or even on all the previously simulated samples, if storage allows. For instance, the q_{it}'s can include large tails proposals as in the *defensive sampling* strategy of Hesterberg (1995), to ensure finite variance. Similarly, Warnes' (2001) non-parametric Gaussian kernel approximation can be used as a proposal.[19] (See also in Stavropoulos and Titterington 2001 the *smooth bootstrap* technique as an earlier example of PMC algorithm.)

[18]The generic Rao–Blackwellised improvement was introduced in the original MCMC paper of Gelfand and Smith (1990) and studied by Liu et al. (1994) and Casella and Robert (1996). More recent developments are proposed in Cornuet et al. (2009), in connection with adaptive algorithms like PMC.

[19]Using a Gaussian non-parametric kernel estimator amounts to (a) sampling from the $x_i^{(t)}$'s with equal weights and (b) using a normal random walk move from the selected $x_i^{(t)}$, with standard deviation equal to the bandwidth of the kernel.

A major difference between the PMC algorithm and earlier proposals in the particle system literature is that past dependent moves as those of Gilks and Berzuini (2001) and Del Moral et al. (2006) remain within the MCMC framework, with Markov transition kernels with stationary distribution equal to π.

Example 19 (Continuation of Example 14). We consider here the implementation of the PMC algorithm in the case of the the normal mixture (26.11). As in Example 15, a PMC sampler can be efficiently implemented *without* the (Gibbs) augmentation step, using normal random walk proposals based on the previous sample of $\boldsymbol{\mu} = (\mu_1, \mu_2)$'s. Moreover, the difficulty inherent to random walks, namely the selection of a "proper" scale, can be bypassed because of the adaptivity of the PMC algorithm. Indeed, the proposals can be associated with a range of variances v_k $(1 \leq k \leq K)$ ranging from, e.g., 10^3 down to 10^{-3}. At each step of the algorithm, the new variances can be selected proportionally to the performances of the scales v_k on the previous iterations. For instance, a scale can be chosen proportionally to its *non-degeneracy rate* in the previous iteration, that is, the percentage of points generated with the scale v_k that survived after resampling.[20] The weights are then of the form

$$\varrho_i \propto \frac{f\left(\mathbf{x}\left|(\mu_1)_i^{(t)}, (\mu_2)_i^{(t)}\right.\right)\pi\left((\mu_1)_i^{(t)}, (\mu_2)_i^{(t)}\right)}{\sum_{j=1}^{n}\varphi\left((\mu_1)_i^{(t)}\left|(\mu_1)_j^{(t-1)}, v_k\right.\right)\varphi\left((\mu_2)_i^{(t)}\left|(\mu_2)_j^{(t-1)}, v_k\right.\right)},$$

where $\varphi(q|s, v)$ is the density of the normal distribution with mean s and variance v at the point q.

Compared with an MCMC algorithm in the same setting (see Examples 14 and 15), the main feature of this algorithm is its ability to deal with multiscale proposals in an unsupervised manner. The upper row of Fig. 26.21 produces the frequencies of the five variances v_k used in the proposals along iterations: The two largest variances v_k most often have a zero survival rate, but sometimes experience bursts of survival. In fact, too large a variance mostly produces points that are irrelevant for the posterior distribution, but once in a while a point $\boldsymbol{\theta}_j^{(t)}$ gets close to one of the modes of the posterior. When this occurs, the corresponding ϱ_j is large and $\boldsymbol{\theta}_j^{(t)}$ is thus heavily resampled. The upper right graph shows that the other proposals are rather evenly sampled along iterations. The influence of the variation in the proposals on the estimation of the means μ_1 and μ_2 can be seen on the middle and lower panels of Fig. 26.21. First, the cumulative averages quickly stabilise over iterations, by virtue of the general importance sampling proposal. Second, the corresponding variances take longer to stabilise but this is to be expected, given the regular reappearance of subsamples with large variances.

[20] When the survival rate of a proposal distribution is null, in order to avoid the complete removal of a given scale v_k, the corresponding number r_k of proposals with that scale is set to a positive value, like 1% of the sample size.

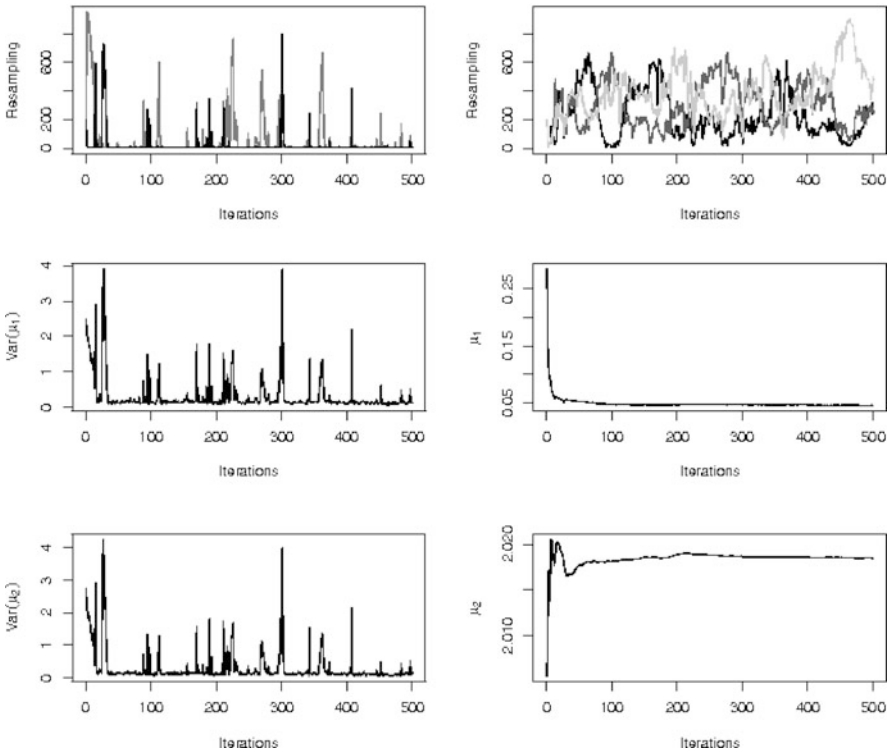


Fig. 26.20 Histograms of the PMC sample: sample at iteration 5 *(left)* before resampling and *(right)* after resampling

In comparison with Figs. 26.11 and 26.12, Fig. 26.22 shows that the sample produced by the PMC algorithm is quite in agreement with the modal zone of the posterior distribution. The second mode, which is much lower, is not preserved in the sample after the first iteration. Figure 26.20 also shows that the weights are quite similar, with no overwhelming weight in the sample.

The generality in the choice of the proposal distributions q_{it} is obviously due to the abandonment of the MCMC framework. The difference with an MCMC framework is not simply a theoretical advantage: as seen in Sect. 26.5.1, proposals based on the whole past of the chain do not often work. Even algorithms validated by MCMC steps may have difficulties: in one example of Cappé et al. (2004), a Metropolis–Hastings scheme does not work well, while a PMC algorithm based on the same proposal produces correct answers. Population Monte Carlo algorithms offer a theoretically valid solution to the adaptivity issue, with practical gains as well, as exemplified in Wraith et al. (2009). The connection with nonparametric Bayes estimation has not yet been sufficiently pursued but the convergence of kernel-like proposals demonstrated by Cappé et al. (2008) shows this is a promising direction.

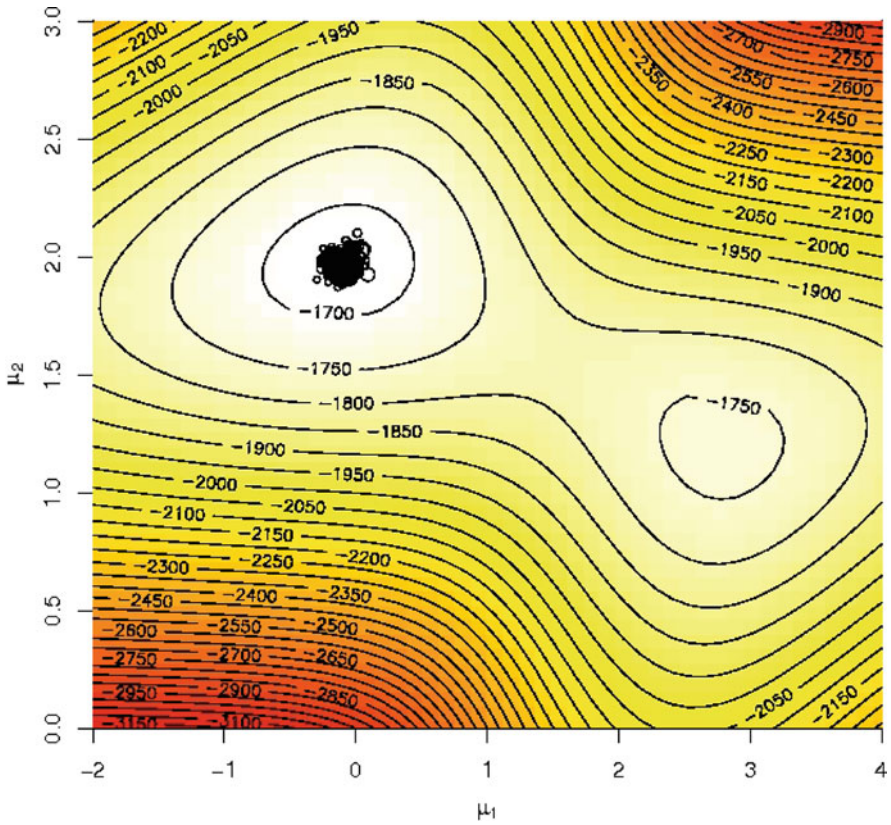


Fig. 26.21 Performances of the mixture PMC algorithm for 1,000 observations from a $0.2\mathcal{N}(0,1) + 0.8\mathcal{N}(2,1)$ distribution, with $\boldsymbol{\theta} = 1\ \lambda = 0.1$, $v_k = 5, 2, 0.1, 0.05, 0.01$, and a population of 1,050 particles: *(upper left)* Number of resampled points for the variances $v_1 = 5$ (darker) and $v_2 = 2$; *(upper right)* Number of resampled points for the other variances, $v_3 = 0.1$ is the darkest one; *(middle left)* Variance of the simulated μ_1's along iterations; *(middle right)* Cumulated average of the simulated μ_1's over iterations; *(lower left)* Variance of the simulated μ_2's along iterations; *(lower right)* Cumulated average of the simulated μ_2's over iterations *(Source: Cappé et al. 2004)*

26.5.3 Approximate Bayesian Computation

One particular application of the Accept–Reject algorithm that has found a niche of its own in population genetics is called approximate Bayesian computation (ABC), following the denomination proposed by Pritchard et al. (1999). It is in fact an appealing substitute for "exact" (meaning convergent) Bayesian algorithms

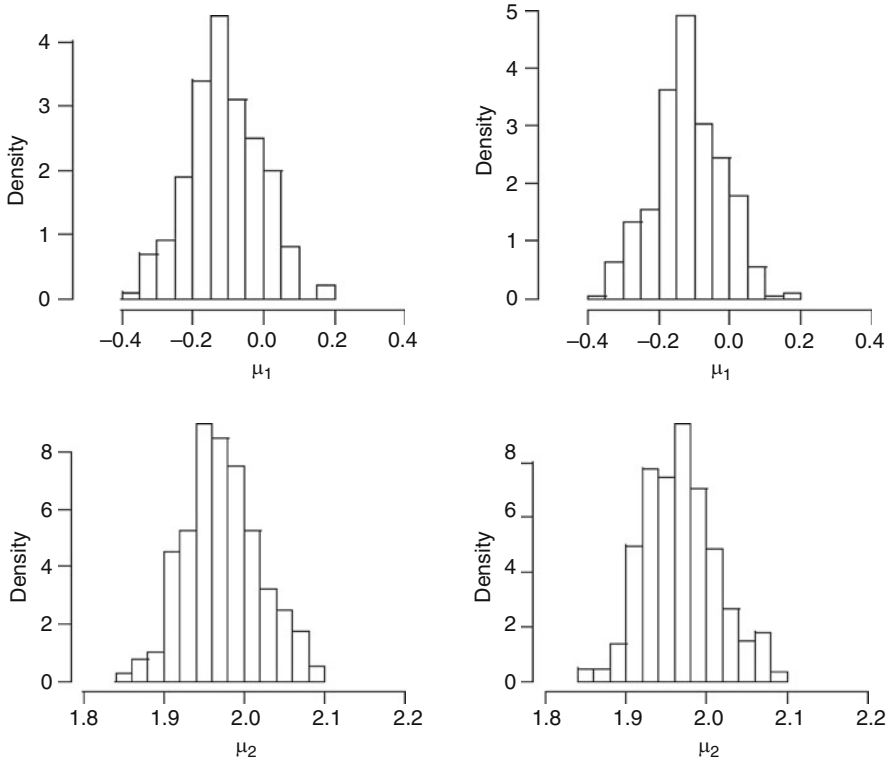


Fig. 26.22 Representation of the log-posterior distribution with the PMC weighted sample after 30 iterations (the weights are proportional to the circles at each point) *(Source: Cappé et al. 2004)*

in settings where the likelihood function $f(x|\boldsymbol{\theta})$ is not available, even up to a normalising constant, but where it can easily be simulated. The range of applications is thus much wider than population genetics and covers for instance graphical models and inverse problems.

Assuming, thus, that a posterior distribution $\pi(\boldsymbol{\theta}|x_0) \propto \pi(\boldsymbol{\theta}) f(x_0|\boldsymbol{\theta})$ is to be simulated, a rudimentary accept-reject algorithm generates values from the prior and from the likelihood until the simulated observation is equal to the original observation x_0:

Repeat

Generate $\boldsymbol{\theta} \sim \pi(\boldsymbol{\theta})$ and $X \sim f(x|\boldsymbol{\theta})$

until $X = x_0$

Since the conditional probability of acceptance is $f(x_0|\boldsymbol{\theta})$, the distribution of the accepted $\boldsymbol{\theta}$ is truly $\pi(\boldsymbol{\theta}|x_0)$, even when $f(x|\boldsymbol{\theta})$ cannot be computed.

The above algorithm is valid, but it is unfortunately restricted to settings where (a) $\pi(\boldsymbol{\theta})$ is a proper prior and (b) $\Pr_{\boldsymbol{\theta}}(X = x_0)$ has a positive probability of occurrence. Even in population genetics where the outcome X is a discrete

random variable, the size of the state-space is often such that this algorithm cannot be implemented. The proposal of Pritchard et al. (1999) is to replace the *exact* acceptance condition $X = x_0$ in the above with an *approximate* condition $d(X, x_0) < \epsilon$, where d is a distance and ϵ is a tolerance level. The corresponding ABC algorithm is thus:

– Approximate Bayesian computation Algorithm –

For $i = 1, \ldots, n$,

1 Generate $\boldsymbol{\theta}_i \sim \pi(\boldsymbol{\theta})$
2 Generate $x_i \sim f(x|\boldsymbol{\theta}_i)$ and compute $d(x_0, x_i)$

Accept the $\boldsymbol{\theta}_i$'s such that $d(x_i, x) < \epsilon$.

The output of the ABC algorithm is distributed from the distribution with density proportional to $\pi(\boldsymbol{\theta})\,\mathrm{Pr}_{\boldsymbol{\theta}}\{d(X, x_0) < \epsilon\}$, where $\mathrm{Pr}_{\boldsymbol{\theta}}$ represents the sampling distribution of X, indexed by $\boldsymbol{\theta}$. This density is somehow mistakenly denoted by $\pi\{\boldsymbol{\theta} \mid d(x, x_0) < \epsilon\}$, where the conditioning corresponds to the marginal distribution of $d(x, x_0)$ given x_0. While unavoidable, this inherent approximation step makes the ABC method difficult to assess and to compare with regular Monte Carlo approaches, even though recent works replace it within a non-parametric framework that aims at approximating the conditional density function $f(x|\boldsymbol{\theta})$ and hence envision ϵ as a potential bandwidth (Beaumont et al. 2002; Blum and François 2010).

Improvements on the original scheme have been achieved either by modifying the proposal distribution of the parameter $\boldsymbol{\theta}$, in order to increase the density of x's within the vicinity of y (Bortot et al. 2007; Marjoram et al. 2003), or, as stated above, by envisioning the problem as a conditional density estimation issue and by developing techniques to allow for larger ϵ (Beaumont et al. 2002). For instance, Marjoram et al. (2003) defined a Markov chain Monte Carlo (MCMC) version of the ABC algorithm that enjoys the same validity as the original, namely that, if a Markov chain $(\boldsymbol{\theta}^{(t)})$ is created via the transition function

$$\boldsymbol{\theta}^{(t+1)} = \begin{cases} \boldsymbol{\theta}' \sim K(\boldsymbol{\theta}' \mid \boldsymbol{\theta}^{(t)}) & \text{if } x \sim f(x \mid \boldsymbol{\theta}') \text{ is such that } x = x_0 \\ & \text{and } u \sim \mathcal{U}(0,1) \leq \dfrac{\pi(\boldsymbol{\theta}')K(\boldsymbol{\theta}^{(t)} \mid \boldsymbol{\theta}')}{\pi(\boldsymbol{\theta}^{(t)})K(\boldsymbol{\theta}' \mid \boldsymbol{\theta}^{(t)})}\,, \\ \boldsymbol{\theta}^{(t)} & \text{otherwise,} \end{cases}$$

its stationary distribution is the posterior $\pi(\boldsymbol{\theta} \mid x_0)$. Again, in most settings, exact equality is not achievable and the above constraint $x = x_0$ is replaced with the approximation $d(x, x_0) < \epsilon$. In Beaumont et al. (2009), a population Monte Carlo modification of ABC is introduced, resulting into the algorithm:

– PMC-ABC Algorithm –

Given a decreasing sequence of tolerance thresholds $\epsilon_1 \geq \ldots \geq \epsilon_T$,

1. At iteration $t = 1$,

   For $i = 1, ..., N$
   Simulate $\boldsymbol{\theta}_i^{(1)} \sim \pi(\boldsymbol{\theta})$ and $x \sim f(x \mid \boldsymbol{\theta}_i^{(1)})$ until $\varrho(x, y) < \epsilon_1$
   Set $\omega_i^{(1)} = 1/N$

   Take τ_1^2 as twice the empirical variance of the $\boldsymbol{\theta}_i^{(1)}$'s
2. At iteration $2 \leq t \leq T$,

   For $i = 1, ..., N$, repeat

   Pick $\boldsymbol{\theta}_i^\star$ from the $\boldsymbol{\theta}_j^{(t-1)}$'s with probabilities $\omega_j^{(t-1)}$
   generate $\boldsymbol{\theta}_i^{(t)} \mid \boldsymbol{\theta}_i^\star \sim \mathcal{N}(\boldsymbol{\theta}_i^\star, \tau_t^2)$ and $x \sim f(x \mid \boldsymbol{\theta}_i^{(t)})$

   until $\varrho(x, y) < \epsilon_t$
   Set $\omega_i^{(t)} \propto \pi(\boldsymbol{\theta}_i^{(t)}) / \sum_{j=1}^N \omega_j^{(t-1)} \varphi\left\{\tau_t^{-1}\left(\boldsymbol{\theta}_i^{(t)} - \boldsymbol{\theta}_j^{(t-1)}\right)\right\}$

   Take τ_{t+1}^2 as twice the weighted empirical variance of the $\boldsymbol{\theta}_i^{(t)}$'s

From a practical viewpoint, the number of iterations T can be controlled via the modifications in the parameters of K_t, a stopping rule being that the iterations should stop when those parameters have settled, while the more fundamental issue of selecting a sequence of ϵ_t's towards a proper approximation of the true posterior can rely on the stabilisation of the estimators of some quantities of interest associated with this posterior. But there is generally no fixed-cost solution that let ϵ_t decrease to zero with t. Note also that the SMC algorithm of Del Moral et al. (2006) has been recently extended to this ABC setup, making the resulting algorithm a direct competitor of the above PMC-ABC algorithm.

26.6 Conclusion

This short overview of the problems and solutions considered for Bayesian Statistics is nothing but an introduction to the game: there are much more complex problems than those illustrated above and much more advanced techniques than those presented in these pages. The reader is then encouraged to enter the literature on the topic, maybe with other introductory surveys like Cappé and Robert (2000) and Andrieu et al. (2004), but mostly through books like Chen et al. (2000), Doucet et al. (2001), Liu (2001), Robert and Casella (2004), Albert (2007), Marin and Robert (2007a), and Robert and Casella (2009).

We have not mentioned so far entries to Bayesian softwares like winBUGS, developed by the MRC Unit in Cambridge (Gilks et al. 1994; Spiegelhalter et al. 1999), Ox (Doornik et al. 2002), BATS (Pole et al. 1994), BACC (Geweke 1999) and the Minitab package of Albert (1996), which all cover some aspects of Bayesian computing. Obviously, these packages require some expertise from the user and are thus more difficult of use than the classical open source or commercial softwares like R, Splus, Statgraphics, StatXact, SPSS or SAS. In other words, they are not *black boxes* that could be used by laymen with no statistical background. But this entrance fee to the use of Bayesian softwares is inevitable, given the versatile nature of Bayesian analysis: since it offers much more variability than standard inferential procedures, through the choice of prior distributions and loss functions for instance, it also requires more input from the user! And, once these preliminary steps have been overcome, the programming involved in a software like winBUGS is rather limited and certainly not harder than writing a code in R or Matlab.[21]

As stressed in this Chapter, computational issues are central to the design and implementation of Bayesian analysis. The new era opened by the MCMC methodology has brought much more freedom in the use of Bayesian methods, as reflected by the increase of Bayesian studies in applied Statistics. As usually the case, a strong increase in the use of a methodology also sees a corresponding increase in its misuse! Inconsistent data-dependent priors and improper posteriors are sometimes appearing in studies and, more generally, the assessment of prior modelling (or even of MCMC convergence) are rarely conducted with sufficient care. This is somehow a price to pay for the wider range of Bayesian studies, while the improvement of corresponding software should bring more guidelines and warnings about these misuses of Bayesian analysis.

Acknowledgements This work had been partly supported by the Agence Nationale de la Recherche (ANR, 212, rue de Bercy 75,012 Paris) through the 2009–2012 projects Big'MC and EMILE.

References

Abowd, J., Kramarz, F., Margolis, D.: High-wage workers and high-wage firms. Econometrica **67**, 251–333 (1999)

Albert, J.: Bayesian Computation Using Minitab. Wadsworth Publishing Company (1996)

Albert, J.H.: Bayesian Computation with R. Springer, New York, (2007)

Andrieu, C., Robert, C.P.: Controlled Markov chain Monte Carlo methods for optimal sampling. Technical Report 0125, Université Paris Dauphine (2001)

Andrieu, C., Doucet, A., Robert, C.P.: Computational advances for and from Bayesian analysis. Stat. Sci. **19**(1), 118–127 (2004)

[21] An R package called `mcsm` has been developed in association with Robert and Casella (2009) for training about Monte Carlo methods.

Bauwens, L., Richard, J.F.: A 1-1 Poly-t random variable generator with application to Monte Carlo integration. J. Econometrics **29**, 19–46 (1985)
Beaumont, M.A., Zhang, W., Balding, D.J.: Approximate Bayesian computation in population genetics. Genetics **162**, 2025–2035 (2002)
Beaumont, M.A., Cornuet, J.-M., Marin, J.-M., Robert, C.P.: Adaptive approximate Bayesian computation. Biometrika **96**(4), 983–990 (2009)
Berkhof, J., van Mechelen, I., Gelman, A.: A Bayesian approach to the selection and testing of mixture models. Statistica Sinica **13**, 423–442 (2003)
Blum, M.G.B., François, O.: Non-linear regression models for approximate Bayesian computation. Stat. Comput. **20**, 63–73 (2010)
Bortot, P., Coles, S.G., Sisson, S.A.: Inference for stereological extremes. J. Am. Stat. Assoc. **102**, 84–92 (2007)
Cappé, O., Robert, C.P.: MCMC: Ten years and still running! J. Am. Stat. Assoc. **95**(4), 1282–1286 (2000)
Cappé, O., Guillin, A., Marin, J.-M., Robert, C.P.: Population Monte Carlo. J. Comput. Graph. Stat. **13**(4), 907–929 (2004)
Cappé, O., Moulines, E., Rydén, T.: Inference in Hidden Markov Models. Springer, New York (2005)
Cappé, O., Douc, R., Guillin, A., Marin, J.-M., Robert, C.P.: Adaptive importance sampling in general mixture classes. Stat. Comput. **18**, 447–459 (2008)
Carlin, B.P., Chib, S.: Bayesian model choice through Markov chain Monte Carlo. J. Roy. Stat. Soc. B. **57**(3), 473–484 (1995)
Casella, G., Robert, C.P.: Rao-Blackwellisation of sampling schemes. Biometrika **83**(1), 81–94 (1996)
Celeux, G., Hurn, M.A., Robert, C.P.: Computational and inferential difficulties with mixtures posterior distribution. J. Am. Stat. Assoc. **95**(3), 957–979 (2000)
Chen, M.H., Shao, Q.M., Ibrahim, J.G.: Monte Carlo Methods in Bayesian Computation. Springer, New York (2000)
Chib, S.: Marginal likelihood from the Gibbs output. J. Am. Stat. Assoc. **90**, 1313–1321 (1995)
Chopin, N.: Inference and model choice for time-ordered hidden Markov models. J. Roy. Stat. Soc. B. **69**(2), 269–284 (2007)
Chopin, N., Robert, Cp.P.: Properties of nested sampling. Biometrika (2010); To appear, see arXiv:0801.3887. **97**, 741–755
Cornuet, J.-M., Marin, J.-M., Mira, A., Robert, C.P.: Adaptive multiple importance sampling. Technical Report arXiv.org:0907.1254, CEREMADE, Université Paris, Dauphine (2009)
Del Moral, P., Doucet, A., Jasra, A.: Sequential Monte Carlo samplers. J. Roy. Stat. Soc. B. **68**(3), 411–436 (2006)
Dickey, J.M.: The weighted likelihood ratio, linear hypotheses on normal location parameters. Ann. Math. Stat. **42**, 204–223 (1971)
Diebolt, J., Robert, C.P.: Estimation of finite mixture distributions by Bayesian sampling. J. Roy. Stat. Soc. B. **56**, 363–375 (1994)
Doornik, J.A., Hendry, D.F., Shephard, N.: Computationally-intensive econometrics using a distributed matrix-programming language. Philo. Trans. Roy. Soc. London **360**, 1245–1266 (2002)
Douc, R., Guillin, A., Marin, J.-M., Robert, C.P.: Convergence of adaptive mixtures of importance sampling schemes. Ann. Stat. **35**(1), 420–448 (2007a)
Douc, R., Guillin, A., Marin, J.-M., Robert, C.P.: Minimum variance importance sampling via population Monte Carlo. ESAIM: Probab. Stat. **11**, 427–447 (2007b)
Doucet, A., de Freitas, N., Gordon, N.: Sequential Monte Carlo Methods in Practice. Springer, New York (2001)
Frühwirth-Schnatter, S.: Markov chain Monte Carlo estimation of classical and dynamic switching and mixture models. J. Am. Stat. Assoc. **96**(453), 194–209 (2001)
Frühwirth-Schnatter, S.: Estimating marginal likelihoods for mixture and Markov switching models using bridge sampling techniques. Econometrics J. **7**(1), 143–167 (2004)

Frühwirth-Schnatter, S.: Finite Mixture and Markov Switching Models. Springer, New York (2006)
Gelfand, A.E., Smith, A.F.M.: Sampling based approaches to calculating marginal densities. J. Am. Stat. Assoc. **85**, 398–409 (1990)
Gelman, A., Gilks, W.R., Roberts, G.O.: Efficient Metropolis jumping rules. In: Berger, J.O., Bernardo, J.M., Dawid, A.P., Lindley, D.V., Smith, A.F.M. (eds.) Bayesian Statistics 5, pp. 599–608. Oxford University Press, Oxford (1996)
Geweke, J.: Using simulation methods for Bayesian econometric models: Inference, development, and communication (with discussion and rejoinder). Econom. Rev. **18**, 1–126 (1999)
Geweke, J.: Interpretation and inference in mixture models: Simple MCMC works. Comput. Stat. Data Anal. **51**(7), 3529–3550 (2007)
Gilks, W.R., Berzuini, C.: Following a moving target–Monte Carlo inference for dynamic Bayesian models. J. Roy. Stat. Soc. B. **63**(1), 127–146 (2001)
Gilks, W.R., Thomas, A., Spiegelhalter, D.J.: A language and program for complex Bayesian modelling. The Statistician **43**, 169–178 (1994)
Gilks, W.R., Roberts, G.O., Sahu, S.K.: Adaptive Markov chain Monte Carlo. J. Am. Stat. Assoc. **93**, 1045–1054 (1998)
Gordon, N., Salmond, J., Smith, A.F.M.: A novel approach to non-linear/non-Gaussian Bayesian state estimation. IEEE Proceedings on Radar and Signal Processing **140**, 107–113 (1993)
Green, P.J.: Reversible jump MCMC computation and Bayesian model determination. Biometrika **82**(4), 711–732 (1995)
Haario, H., Saksman, E., Tamminen, J.: Adaptive proposal distribution for random walk Metropolis algorithm. Comput. Stat. **14**(3), 375–395 (1999)
Haario, H., Saksman, E., Tamminen, J.: An adaptive Metropolis algorithm. Bernoulli **7**(2), 223–242 (2001)
Hesterberg, T.: Weighted average importance sampling and defensive mixture distributions. Technometrics **37**, 185–194 (1995)
Iba, Y.: Population-based Monte Carlo algorithms. Trans. Jpn. Soc. Artif. Intell. **16**(2), 279–286 (2000)
Jasra, A., Holmes, C.C., Stephens, D.A.: Markov Chain Monte Carlo methods and the label switching problem in Bayesian mixture modeling. Stat. Sci. **20**(1), 50–67 (2005)
Jeffreys, H.: Theory of Probability. Oxford Classic Texts in the Physical Sciences. (3rd edn.), Oxford University Press, Oxford (1961)
Lee, K., Marin, J.-M., Mengersen, K.L., Robert, C.P.: Bayesian inference on mixtures of distributions. In: Narasimha Sastry, N.S., Delampady, M., Rajeev, B. (eds.) Perspectives in Mathematical Sciences I: Probability and Statistics, pp. 165–202. World Scientific, Singapore (2009)
Liu, J.S.: Monte Carlo Strategies in Scientific Computing. Springer, New York (2001)
Liu, J.S., Wong, W.H., Kong, A.: Covariance structure of the Gibbs sampler with applications to the comparisons of estimators and sampling schemes. Biometrika **81**, 27–40 (1994)
Marin, J.-M., Robert, C.P.: Bayesian Core. Springer, New York (2007a).
Marin, J.-M., Robert, C.P.: Importance sampling methods for Bayesian discrimination between embedded models. In: Chen, M.-H., Dey, D.K., Müller, P., Sun, D., Ye, K. (eds.) Frontiers of Statistical Decision Making and Bayesian Analysis. Springer, New York (2007b); To appear, see arXiv:0910.2325.
Marjoram, P., Molitor, J., Plagnol, V., Tavaré, S.: Markov chain Monte Carlo without likelihoods. Proc. Natl. Acad. Sci. USA **100**(26), 15324–15328 (2003)
McCullagh, P., Nelder, J.: Generalized Linear Models. Chapman and Hall, New York (1989)
Meng, X.L., Wong, W.H.: Simulating ratios of normalizing constants via a simple identity: a theoretical exploration. Stat. Sinica. **6**, 831–860 (1996)
Metropolis, N., Ulam, S.: The Monte Carlo method. J. Am. Stat. Assoc. **44**, 335–341 (1949)
Neal, R.M.: Slice sampling (with discussion). Ann. Stat. **31**, 705–767 (2003)
Nobile, A.: A hybrid Markov chain for the Bayesian analysis of the multinomial probit model. Stat. Comput. **8**, 229–242 (1998)

Pole, A., West, M., Harrison, P.J.: Applied Bayesian Forecasting and Time Series Analysis. Chapman-Hall, New York (1994)

Pritchard, J.K., Seielstad, M.T., Perez-Lezaun, A., Feldman, M.W.: Population growth of human Y chromosomes: a study of Y chromosome microsatellites. Mol. Biol. Evol. **16**, 1791–1798 (1999)

Richardson, S., Green, P.J.: On Bayesian analysis of mixtures with an unknown number of components (with discussion). J. Roy. Stat. Soc. B. **59**, 731–792 (1997)

Robert, C.P.: The Bayesian Choice. paperback edn, Springer, New York (2007)

Robert, C.P., Casella, G.: Monte Carlo Statistical Methods. (2nd edn.), Springer, New York (2004)

Robert, C.P., Casella, G.: Introducing Monte Carlo Methods with R. Springer, New York (2009)

Robert, C.P., Casella, G.: A history of Markov chain Monte Carlo-subjective recollections from incomplete data. In: Brooks, S., Gelman, A., Meng, X.L., Jones, G. (eds.) Handbook of Markov Chain Monte Carlo: Methods and Applications. Chapman and Hall, New York (2010); arXiv0808.2902.

Robert, C.P., Marin, J.-M.: On resolving the Savage–Dickey paradox. Technical Report arxiv.org:0910.1452, CEREMADE, Université Paris Dauphine (2009)

Robert, C.P., Wraith, D.: Computational methods for Bayesian model choice. In: Paul, M.G., Chun-Yong, C. (eds.) MaxEnt 2009 proceedings, vol. 1193, AIP (2009)

Roberts, G.O., Rosenthal, J.S.: Examples of adaptive MCMC. J. Comp. Graph. Stat. **18**, 349–367 (2009)

Roeder, K.: Density estimation with confidence sets exemplified by superclusters and voids in galaxies. J. Am. Stat. Assoc. **85**, 617–624 (1992)

Rosenthal, J.S.: Amcm: An R interface for adaptive MCMC. Comput. Stat. Data Anal. **51**, 5467–5470 (2007)

Shephard, N., Pitt, M.K.: Likelihood analysis of non-Gaussian measurement time series. Biometrika **84**, 653–668 (1997)

Skilling, J.: Nested sampling for general Bayesian computation. Bayesian Anal. **1**(4), 833–860 (2006)

Spiegelhalter, D.J., Thomas, A., Best, N.G.: WinBUGS Version 1.2 User Manual. Cambridge (1999)

Stavropoulos, P., Titterington, D.M.: Improved particle filters and smoothing. In: Doucet, A., deFreitas, N., Gordon, N. (eds.) Sequential MCMC in Practice. Springer, New York (2001)

Tanner, M., Wong, W.: The calculation of posterior distributions by data augmentation. J. Am. Stat. Assoc. **82**, 528–550 (1987)

Verdinelli, I., Wasserman, L.: Computing Bayes factors using a generalization of the Savage–Dickey density ratio. J. Am. Stat. Assoc. **90**, 614–618 (1995)

Wraith, D., Kilbinger, M., Benabed, K., Cappé, O., Cardoso, J.-F., Fort, G., Prunet, S., Robert, C.P.: Estimation of cosmological parameters using adaptive importance sampling. Phys. Rev. D. **80**, 023507 (2009)

Chapter 27
Computational Methods in Survival Analysis

Toshinari Kamakura

Survival analysis is widely used in the fields of medical science, pharmaceutics, reliability and financial engineering, and many others to analyze positive random phenomena defined by event occurrences of particular interest. In the reliability field, we are concerned with the time to failure of some physical component such as an electronic device or a machine part. This article briefly describes statistical survival techniques developed recently from the standpoint of statistical computational methods focussing on obtaining the good estimates of distribution parameters by simple calculations based on the first moment and conditional likelihood for eliminating nuisance parameters and approximation of the likelihoods. The method of partial likelihood (Cox 1972, 1975) was originally proposed from the view point of conditional likelihood for avoiding estimating the nuisance parameters of the baseline hazards for obtaining simple and good estimates of the structure parameters. However, in case of heavy ties of failure times calculating the partial likelihood does not succeed. Then the approximations of the partial likelihood have been studied, which will be described in the later section and a good approximation method will be explained. We believe that the better approximation method and the better statistical model should play an important role in lessening the computational burdens greatly.

27.1 Introduction

Let T be a positive random variable with density function $f(t)$ and distribution function $F(t)$. The survival function $S(t)$ is then defined as

$$S(t) = 1 - F(t) = \Pr\{T > t\},$$

T. Kamakura (✉)
Chuo University, Tokyo, Japan
e-mail: kamakura@indsys.chuo-u.ac.jp

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_27,

and the hazard function or hazard rate as

$$\lambda(t) = \lim_{h \to 0} \frac{\Pr\{t < T \leq t + h | T > t\}}{h} .$$

The hazard function can also be expressed as

$$\lambda(t) = \frac{f(t)}{S(t)} . \tag{27.1}$$

The right-hand side (RHS) of (27.1) becomes

$$\frac{f(t)}{S(t)} = -\frac{\mathrm{d}}{\mathrm{d}t} \log S(t) ,$$

and inversely

$$S(t) = \exp\left\{ -\int_0^t \lambda(u) \mathrm{d}u \right\} . \tag{27.2}$$

27.1.1 Nonparametric Model

We assume that the observed data set consists of failure or death times t_i and censoring indicators $\delta_i, i = 1, \cdots, n$. The indicator δ is unity for the case of failure and zero for censoring. The censoring scheme is an important concept in survival analysis in that one can observe partial information associated with the survival random variable. This is due to some limitations such as loss to follow-up, drop-out, termination of the study, and others.

The Kaplan–Meier method (Kaplan and Meier 1958) is currently the standard for estimating the nonparametric survival function. For the case of a sample without any censoring observations, the estimate exactly corresponds to the derivation from the empirical distribution. The dataset can be arranged in table form (Table 27.1), i.e.,

where, t_i is the i-th order statistic when they are arranged in ascending order for distinct failure times, d_i is the number of failures at the time of t_i, and n_i is the number of survivors at time $t_i - 0$. Under this notation the Kaplan–Meier estimate becomes

$$\widehat{S}(t) = \prod_{j : t_j < t} \left(1 - \frac{d_j}{n_j} \right) . \tag{27.3}$$

Table 27.1 Failure time data

Failure times	t_1	t_2	$\cdots$	t_i	$\cdots$	t_k
Number of failures	d_1	d_2	$\cdots$	d_i	$\cdots$	d_k
Number of individuals of risk set	n_1	n_2	$\cdots$	n_i	$\cdots$	n_k

The standard error of the Kaplan–Meier estimate is

$$\mathrm{SE}\left\{\widehat{S}(t)\right\} = \left[\widehat{S}(t)\right]\left\{\sum_{j:t_j<t}\frac{d_j}{n_j(n_j-d_j)}\right\}^{1/2}. \tag{27.4}$$

The above formula is called "Greenwood's formula" described by Greenwood (1926).

27.1.2 Parametric Models

The most important and widely-used models in survival analysis are exponential, Weibull, log-normal, log-logistic, and gamma distributions. The first two models will be introduced for later consideration. The exponential distribution is simplistic and easy to handle, being similar to a standard distribution in some respects, while the Weibull distribution is a generalization of the exponential distribution and allows inclusion of many types of shapes. Their density functions are

$$f(t;\lambda) = \lambda e^{-\lambda t} \quad (\lambda, t > 0) \tag{27.5}$$

$$f(t;m,\eta) = \frac{m}{\eta}\left(\frac{t}{\eta}\right)^{m-1}\exp\left\{-\left(\frac{t}{\eta}\right)^{m}\right\} \quad (m,\eta,t>0), \tag{27.6}$$

where the parameter λ is sometimes called the failure rate in reliability engineering. Two models may include additional threshold parameters, or guarantee times. Let γ be this threshold parameter. The Weibull density function then becomes

$$f(t;m,\eta,\gamma) = \frac{m}{\eta}\left(\frac{t-\gamma}{\eta}\right)^{m-1}\exp\left\{-\left(\frac{t-\gamma}{\eta}\right)^{m}\right\} \quad (m,\eta,\gamma,t>0)\,. \tag{27.7}$$

Here, note that in the case of $m = 1$, the Weibull probability density function is exactly the exponential density function placing $\lambda = 1/\eta$, and that we cannot observe any failure times before threshold time ($t < \gamma$) or an individual cannot die before this time.

As the Weibull distribution completely includes the exponential distribution, only the Weibull model will be discussed further. The Weibull distribution is widely used in reliability and biomedical engineering because of goodness of fit to data and ease of handling. The main objective in lifetime analysis sometimes involves (1) estimation of a few parameters which define the Weibull distribution, and (2) evaluation of the effects of some environmental factors on lifetime distribution using regression techniques. Inference on the quantiles of the distribution has been previously studied in detail (Johnson et al. 1994).

The maximum likelihood estimate (MLE) is well known, yet it is not expressed explicitly in closed form. Accordingly, some iterative computational methods are used. Menon (Menon 1963) provided a simple estimator of $1/m$, being a consistent estimate of $1/m$, with a bias that tends to vanish as the sample size increases. Later, Cohen (Cohen 1965; Cohen and Whitten 1988) presented a practically useful chart for obtaining a good first approximation to the shape parameter m using the property that the coefficient of variation of the Weibull distribution is a function of the shape parameter m, i.e., it does not depend on η. This is described as follows.

Let T be a random variable with probability density function (27.6), the rth moment around the origin is then calculated as

$$E[T^r] = \eta^r \Gamma\left(1 + \frac{r}{m}\right).$$

Here $\Gamma(\cdot)$ is the complete gamma function. From this, the first two moments obtained are the mean life and variance, i.e.,

$$E[T] = \eta\Gamma\left(1 + \frac{1}{m}\right),$$

$$\mathrm{Var}[T] = \eta^2\left\{\Gamma\left(1 + \frac{2}{m}\right) - \Gamma^2\left(1 + \frac{1}{m}\right)\right\}.$$

Considering that the coefficient of variation

$$\mathrm{CV} = \sqrt{(\mathrm{Var}[T])}/E[T]$$

does not depend on the parameter η allows obtaining simple and robust moment estimates, which may be the initial values of the maximum likelihood calculations. Dubey (1967) studied the behavior of the Weibull distribution in detail based on these moments, concluding that the Weibull distribution with shape parameter $m = 3.6$ is relatively similar to the normal distribution.

Regarding the three-parameter Weibull described by (27.7), Cohen and Whitten (1988) suggested using the method of moments equations, noting that

$$\begin{aligned} E[T] &= \gamma + \eta\Gamma_1(m), \\ \mathrm{Var}[T] &= \eta^2\left\{\Gamma_2(m) - \Gamma_1^2(m)\right\}, \\ E[X_{(1)}] &= \gamma + \frac{\eta}{n^{1/m}}\Gamma_1(m), \end{aligned}$$

and equating them to corresponding samples, where $\Gamma_r(m) = \Gamma(1 + r/m)$.

As for obtaining an inference on the parameter of the mean parameter $\mu = E(T)$, this has not yet been investigated and will now be discussed. When one would

like to estimate μ, use of either the MLE or the standard sample mean is best for considering the case of an unknown shape parameter. This is true because the asymptotic relative efficiency of the sample mean to the MLE is calculated as

$$\begin{aligned}\mathrm{ARE}(\bar{T}) &= \frac{n A\mathrm{var}(\tilde{\mu})}{n A\mathrm{var}(\bar{T})} \\ &= \frac{6}{m^2\pi^2} \cdot \frac{1}{\mathrm{CV}^2}\left[\frac{\pi^2}{6} + \{c - 1 + \psi(1 + 1/m)\}^2\right], \end{aligned} \tag{27.8}$$

where c is Euler's constant, $\psi(\cdot)$ a digamma function, $\tilde{\mu}$ the MLE, and $\bar{T}$ the sample mean.

Table 27.2 gives the ARE with respect to various values of m. Note the remarkably high efficiency of the sample mean, especially for $m \geq 0.5$, where more than 90% efficiency is indicated. The behavior of $\mathrm{ARE}(\bar{T})$ form $m > 1$ is that $\mathrm{ARE}(\bar{X})$ has a local minimum 0.9979 at $m = 1.7884$ and a local maximum 0.9986 at $m = 3.1298$, and that for the larger m, $\mathrm{ARE}(\bar{T})$ monotonically decreases in m and the infimum of $\mathrm{ARE}(\bar{T})$ is given in $m \to \infty$;

$$\lim_{m\to\infty} \mathrm{ARE}(\bar{T}) = \frac{6(\pi^2 + 6)}{\pi^4} \cong 0.9775\,. \tag{27.9}$$

When m is known and tends to infinity, the behavior of $\mathrm{ARE}(\bar{T})$ is as follows:

$$\lim_{m\to\infty} \frac{1}{(m\mathrm{CV})^2} = \frac{6}{\pi^2} \cong 0.6079\,. \tag{27.10}$$

A higher relative efficiency of the sample mean for unknown m is shown compared to known m. From a practical standpoint, the sample mean is easily calculated for a point estimation of the Weibull mean if no censored data are included. These results support the benefits of using the sample mean for the complete sample.

Table 27.2 ARE of the sample mean to the MLE

m	eff	m	eff	m	eff
0.1	0.0018	1.1	0.9997	2.1	0.9980
0.2	0.1993	1.2	0.9993	2.2	0.9981
0.3	0.5771	1.3	0.9988	2.3	0.9982
0.4	0.8119	1.4	0.9984	2.4	0.9983
0.5	0.9216	1.5	0.9981	2.5	0.9984
0.6	0.9691	1.6	0.9980	2.6	0.9984
0.7	0.9890	1.7	0.9979	2.7	0.9985
0.8	0.9968	1.8	0.9979	2.8	0.9985
0.9	0.9995	1.9	0.9979	2.9	0.9985
1.0	1.0000	2.0	0.9980	3.0	0.9986

27.2 Estimation of Shape or Power Parameter

Let us now consider the class of the lifetime distributions, whose distribution functions are expressed by

$$F(t;\alpha,\gamma,\sigma) = G\left(\left(\frac{t-\gamma}{\sigma}\right)^{\alpha}\right), \tag{27.11}$$

where $G(\cdot)$ is also a distribution function. For the Weibull model, $G(t) = 1-\exp(-t)$ is an exponential distribution. Nagatsuka and Kamakura (Nagatsuka and Kamakura 2003, 2004) proposed a new method using the location-scale-free transformation of data set to estimate the power parameter in the Castillo–Hadi model (Castillo and Hadi 1995). That is, let $T_1,\ldots,T_n$ be independently distributed according to the distribution function (27.11). Consider the W-transformation to be defined as

$$W_i = \frac{T_i - T_{(1)}}{T_{(n)} - T_{(1)}}, \quad (i = 2,\ldots,n-1), \tag{27.12}$$

where $T_{(k)}$ is the k-th order statistic of T_i's. The new random variables W_i's derived by this W-transformation are then free from location and scale parameter. The arithmetic mean of W_i's gives the approximation to the original distribution of T. Let $V_i, i = 1,\ldots,n$ be i.i.d. distributed with common distribution function $F_V(v)$, and let the i-th order statistic $V_{(i)}$ have the marginal distribution function $F_{V_{(i)}}(v)$. Then

$$F_v(v) = \frac{1}{n}\sum_{i=1}^{n} F_{V_{(i)}}(v)\,. \tag{27.13}$$

This equation indicates that the arithmetic mean of the marginal distributions of n order statistics is exactly the original distribution. In the case of the Castillo–Hadi Model, Nagatsuka and Kamakura (2004) provided a theorem regarding this approximation, i.e.,

Theorem 1. *(Nagatsuka and Kamakura 2004)*
The mixture of the marginal distributions of $W_{(i)}$, $i = 2,\ldots,n-1$:

$$F^{(n)}(w) = \frac{1}{n-2}\sum_{i=2}^{n-1} F_{W_{(i)}}(w) \tag{27.14}$$

is the approximate distribution of W_i's and the limiting distribution (27.14) is the power function distribution with parameter $1/\alpha$. That is

$$\lim_{n\to\infty}\frac{1}{n-2}\sum_{i=2}^{n-1} F_{W_{(i)}}(w) = w^{\frac{1}{\alpha}}, \quad 0 < w < 1.$$

In the case of the Weibull distribution, the marginal distribution of $W_{(i)}$ is calculated as

$$\begin{aligned}
F_{W_{(i)}}(w) &= \Pr\left(W_{(i)} \le w\right) \\
&= \Pr\left(\frac{T_{(i)} - T_{(1)}}{T_{(n)} - T_{(1)}} \le w\right) \\
&= \int_0^\infty \int_u^\infty n(n-1) f(u) f(v) \Bigg[\sum_{k=i-1}^{n-2} \binom{n-2}{k} \\
&\quad \times \{F((1-w)u + wv) - F(u)\}^k \\
&\quad \times \{F(v) - F((1-w)u + wv)\}^{n-k-2} \Bigg] \mathrm{d}v\mathrm{d}u \\
&= \int_0^1 \int_u^1 n(n-1) \sum_{k=i-1}^{n-2} \binom{n-2}{k} \times [1 - \exp\{-\alpha(w,m,u,v)\} - u]^k \\
&\quad \times [v - (1 - \exp\{-\alpha(w,m,u,v)\})]^{n-k-2} \,, \qquad (27.15)
\end{aligned}$$

where

$$\alpha(w,m,u,v) = \left[(1-w)\{-\log(1-u)\}^{\frac{1}{m}} + w\{-\log(1-v)\}^{\frac{1}{m}}\right]^m .$$

Calculations show that $F^{(n)}(w)$ has a first moment of

$$\begin{aligned}
\mu_n(m) &= \int_0^\infty \left\{1 - F^{(n)}(w)\right\} \mathrm{d}w \\
&= -\frac{1}{n-2} + \frac{n(n-1)}{m} \int_0^1 \int_u^1 (v-u)^{n-3} \\
&\quad \times \frac{\Gamma\left(\frac{1}{m}, -\log(1-u), -\log(1-v)\right)}{\{-\log(1-v)\}^{\frac{1}{m}} - \{-\log(1-u)\}^{\frac{1}{m}}} \mathrm{d}v\mathrm{d}u \,. \qquad (27.16)
\end{aligned}$$

where $\Gamma(\cdot,\cdot,\cdot)$ is the incomplete generalized gamma function defined by

$$\Gamma(a, z_0, z_1) = \int_{z_0}^{z_1} t^{a-1} e^{-t} \mathrm{d}t \,.$$

Now, an estimating of the shape parameter m is obtained by equating the theoretical population mean with sample mean of W-transformed W's. Nagatsuka and Kamakura (2003) provided a table for obtaining estimates and concluded based on simulation studies that the robust estimate of m is possible without using any existing threshold parameter.

27.3 Regression Models

Survival analysis is now a standard statistical method for lifetime data. Fundamental and classical parametric distributions are also very important, but regression methods are very powerful to analyze the effects of some covariates on life lengths. Cox (1972) introduced a model for the hazard function $\lambda(t;x)$ with survival time T for an individual with possibly time-dependent covariate x, i.e.,

$$\lambda(t;x) = \lambda_0(t)\exp(\beta^\top x)\,, \tag{27.17}$$

where $\lambda_0(t)$ is an arbitrary and unspecified base-line hazard function and $x^\top = (x_1,\ldots,x_p)$ and $\beta^\top = (\beta_1,\ldots,\beta_p)$. Cox generalized (27.17) this to a discrete logistic model expressing y as

$$\frac{\lambda(t;x)}{1-\lambda(t;x)} = \frac{\lambda_0(t)}{1-\lambda_0(t)}\exp(\beta^\top x)\,. \tag{27.18}$$

Kamakura and Yanagimoto (1983) compared the estimators of regression parameters in the proportional hazards model (27.17) or (27.18) when we take the following methods; the Breslow–Peto (Breslow 1974; Peto 1972) method, the partial likelihood (Cox 1972, 1975) method and the generalized maximum likelihood method (Kalbfleish and Prentice 1980; Miller 1981).

27.3.1 The Score Test

In many applications it is necessary to test the significance of the estimated value, using for example the score test or the likelihood ratio test based on asymptotic results of large sample theory. First we express the three likelihood factors defined at each failure time as L_{BP}, L_{PL}, L_{GML} corresponding to the Breslow–Peto, the partial likelihood and the generalized maximum likelihood methods, respectively;

$$L_{BP}(\beta) = \frac{\prod_{i=1}^{r}\exp(\beta^\top x_i)}{\left\{\sum_{i=1}^{n}\exp(\beta^\top x_i)\right\}^r}, \tag{27.19}$$

$$L_{PL}(\beta) = \frac{\prod_{i=1}^{r}\exp(\beta^\top x_i)}{\sum_{\Psi}\prod_{i=1}^{r}\exp\left(\beta^\top x_{\psi_i}\right)}, \tag{27.20}$$

$$L_{GML}(\beta) = \frac{\prod_{i=1}^{r}\lambda\exp(\beta^\top x_i)}{\prod_{i=1}^{n}\left\{1+\lambda\exp(\beta^\top x_i)\right\}}\,, \tag{27.21}$$

where $x_1,\ldots,x_n$ denote covariate vectors for n individuals at risk at a failure time and $x_1,\ldots,x_r$ correspond to the failures, and Ψ denotes the set of all subsets

$\{\psi_1, \ldots, \psi_r\}$ of size r from $\{1, \ldots, n\}$. The overall likelihood obtained by each method is the product of these cases of many failure times. It can be shown that the first derivatives of the three log likelihoods with respect β have the same values, i.e.,

$$\sum_{i=1}^{r} x_{ji} - \frac{r}{n} \sum_{i=1}^{n} x_{ji} \quad (j = 1, \ldots, p)$$

at $\beta = 0$.

The Hessian matrices of the log likelihoods evaluated at $\beta = 0$ are respectively,

$$-\left(\frac{r}{n}\right) S,$$
$$-\left\{\frac{r(n-r)}{n(n-1)}\right\} S,$$
$$-\left\{\frac{r(n-r)}{n^2}\right\} S,$$

where S is a matrix whose elements $s_j k$ are defined by

$$s_{jk} = \sum_{i=1}^{n} (x_{ji} - \bar{x}_{j.})(x_{ki} - \bar{x}_{k.}) .$$

The first two results were derived by Farewell and Prentice (1980). Maximizing out λ from L_{GML} gives the last one, which is obtained in an unpublished manuscript. Since

$$\frac{r}{n} \geq \frac{r(n-r)}{n(n-1)} > \frac{r(n-r)}{n^2},$$

we conclude that the Breslow–Peto approach is the most conservative one.

27.3.2 Evaluation of Estimators in the Cox Model

Farewell and Prentice (1980) pointed out in their simulation study that when the discrete logistic model is true the Breslow–Peto method causes downward bias compared to the partial likelihood method. This was proven in Kamakura and Yanagimoto (1983) for any sample when β is scalar-valued, i.e.,

Theorem 2. *(Kamakura and Yanagimoto 1983)*
Let $\widehat{\beta}_{BP}$ be the maximum likelihood estimator of $L_{BP}(\beta)$ and $\widehat{\beta}_{PL}$ be that of $L_{BP}(\beta)$. Suppose that all x_i's are not identical. Then both $\widehat{\beta}_{BP}$ and $\widehat{\beta}_{PL}$ are unique, if they exist, and $sgn(\widehat{\beta}_{BP}) = sgn(\widehat{\beta}_{PL})$ and

$$\left|\widehat{\beta}_{BP}\right| \leq \left|\widehat{\beta}_{PL}\right| . \tag{27.22}$$

The equality in (27.22) holds when $\widehat{\beta}_{PL}$ is equal to zero or the number of ties r is equal to one.

Corollary 1. *(Kamakura and Yanagimoto 1983)*
The likelihood ratio test for $\beta = 0$ against $\beta \neq 0$ is also conservative if we use the Breslow–Peto method. The statement is also valid in the multivariate case.

This theorem and corollary confirm the conservatism of the Breslow–Peto approximation in relation to Cox's discrete model (Oaks 2001).

27.3.3 Approximation of Partial Likelihood

Yanagimoto and Kamakura (1984) proposed an approximation method using full likelihood for the case of Cox's discrete model. Analytically the same problems appear in various fields of statistics. Prentice and Breslow (1978) and Farewell (1979) remarked that the inference procedure using the logistic model contains the same problems in case-control studies where data are summarized in multiple 2×2 or $k \times 2$ tables. The proportional hazards model provides a type of logistic model for the contingency table with ordered categories (Pregibon 1982). As an extension of the proportional hazards model, the proportional intensity model in the point process is employed to describe an asthma attack in relation to environmental factors (Korn and Whittemoore 1979; Yanagimoto and Kamakura 1984). For convenience, although in some cases partial likelihood becomes conditional likelihood, we will use the term of partial likelihood.

It is worthwhile to explore the behavior of the maximum full likelihood estimator even when the maximum partial likelihood estimator is applicable. Both estimators obviously behave similarly in a rough sense, yet they are different in details. Identifying differences between the two estimators should be helpful in choosing one of the two.

We use the notation described in the previous section for expressing the two likelihoods. Differentiating $\log L_{PL}$ gives

$$LP(\beta) = \sum_{i=1}^{r} x_i - \frac{\sum_{\Psi} \sum_{\psi} x_j \exp\left(\beta^{\top} \sum_{\psi} x_j\right)}{\sum_{\Psi} \exp\left(\beta^{\top} \sum_{\psi} x_j\right)} = 0\,.$$

Differentiating $\log L_{GML}$ with respect to β and λ allows obtaining the maximum full likelihood estimator, i.e.,

$$\sum_{i=1}^{r} x_i - \sum_{i=1}^{n} \lambda x_i \frac{\exp(\beta^{\top} x_i)}{1 + \lambda \exp(\beta^{\top} x_i)} = 0$$

and

$$\frac{r}{\lambda} - \sum_{i=1}^{n} \frac{\exp(\beta^\top x_i)}{1 + \lambda \exp(\beta^\top x_i)}.$$

From the latter equation $\lambda(\beta)$ is uniquely determined for any fixed β. Using $\lambda(\beta)$, we define

$$LF(\beta) = \sum_{i=1}^{r} x_i - \sum_{i=1}^{n} \lambda(\beta) x_i \frac{\exp(\beta^\top x_i)}{1 + \lambda \exp(\beta^\top x_i)}.$$

The maximum full likelihood estimator, $\widehat{\beta}_{GML}$, is a root of the equation $LF(\beta) = 0$. We denote $\lambda(\beta)$ by λ for simplicity.

Note that the entire likelihoods are the products over all distinct failure times T. Thus the likelihood equations in a strict sense are $\sum LP_t(\beta) = 0$ and $\sum LF_t(\beta) = 0$, where the summations extend over t in T. As far as we are concerned, the results in a single failure time can be straightforwardly extended to those with multiple failure times. Let us now focus on likelihood equations of a single failure time and suppress the suffix t.

Proposition 1. *(Yanagimoto and Kamakura 1984)*
Let $K(\beta)$ be either of $LF(\beta)$ or $LP(\beta)$. Denote $\sum_{i=1}^{n} x_i/n$ by $\bar{x}$, and $x_{(1)}+\cdots+x_{(r)}$ and $x_{(n-r+1)} + \cdots + x_{(n)}$ by $L(x;r)$ and $U(x;r)$ respectively, where $x_{(1)}, \ldots, x_{(n)}$ are ordered covariates in ascending order. $K(\beta)$ accordingly has the following four properties:

(1) $K(0) = x_1 + \cdots + x_r - r\bar{x}$.
(2) $K'(\beta)$ *is negative for any* β, *that is,* $K(\beta)$ *is strictly decreasing.*
(3) $\lim_{\beta \to -\infty} K(\beta) = U(x;r)$.
(4) $\lim_{\beta \to \infty} K(\beta) = L(x;r)$.

Extension to the case of vector parameter β is straightforward. From Proposition 1 it follows that if either of the two estimators exists, then the other also exists and they are uniquely determined. Furthermore, both the estimators have a common sign.

Theorem 3. *(Yanagimoto and Kamakura 1984)*
Suppose that $\sum(x_i - \bar{x})^2 \neq 0$. The functions $LP(\beta)$ and $LF(\beta)$ then have a unique intersection at $\beta = 0$. It also holds that $LP(\beta) < LF(\beta)$ for $\beta > 0$. The reverse inequality is valid for $\beta < 0$.

The above theorem proves that $\widehat{\beta}_{GML} > \widehat{\beta}_{PL}$ for the case of $LP(0) = LF(0) > 0$.

To quantitatively compare the behaviors of $LF(\beta)$ and $LP(\beta)$, their their power expansions are presented near the origin. Since both functions behave similarly, it is expected that the quantitative difference near the origin is critical over a wide range of β. Behavior near the origin is of practical importance for studying the estimator and test procedure.

Proposition 2. *(Yanagimoto and Kamakura 1984)*
The power expansions of $LF(\beta)$ and $LP(\beta)$ near the origin up to the third order are as follows: for $n \geq 4$,

(1)

$$LF(\beta) \approx \sum_{i=1}^{r} x_i - \left[r\bar{x} + \frac{r(n-r)}{n^2} s_2 \beta + \frac{1}{2} \frac{r(n-r)(n-2r)}{n^3} s_3 \beta^2 + \frac{1}{6} \frac{r(n-r)}{n^5} \left\{ n(n^2 - 6rn + 6r^2) s_4 - 3(n-2r)^2 s_2^2 \right\} \beta^3 \right],$$

(2) (Cox 1970)

$$LP(\beta) \approx \sum_{i=1}^{r} x_i - \left[r\bar{x} + \frac{r(n-r)}{n(n-1)} s_2 \beta + \frac{1}{2} \frac{r(n-r)(n-2r)}{n(n-1)(n-2)} s_3 \beta^2 + \frac{1}{6} \frac{r(n-r)}{n^2(n-1)(n-2)(n-3)} \left\{ n(n^2 - 6rn + 6r^2 + n) s_4 + 3(r-1)n(n-r-1) s_2^2 \right\} \beta^3 \right],$$

where $s_k = \sum (x_i - \bar{x})^k$, $k = 2, 3$ and 4.

The function $LF(\beta)$ has a steeper slope near the origin than $LP(\beta)$. The relative ratio is $n/(n-1)$, which indicates that $LF(n\beta/(n-1))$ is close to $LP(\beta)$ near the origin. The power expansion of $LA(\beta) = LF(n\beta/(n-1))$ is expressed by

$$LA(\beta) \approx \sum_{i=1}^{r} x_i - \left\{ r\bar{x} + \frac{r(n-r)}{n(n-1)} s_2 \beta + \left(\frac{n}{n-1} \right)^2 c_3 \beta^2 + \left(\frac{n}{n-1} \right)^3 c_4 \beta^3 \right\}, \tag{27.23}$$

where c_3 and c_4 are coefficients of order 2 and 3 of $LF(\beta)$. Although $LA(\beta)$ is defined to adjust the coefficient of $LF(\beta)$ of order 1 to that of $LP(\beta)$, the coefficient of order 2 of $LA(\beta)$ becomes closer to that of $LP(\beta)$ than that of $LF(\beta)$. The following approximations are finally obtained.

$$LP(\beta) \approx LA(\beta), \tag{27.24}$$

$$\widehat{\beta}_{PL} \approx \frac{(n-1)\widehat{\beta}_{GML}}{n}. \tag{27.25}$$

The proposed approximated estimator and test statistic are quite helpful in cases of multiple 2×2 table when the value of both n and r are large (Yanagimoto and Kamakura 1984).

27.4 Multiple Failures and Counting Processes

The standard methods of survival analysis can be generalized to include multiple failures simply defined as a series of well-defined event occurrences. For example, in software reliability, engineers are often interested in detecting software bugs. Inference from a single counting process has been studied in detail (Cox and Lewis 1966; Musa et al. 1987), with multiple independent processes being considered as a means to estimate a common cumulative mean function from a nonparametric or semi-parametric viewpoint (Lawless and Nadeau 1993; Nelson 1992). Kamakura (1996) discussed problems associated with parametric conditional inference in models with a common trend parameter or possibly different base-line intensity parameters.

27.4.1 Intensity Function

For multiple failures, intensity functions correspond to hazard functions in that the intensity function is defined as discussed next.

In time interval $[t_0, t]$ we define the number of occurrences of events or failures as $N(t)$. The Poisson counting process $\{N(t) : t \geq t_0\}$ is given such that it satisfies the following three conditions for $t \geq t_0$.

1. $\Pr\{N(t_0) = 0\} = 1$
2. The increment $N_{s,t} = N(t) - N(s)$ $(t_0 \geq s < t)$ has a Poisson distribution with the mean parameter $\Lambda_t - \Lambda_s$, for some positive and increasing function in t.
3. $\{N_t : t \geq t_0\}$ is a process of independent increments. That is, for any $(t_0 <)$ $t_1 < t_2 < \cdots < t_n$, n increments, $N(t_1) - N(t_0), \ldots, N(t_n) - N(t_{n-1})$ are mutually independent.

For this counting process $\{N(t) : t \geq t_0\}$ we can define the intensity function as

$$\lambda(t) = \lim_{\Delta \to 0} \frac{1}{\Delta} \Pr\{N(t + \Delta t) - N(t) = 1 | H(t)\}, \tag{27.26}$$

where $H(t)$is the history of the process up to t:

$$H(t) = \{N(u) : \ t_0 \leq u \leq t\}.$$

Note that

$$\Lambda(t) = \int_{t_0}^{t} \lambda(t) \mathrm{d}t .$$

Expectation of $E[N_{s,t}]$ becomes

$$E[N_{s,t}] = \sum_{n=0}^{\infty} n \Pr\{N_{n,s} = n\} = \Lambda_t - \Lambda_s \,, \tag{27.27}$$

and

$$\lambda(t) = \frac{\mathrm{d}}{\mathrm{d}t}\Lambda_t = \frac{\mathrm{d}}{\mathrm{d}t}E[N(t)] \,. \tag{27.28}$$

The nonparametric estimate of the intensity function is easy to determine and is quite useful for observing the trend of a series of events. If a data set of failure times $\{t_1, t_2, \ldots, t_n\}$ is available, assuming constant intensity in $(t_{k_1}, t_k]$, then

$$\lambda(t) = \lambda_k \quad (t_{k-1} < t \leq t_k) \,,$$

and the nonparametric ML estimates becomes

$$\lambda_k = \frac{1}{t_k - t_{k-1}} \quad (k = 1, \ldots, n) \,, \tag{27.29}$$

where $t_0 = 0$.

27.4.2 Multiple Counting Processes

We assume several independent counting processes $\{N_k(t_k)$, i.e., $0 < t_k \leq \tau_k, k = 1, \ldots, K\}$. The cumulative mean function for $N_k(t)$ is expressed by

$$M_k(t) = E\left\{N_k(t)\right\} \,. \tag{27.30}$$

Nelson (1992) described a method for estimating the cumulative mean function of an identically distributed process without assuming any Poisson process structure, while Lawless and Nadeau (1993) developed robust variance estimates based on the Poisson process. All these methods are basically concerned with nonparametric estimation. Here, parametric models for effectively acquiring information on the trend of an event occurrence are dealt with. Kamakura (1996) considered generalized versions of two primal parametric models to multiple independent counting processes under the framework of a nonhomogeneous Poisson process.

Cox and Lewis (Cox and Lewis 1966) considered a log-linear model for trend testing a singe counting process, i.e.,

$$\lambda(t) = \exp(\alpha + \beta t) \,, \tag{27.31}$$

where $\lambda(t)$ is the intensity function corresponding to the derivative of the mean function in the continuous case. Note that for a single case the subscript k is omitted.

They assumed the above nonhomogeneous Poisson process and gave a simple test statistic for $H_0 : \beta = 0$ against $H_A : \beta \neq 0$, i.e.,

$$U = \frac{\sum_{i=1}^{n} t_i - \frac{1}{2}\tau_0}{\tau_0 \sqrt{\frac{n}{12}}} . \tag{27.32}$$

The distribution of this statistic steeply converges to the standard normal distribution when $n \to \infty$. This statistic is sometimes called the U statistic and is frequently applied to trend testing in reliability engineering.

Kamakura (1996) generalized this log-linear model to the multiple case, with the log-linear model for k-th individual being

$$\lambda_k(t) = \exp(\alpha_k + \beta t) . \tag{27.33}$$

In this modeling we assume the common trend parameter β and are mainly interested in estimating and testing this parameter. The full likelihood for the model becomes

$$\begin{aligned} L(\beta, \alpha_1, \alpha_2, \ldots, \alpha_K) &= \prod_{k=1}^{K} \left[\left\{ \prod_{i=1}^{n_k} \lambda_k(t_{ki}) \right\} \exp\left\{ -\int_0^{\tau_k} \lambda_k(u)\mathrm{d}u \right\} \right] \\ &= \exp\left\{ \sum_{k=1}^{K} n_k \alpha_k + \beta \sum_{k=1}^{K} \sum_{i=1}^{n_k} t_{ki} - \frac{1}{\beta} \sum_{k=1}^{K} e^{\alpha_k} \left(e^{\beta \tau_k} - 1 \right) \right\} . \end{aligned} \tag{27.34}$$

If K is large, it is difficult to compute all parameter estimates based on such full likelihood.

Given $N_k(\tau_k) = n_k, k = 1, 2, \ldots, K$, conditional likelihood is considered as

$$CL(\beta | N_k(\tau_k) = n_k, i = 1, \ldots, K) = \frac{\prod_{k=1}^{K} (n_k!) \beta^{\sum n_k} e^{\beta \sum\sum t_{ki}}}{\prod_{k=1}^{K} \left(e^{\beta \tau_k} - 1 \right)^{n_k}} . \tag{27.35}$$

Note that the nuisance parameter α_k's do not appear. Fisher information is calculated as

$$\begin{aligned} I(\beta) &= E\left[-\frac{\partial^2 \log CL}{\partial \beta^2} \right] \\ &= \begin{cases} \sum_{k=1}^{K} n_k \left\{ \frac{1}{\beta^2} - \frac{\tau_k^2 e^{-\beta\tau_k}}{\left(1 - e^{-\beta\tau_k}\right)^2} \right\} & (\beta \neq 0) \\ \frac{1}{12} \sum_{k=1}^{K} n_k \tau_k^2 & (\beta = 0) \end{cases} . \end{aligned} \tag{27.36}$$

The test statistic obtained from the above calculations becomes

$$U_k = \frac{\log CL|_{\beta=0}}{\sqrt{I(0)}} = \frac{\sum_{k=1}^{K}\sum_{i=1}^{n_k} t_{ki} - \frac{1}{2}\sum_{k=1}^{K} n_k \tau_k}{\sqrt{\frac{1}{12}\sum_{k=1}^{K} n_k \tau_k^2}}. \tag{27.37}$$

To obtain the conditional estimate, numerical calculations are required such as Newton–Raphson method. However, the log conditional likelihood and its derivatives are not computable at the origin of the parameter β. In such a case, Taylor series expansions of the log conditional likelihood are used around the origin (Kamakura 1996).

27.4.3 Power Law Model

Crow (1982) considered the power law model, sometimes called the Weibull process model. This model was generalized to the multiple case using the following intensity for the k-th individual (Kamakura 1996):

$$\lambda_k(t) = \boldsymbol{\theta}_k m t^{m-1}. \tag{27.38}$$

In this case it is easy to calculate the MLE. Direct calculation of the likelihood gives rise to the MLE $\widehat{m}$ and $\widehat{\boldsymbol{\theta}_k}$ i.e.,

$$\widehat{m} = \frac{\sum_{k=1}^{K} n_k}{\sum_{k=1}^{K}\sum_{i=1}^{n_k} \log\left(\frac{\tau_k}{t_{ki}}\right)}, \tag{27.39}$$

$$\widehat{\boldsymbol{\theta}_k} = \frac{n_k}{\tau_k^{\widehat{m}}}. \tag{27.40}$$

Putting

$$Z = \frac{2m\sum_{k=1}^{K} n_k}{\widehat{m}}, \tag{27.41}$$

the distribution of Z becomes a chi-square with $2\sum_{k=1}^{K} n_k$ degrees of freedom. Based on this result we can make an inference of the common parameter m.

27.4.4 Models Suitable for Conditional Estimation

Estimation based on conditional likelihood allows effectively eliminating the nuisance parameter and obtaining information on the structure parameter. Let us now

consider the class of nonhomogeneous Poisson process models which are specified by the intensity parameterized by two parameters. The first parameter α is concerned with the base line occurrences for the individual, while the second parameter β is concerned with the trend of intensity. For simplicity, the property of the intensity for $K = 1$ is examined. Using conditional likelihood is convenient because the nuisance parameter α need not be known. This is of great importance in multiple intensity modeling, i.e.,

Theorem 4. *(Kamakura 1996)*
Conditional likelihood does not include the nuisance parameter α iff the intensity is factorized as two factors, a function of α and a function of β and the time t, in the class of nonhomogeneous Poisson process models. That is, the intensity is expressed as

$$\lambda(t;\alpha,\beta) = h(\alpha)g(\beta;t)\,, \quad a.s. \tag{27.42}$$

Several intensity models for software reliability are described in Musa et al. (1987): the log-linear model, geometric model, inverse linear model, inverse polynomial model, and power law model, all of which are included in this class satisfying the condition of the theorem.

Acknowledgements This work has been partially supported financially by Chuo University as one of the 2003 Research Projects for Promotion of Advanced Research at Graduate School.

References

Breslow, N.E.: Covariance analysis of censored survival data. Biometrics **30**, 89–99 (1974)
Castillo, E., Hadi, A.S.: Modeling lifetime data with application to fatigue models. J. Am. Stat. Assoc. **90**, 1041–1054 (1995)
Cohen, A.C.: Maximum likelihood estimation in the Weibull distribution based on complete and censored samples. Technometrics **5**, 579–588 (1965)
Cohen, A.C., Whitten, B.J.: Parameter Estimation in Reliability and Life Span Models. Marcel Dekker, New York (1988)
Cox, D.R.: The Analysis of Binary Data. Methuen, London (1970)
Cox, D.R.: Regression models and life tables (with discussion). J. Roy. Stat. Soc. B **34**, 187–220 (1972)
Cox, D.R.: Partial likelihood. Biometrika **62**, 269–276 (1975)
Cox, D.R., Lewis, P.A.W.: The Statistical Analysis of Series of Events. Methuen, London (1966)
Crow, L.H.: Confidence interval procedures for the Weibull process with applications to reliability growth. Technometrics **24**, 67–72 (1982)
Dubey, S.D.: Normal and Weibull distributions. Nav. Res. Logist. Q. **14**, 69–79 (1967)
Farewell, V.T.: Some results on the estimation of logistic models based on retrospective data. Biometrika **66**, 27–32 (1979)
Farewell, V.T., Prentice, R.L.: The approximation of partial likelihood with emphasis on case-control studies. Biometrika **67**, 273–278 (1980)
Greenwood, M.: A report on the natural duration of cancer: Appendix I The Errors of Sampling of the Survivorshi Tables: Reports on Public Health and Medical Subjects. His Majesty's Stationery Office, London (1926)

Johnson, N.L., Kotz, S., Balakrishna, N.: Continuous Univariate Distributions. John Wiley, New York (1994)
Kalbfleish, J.D., Prentice, R.L.: The Statistical Analysis of Failure Time Data. John Wiley, New York (1980)
Kamakura, T.: Trend analysis of multiple counting processes. In: Jewell, N.P., Kimber, A.C., Lee, M.-L.T., Whitmore, G.A. (eds.) Lifetime Data: Models in Reliability and Survival Analysis, pp. 149–156. Kluwer Academic Publishers, Dordrecht (1996)
Kamakura, T., Yanagimoto, T.: Evaluation of the regression parameter estimators in the proportional hazard model. Biometrika **70**, 530–533 (1983)
Kaplan, E.L., Meier, P.: Nonparametric estimation from incomplete observations. J. Am. Stat. Assoc. **53**, 457–481 (1958)
Korn, E.L., Whittemoore, A.: Methods for analysing panel studies of acute health effects of air pollution. Biometrics **35**, 795–802 (1979)
Lawless, J.F., Nadeau, J.C.: Some simple robust methods for the analysis of recurrent events. University of Waterloo IIQP Research Report (1993)
Menon, M.: Estimation of the shape and scale parameters of the Weibull distributions. Technometrics **5**, 175–182 (1963)
Miller, R.G.: Survival Analysis. John Wiley, New York (1981)
Musa, J.D., Iannino, A., Okumoto, K.: Software Reliablility. McGraw-Hill, New York (1987)
Nagatsuka, H., Kamakura, T.: A new method of inference for Weibull shape parameter (in Japanese). J. Reliability Eng. Assoc. Jpn. **25**, 583–593 (2003)
Nagatsuka, H., Kamakura, T.: Parameter estimation of the shape parameter of the Castillo–Hadi model. Comm. Stat. Theor. Meth. **33**, 15–27 (2004)
Nelson, W.B.: Confidence limits for recurrence data – applied to cost or number of product repairs and of disease episodes. Technometrics **22**, 1023–1031 (1992)
Oaks, D.: Biometrika centenary: Survival analysis. Biometrika **88**, 99–142 (2001)
Peto, R.: Discussion of paper by D. R. Cox. J. Roy. Stat. Soc. B **34**, 205–207 (1972)
Pregibon, D.: Resistant fits for some commonly used logistic models with medical applications. Biometrics **38**, 485–498 (1982)
Prentice, R.L., Breslow, N.E.: Restrospective studies and failure time models. Biometrika **65**, 153–158 (1978)
Yanagimoto, T., Kamakura, T.: The maximum full and partial likelihood estimators in the proportional hazard model. Ann. Inst. Stat. Math. **36**, 363–373 (1984)

Chapter 28
Data and Knowledge Mining

Adalbert Wilhelm

28.1 Data Dredging and Knowledge Discovery

Data mining was one of the buzz-words at the verge of the third millennium. It was already a multi-million dollar industry in the late 1990s and experts expected a continuing growth for the first decade of the 21st century. Although this expectation has not quite materialized in recent years, data mining still is an important field of scientific research with great potential for commercial usage. The ubiquitous computer makes it possible to collect huge data bases that contain potentially valuable information. Sophisticated analysis techniques are needed to explore these large, often heterogeneous, data sets and to extract the small pieces of information that are valuable to the data owner.

The importance of exploring and analyzing real data sets is not new to statistics. It has been reinforced in the late 1960s by John W. Tukey who realized that putting too much emphasis on the mathematical theories of statistics did not help in solving the real world problems. It was his mantra that statistical work is detective work (Tukey 1969) and that one should let the data speak for itself. The branch of exploratory data analysis emerged, but was dismissed by mathematical statisticians for a long period of time. Many of them proclaimed that proper statistical analysis must be based on hypothesis and distributional assumptions. Their argument was that looking at data before formulating a scientific hypothesis will bias the hypothesis towards what the data might show. The term data mining typically was used in a derogatory connotation. The argument culminated in the reproach of improper scientific use, the reproach of torturing the data until it confesses everything.

A. Wilhelm (✉)
Commerzbank Chair of Information Management, School of Humanities and Social Sciences, Jacobs University Bremen gGmbH, Bremen, Germany
e-mail: a.wilhelm@jacobs-university.de

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_28,

The advent of information technology that allowed to easily collect and store data of previously unimaginable quantities brought a rapid change to the scene and superseded academic disputes. Once the computer power and technology was there, that made it easy to collect information of all customers in a super market, or for all customers of a telephone company, the need arose to make use of these large information sources.

In the last few years, a strong impetus on the development of data mining methods came from homeland security issues (Fienberg 2008). Three aspects have received particular attention: the problem of person detection in images and video sequences; the analysis and visualization of social networks; and the integration and combination of various data sources as well as evidences (cf Chen et al. 2008). A further active field is web mining which divides into web content mining, web usage mining (cf. Zhang et al. 2009) and web structure mining.

Web usage mining aims at detecting usage patterns of web sites in order to get a better understanding of users' interests and needs. Both the detection of network paths as well as the detection of similar objects are fundamental concepts in this approach to improve customer satisfaction and to enhance e-commerce.

Web content mining is an application of text mining and aims at detecting regularities and structures in web sites. Based on the typical unstructured data contained in websites, including text, images, and videos, the main targets in web content mining are categorization of documents, clustering of web sites, finding extraction rules and patterns in the text.

Web structure mining tries to detect the linking structure underneath a web site. While web content mining deals with the structure within web documents (intra-document analysis), web structure mining is concerned with the hyperlinks between web pages (inter-document structure). This line of research is closely related to social network analysis and citation analysis. It is using graph theoretic tools and techniques to analyze the possible paths within a website and within the internet.

Data Mining is a thriving field of research and application, to which both statisticians and computer scientists have contributed new ideas and new techniques. In this contribution, we will introduce the main components, tasks, and computational methods for data mining. After an attempt to define data mining, we relate it to the larger field of knowledge discovery in databases in Sect. 28.2. Section 28.3 deals with the two flavors of learning from data: supervised and unsupervised learning. We will then discuss the different data mining tasks in Sect. 28.4, before we present the computational methods to tackle them in Sect. 28.5. In the final Sect. 28.6, we present some recent trends and controversies.

28.2 Knowledge Discovery in Databases

There are almost as many differing definitions of the term "Data Mining" as there are authors who have written about it. Since Data Mining sits at the interface of a variety of fields, e.g. computer science, statistics, artificial intelligence, business

information systems, and machine learning, its definition changes with the field's perspective. Computer scientists, typically, refer to Data Mining as a clearly defined part of the Knowledge Discovery in Databases (KDD) process, while many statisticians use Data Mining as a synonym for the whole KDD process.

To get a flavor of both the variation as well as the common core of data and knowledge mining, we cite some of the definitions used in the literature.

> *KDD is the non-trivial process of identifying valid, novel, potentially useful, and ultimately understandable patterns in data* (Fayyad et al. 1996)

> *Knowledge discovery is a knowledge-intensive task consisting of complex interactions, protracted over time, between a human and a (large) database, possibly supported by a heterogenous suite of tools.* (Brachman and Anand 1996)

> *[Data Mining is] a step in the KDD process consisting of particular data mining algorithms that, under some acceptable computational efficiency limitations, produce a particular enumeration of patterns* (Fayyad et al. 1996)

> *[Data Mining is] a folklore term which indicates application, under human control, of low-level data mining methods. Large scale automated search and interpretation of discovered regularities belong to KDD, but are typically not considered part of data mining* (Kloesgen and Zytkow 1996)

> *[Data Mining is] used to discover patterns and relationships in data, with an emphasis on large observational data bases. It sits at the common frontiers of several fields including Data Base Management, Artificial Intelligence, Machine Learning, Pattern Recognition, and Data Visualization.* (Friedman 1998)

> *[Data Mining is] the process of secondary analysis of large databases aimed at finding unsuspected relationships which are of interest or value to the database owners* (Hand 1998)

> *Data Mining is the analysis of (often large) observational data sets to find unsuspected relationships and to summarize the data in novel ways that are both understandable and useful to the data owner* (Hand et al. 2001)

From these definitions the essence is that we are talking about exploratory analysis of large data sets. Two further aspects are the use of computer-based methods and the notion of secondary and observational data. The latter means that the data do not come from experimental studies and that data was originally collected for some other purpose, either for a study with different goals or for record-keeping reasons. These four characteristics in combination distinguish the field of Data Mining from traditional statistics. The exploratory approach in Data Mining clearly defines the goal of finding patterns and generating hypothesis, which might later on be subject of designed experiments and statistical tests. Data sets can be large at least in two different aspects. The most common one is in form of a large number of observations (cases). Real world applications usually are also large in respect of the number of variables (dimensions) that are represented in the data set. Data Mining is also concerned with this side of largeness. Especially in the field of bioinformatics, many data sets comprise only a small number of cases but a large number of variables. Secondary analysis implies that the data can rarely be

regarded as a random sample from the population of interest and may have quite large selection biases. The primary focus in investigating large data sets tends not to be the standard statistical approach of inferencing from a small sample to a large universe, but more likely partitioning the large sample into homogeneous subsets.

The ultimate goal of Data Mining methods is not to find patterns and relationships as such, but the focus is on extracting knowledge, on making the patterns understandable and usable for decision purposes. Thus, Data Mining is the component in the KDD process that is mainly concerned with extracting patterns, while Knowledge Mining involves evaluating and interpreting these patterns. This requires at least that patterns found with Data Mining techniques can be described in a way that is meaningful to the data base owner. In many instances, this description is not enough, instead a sophisticated model of the data has to be constructed.

Data pre-processing and data cleansing is an essential part in the Data and Knowledge Mining process. Since data mining means taking data from different sources, collected at different time points, and at different places, integration of such data as input for data mining algorithms is an easily recognized task, but not easily done. Moreover, there will be missing values, changing scales of measurement, as well as outlying and erroneous observations. To assess the data quality is a first and important step in any scientific investigation. Simple tables and statistical graphics give a quick and concise overview on the data, to spot data errors and inconsistencies as well as to confirm already known features. Besides the detection of uni- or bivariate outliers graphics and simple statistics help in assessing the quality of the data in general and to summarize the general behavior. It is worth noting that many organizations still report that as much as 80% of their effort for Data and Knowledge Mining goes into supporting the data cleansing and transformation process.

28.3 Supervised and Unsupervised Learning

Data and Knowledge Mining is learning from data. In this context, data are allowed to speak for themselves and no prior assumptions are made. This learning from data comes in two flavors: supervised learning and unsupervised learning. In supervised learning (often also called directed data mining) the variables under investigation can be split into two groups: explanatory variables and one (or more) dependent variables. The target of the analysis is to specify a relationship between the explanatory variables and the dependent variable as it is done in regression analysis. To apply directed data mining techniques the values of the dependent variable must be known for a sufficiently large part of the data set.

Unsupervised learning is closer to the exploratory spirit of Data Mining as stressed in the definitions given above. In unsupervised learning situations all variables are treated in the same way, there is no distinction between explanatory and dependent variables. However, in contrast to the name undirected data mining there is still some target to achieve. This target might be as general as data reduction or more specific like clustering. The dividing line between supervised learning and

unsupervised learning is the same that distinguishes discriminant analysis from cluster analysis. Supervised learning requires that the target variable is well defined and that a sufficient number of its values are given. For unsupervised learning typically either the target variable is unknown or has only been recorded for too small a number of cases.

The large amount of data that is usually present in Data Mining tasks allows to split the data file in three groups: training cases, validation cases and test cases. Training cases are used to build a model and estimate the necessary parameters. The validation data helps to see whether the model obtained with one chosen sample may be generalizable to other data. In particular, it helps avoiding the phenomenon of overfitting. Iterative methods incline to result in models that try to do too well. The data at hand is perfectly described, but generalization to other data yields unsatisfactory outcomes. Not only different estimates might yield different models, usually different statistical methods or techniques are available for a certain statistical task and the choice of a method is open to the user. Test data can be used to assess the various methods and to pick the one that does the best job on the long run.

Although we are dealing with large data sets and typically have abundant cases, partially missing values and other data peculiarities can make data a scarce resource and it might not be easily achievable to split the data into as many subsets as there are necessary. Resampling and cross-validation techniques are often used in combination to data and computer intensive methods in Data Mining.

28.4 Data Mining Tasks

The cycle of data and knowledge mining comprises various analysis steps, each step focusing on a different aspect or task. Hand et al. (2001) propose the following categorization of data mining tasks.

28.4.1 Description and Summarization

At the beginning of each data analysis is the wish and the need to get an overview on the data, to see general trends as well as extreme values rather quickly. It is important to familiarize with the data, to get an idea what the data might be able to tell you, where limitations will be, and which further analyses steps might be suitable. Typically, getting the overview will at the same time point the analyst towards particular features, data quality problems, and additional required background information. Summary tables, simple univariate descriptive statistics, and simple graphics are extremely valuable tools to achieve this task.

Unwin et al. (2002) report from a study of 50,000 car insurance policies during which the following difficulties emerged amongst others (see Fig. 28.1).

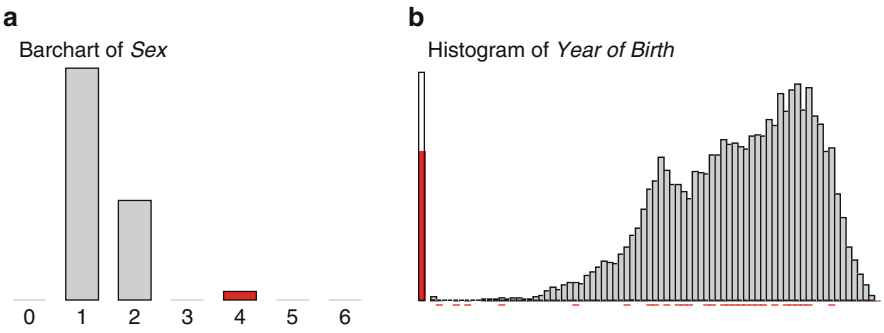


Fig. 28.1 Linked highlighting reveals structure in the data and explains unusual results of one variable quite reasonably. Barchart of Sex of car insurance policy holders on the left, Histogram of year of birth of policy holders on the right. Highlighted are cases with *Sex=4* (*firm*). The red lines under some of the bins in the histogram indicate small counts of highlighted cases that can't be displayed proportionally

(a) Barcharts of the categorical variables revealed that several had too many categories. Sex had seven, of which four were so rare as to presumably be unknowns or errors of some kind. The third large category turned out to be very reasonable: if a car was insured by a firm, the variable sex was coded as "firm". This had not been explained in advance and was obviously useful for a better grasp of the data.
(b) A histogram of date of birth showed missing values, a fairly large number (though small percentage) of underage insured persons, and a largish number born in 1900, who had perhaps been originally coded as "0" or "00" for unknown. Any analytic method using such a variable could have given misleading results.
(c) Linking the barchart of gender from (a) and the histogram of age from (b) showed quite plausibly that many firms had date of birth coded as missing, but not all. This led to further informative discussions with the data set owners.

Checking data quality is by no means a negative part of the process. It leads to deeper understanding of the data and to more discussions with the data set owners. Discussions lead to more information about the data and the goals of the study.

Speed of the data processing is an important issue at this step. For simple tasks – and data summary and description are typically considered to be simple tasks, although it is generally not true – users are not willing to spend much time. A frequency table or a scatterplot must be visible in the fraction of a second, even when it comprises a million observations. Only some computer programs are able to achieve this. Another point is a fast scan through all the variables: if a program requires an explicit and lengthy specification of the graph or table to be created, a user typically will end this tedious endeavor after a few instances. Generic functions with context-sensitive and variable-type-dependent responses provide a viable solution to this task. On the level of standard statistical data sets this is provided by software like XploRe, S-Plus and R with their generic functions

summary and *plot*. Generic functions of this kind can be enhanced by a flexible and interactive user environment which allows to navigate through the mass of data, to extract the variables that show interesting information on the first glance and that call for further investigation. Currently, no system comes close to meet these demands, future systems hopefully will do.

28.4.2 Descriptive Modeling

General descriptions and summaries are an important starting point but more exploration of the data is usually desired. While the tasks in the previous section have been guided by the goal of summary and data reduction, descriptive modeling tries to find models for the data. In contrast to the subsequent section, the aim of these models is to describe, not to predict models. As a consequence, descriptive models are used in the setting of unsupervised learning. Typical methods of descriptive modeling are density estimation, smoothing, data segmentation, and clustering. There are by now some classics in the literature on density estimation (Scott 1992) and smoothing (Härdle 1991). Clustering is a well-studied and well-known technique in statistics. Many different approaches and algorithms, distance measures and clustering schemes have been proposed. With large data sets all hierarchical methods have extreme difficulties with performance. The most widely used method of choice is k-means clustering. Although k-means is not particularly tailored for a large number of observations, it is currently the only clustering scheme that has gained positive reputation in both the computer science and the statistics community. The reasoning behind cluster analysis is the assumption that the data set contains natural clusters which, when discovered, can be characterized and labeled. While for some cases it might be difficult to decide to which group they belong, we assume that the resulting groups are clear-cut and carry an intrinsic meaning. In segmentation analysis, in contrast, the user typically sets the number of groups in advance and tries to partition all cases in homogeneous subgroups.

28.4.3 Predictive Modeling

Predictive modeling falls into the category of supervised learning, hence, one variable is clearly labeled as target variable Y and will be explained as a function of the other variables X. The nature of the target variable determines the type of model: classification model, if Y is a discrete variable, or regression model, if it is a continuous one. Many models are typically built to predict the behavior of new cases and to extend the knowledge to objects that are new or not yet as widely understood. Predicting the value of the stock market, the outcome of the next governmental election, or the health status of a person are common applications.

Banks use classification schemes to group their costumers into different categories of risk.

Classification models follow one of three different approaches: the discriminative approach, the regression approach, or the class-conditional approach. The discriminative approach aims in directly mapping the explanatory variables X to one of the k possible target categories $y_1, \ldots, y_k$. The input space X is hence partitioned into different regions which have a unique class label assigned. Neural networks and support vector machines are examples for this. The regression approach (e.g. logistic regression) calculates the posterior class distribution $P(Y \mid x)$ for each case and chooses the class for which the maximum probability is reached. Decision trees (CART, C5.0, CHAID) classify for both the discriminative approach and the regression approach, because typically the posterior class probabilities at each leaf are calculated as well as the predicted class. The class-conditional approach starts with specifying the class-conditional distributions $P(X \mid y_i, \boldsymbol{\theta}_i)$ explicitly. After estimating the marginal distribution $P(Y)$, Bayes rule is used to derive the conditional distribution $P(Y \mid x)$. The name Bayesian classifiers is widely used for this approach, erroneously pointing to a Bayesian approach versus a frequentist approach. Mostly, plug-in estimates $\hat{\boldsymbol{\theta}}_i$ are derived via maximum likelihood. The class-conditional approach is particularly attractive, because they allow for general forms of the class-conditional distributions. Parametric, semi-parametric, and non-parametric methods can be used to estimate the class-conditional distribution. The class-conditional approach is the most complex modeling technique for classification. The regression approach requires fewer parameters to fit, but still more than a discriminative model. There is no general rule which approach works best, it is mainly a question of the goal of the researcher whether posterior probabilities are useful, e.g. to see how likely the "second best" class would be.

28.4.4 Discovering Patterns and Rules

The realm of the previous tasks has been much within the statistical tradition in describing functional relationships between explanatory variables and target variables. There are situations where such a functional relationship is either not appropriate or too hard to achieve in a meaningful way. Nevertheless, there might be a pattern in the sense that certain items, values or measurements occur frequently together. Association Rules are a method originating from market basket analysis to elicit patterns of common behavior.

The practical use of association rules is not restricted to finding the general trend and the norm behavior, association rules have also been used successfully for detecting unusual behavior in fraud detection. Another field of application for association rules has been image and multi media analysis, using them with a focus on the pixel level (cf. Ding et al. 2002; Ordonez and Omiecinski 1999), with a focus on image captions and textual descriptions (cf. Wilhelm et al. 2009) or from a perceptual perspective using a thesaurus (cf. Tešić 2004).

The search for patterns and regularities is also one of the key targets in web usage mining which analyzes the traces humans leave in the internet. Typically, server logs, registration forms and other user information is analyzed to deduce patterns and regularities within users' behavior. Automatic analysis of web logs in various formats has also initiated a broad discussion on privacy issues.

A particular type of patterns are relationships within networks. Initially, developed to analyze and monitor usage data in telephone networks and the world wide web, network analysis addresses now all kind of social networks. Once again, the easy availability of corresponding data in web 2.0 social networks ignited interest in the matter and led to the development of appropriate analysis tools and techniques (cf. Chau and Xu 2008)

28.4.5 Retrieving Similar Objects

The world wide web contains an enormous amount of information in electronic journal articles, electronic catalogs, and private and commercial homepages. Having found an interesting article or picture, it is a common desire to find similar objects quickly. Based on key words and indexed meta-information search engines are providing us with this desired information. They can not only work on text documents, but to a certain extent also on images. Semi-automated picture retrieval combines the ability of the human vision system with the search capacities of the computer to find similar images in a data base.

In particular, the continuously increasing quantity of image data available on the Internet necessitates efficient classification and indexing methods for easy access and usage. The prevalent approach in current web search engines is to associate images with text, thus allowing access to the image database via text queries. This approach restricts the set of searchable images to those associated with text, and can lead to errors if the associations are incorrect. To enable more successful queries based on visual information, alternative procedures relying on image processing have been proposed: using semantic data generated by image interpretation techniques (e.g. Schober, Hermes, & Herzog 2005), or via a visual vocabulary constructed from low-level image features (e.g. Sivic & Zisserman 2003).

First prototypes have been presented for an integrated procedure, relying on both textual information (keywords extracted from captions) and on descriptors of local image features (SIFT; Lowe 1999) in the construction of the visual vocabulary. The visual words are prototypes representing groups of similar image features encountered in the training data set, which are further associated to the extracted keywords based on the frequencies of co-occurrences. This approach permits the classification of novel images into text- derived categories using only image-based data, and, correspondingly, the inclusion of images with no caption information in the search space of text queries.

28.5 Data Mining Computational Methods

When talking about Data Mining methods people tend to refer to numerical methods mostly. However, there are many reasons to consider visual approaches to data mining as equally important. We start with a concise discussion of numerical data mining methods and expand a little bit more on visual data mining later.

28.5.1 Numerical Data Mining

Decision Trees

Decision trees are popular and powerful methods to classify cases as well as to predict values. Their attractiveness is mainly due to the fact that the basic principle of tree-based methods as well as their outcome are easy to understand. The basic principle is the hierarchical division of all observations into subcategories in such a way that the resulting subcategories differ from each other as much as possible while the subcategories itself are as homogenous as possible. The outcome of a decision tree is a rule that can easily be expressed in plain English and as easily in any data base query language to quickly and repeatedly apply it to new data.

Decision trees are a supervised learning technique and require that we can extract a target variable. Depending on the scale of the target variable two types of decision trees are distinguished: classification trees, if the dependent variable is categorical, and regression trees, if the dependent variable is continuous. A full analysis with decision trees comprises two steps, growing a tree and pruning a tree.

Let us briefly describe the basic principle of growing a binary tree. Given observations for n cases on the dependent variable Y and p explanatory variables $X_1, \ldots, X_p$ we first assign all cases into the root node of the tree. The typical algorithms are greedy algorithms that enumerate all possible splits to find the best split for the current node under investigation. Thus, for each explanatory variable X_i all possible splits between values are examined. During this examination the cases in the node are partitioned into two groups (for binary trees) and a diversity measure is calculated for each potential subnode. Commonly used diversity measures are deviance, entropy, misclassification rate, and the Gini coefficient. The one split that yields the largest information gain will be taken. Information gain is measured here in terms of the decrease of the diversity measure when going from the parent node to the subnode. The algorithm uses the same process recursively to build a decision tree for each subnode. A node will not be further split if it is pure or if one of the subnodes would contain too few cases (Fig.28.2).

There are mainly three families of tree growing algorithms:

- The CART family (CART, IND CART, Splus CART, etc.)
- The ML family (ID3, C4.5, C5 and other derivatives, etc.)
- The AID family (THAID, CHAID, XAID, TREEDISC, etc.)

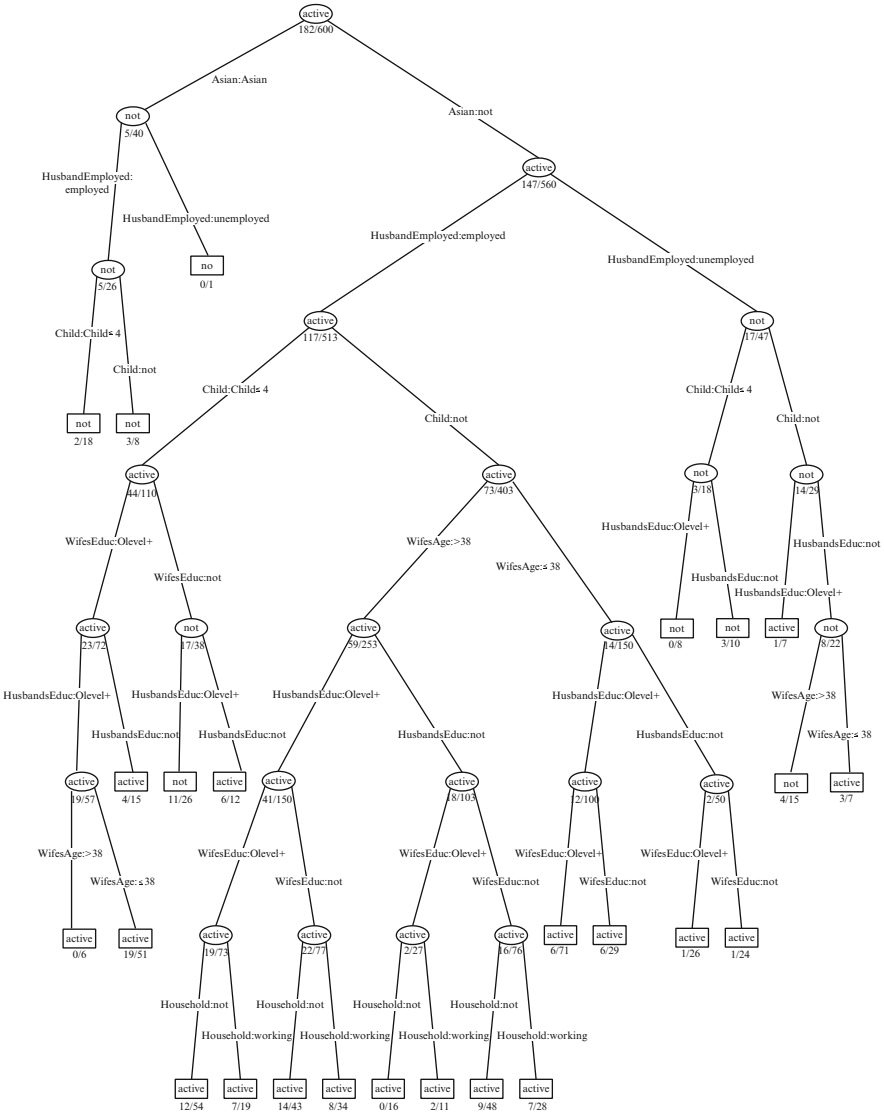


Fig. 28.2 An example of a binary classification tree. Goal of the classification was to describe households, in which the wife was economically active. Each branch is labeled according to the current splitting rule. The label in each node indicates the classification taken by majority vote. The numbers below each node give the number of misclassified observations and the total number of observations present in this node

These families mainly differ in the type of splitting criterion they use and in the background of their origin. The AID family grew out of the social sciences and uses the χ^2-statistic for contingency tables. The ML family has its origin in computer

science, while the CART family is more oriented to statistics using the concept of impurity. Breiman et al. (1984) defined for each node in a tree a measure of *impurity*.

Definition 1. Let $c_1, \ldots, c_K$ be the values of the target variable and $P(c_i \mid n), i = 1, \ldots, K$, the (estimated) probability of class c_i in node n (then $\sum_{i=1}^{K} P(c_i \mid n) = 1$).

The *impurity* of a node n is a nonnegative function $i(n)$ such that:

1. $i(n)$ has its only maximum for $P(c_1 \mid n) = \ldots = P(c_K \mid n) = \frac{1}{K}$, i.e. the node is as "impure" as possible.
2. $i(n)$ is minimal if $\exists i \in 1, \ldots, K$ such that $P(c_i \mid n) = 1$, i.e. the node contains only cases, which have the same target value.

Some very common splitting criteria based upon impurity measures are:

- Entropy

$$i(n) = -\sum_{j=1}^{K} P(j|n) \log P(j|n).$$

- the Gini index of diversity

$$i(n) = \sum_{i \neq j} P(i|n) P(j|n).$$

Another splitting rule, which is not based on an impurity measure, is the *twoing rule*: A node n is split into a left and a right node n_L and n_R such that

$$\frac{P(n_L)P(n_R)}{4} \left(\sum_{j=1}^{K} |P(j \mid n_L) - P(j \mid n_R)| \right)^2$$

is maximised. For binary targets this rule coincides with the CHAID criterion, which calculates the χ^2 value of a 2×2 table.

As Breiman et al. (1984) point out, the choice of the algorithm used is not as crucial as is generally thought:

> within a wide range of splitting criteria the properties of the final tree selected are surprisingly insensitive to the choice of splitting rule (p.38).

Hand (1997), however, mentioned several problems concerning impurity functions for splitting nodes, for instance

> It [the Gini index] has a tendency to produce offspring nodes that are of equal size (p. 69).

Unfortunately, there is no such thing as an optimal splitting criterion. "Optimal" splits very strongly depend on the specific application. When a decision tree is grown, many of the branches will reflect particularities of the training data at hand and will not generalize very well to the test data. This phenomenon is called

overfitting, and pruning the tree is a method to address this issue. The prepruning approach tries to implement the pruning process already in the growing phase. For this purpose, an additional stopping criterion is built in. At each node, a split is only performed if the information gain exceeds a certain threshold. Postpruning reduces the complexity of a tree model by removing the branches of some nodes. Postpruning is more effective, especially when the decision to remove a branch is based on a diversity measure that differs from the one used in the growing phase. Prepruning requires less computation but typically does not avoid the problem of overfitting and leads to rather unreliable trees. Postpruning is often a semi-automated process, including manual interactive pruning as well as cross-validation techniques.

Classification trees have been used as a role model for a predictor in bagging (Breiman 1996). Bagging as well as boosting (Freund and Schapire 1999) are used to improve the accuracy of a single prediction method by gaining stability and robustness, see also Chap. III.18.

Neural Networks

Artificial neural networks are one of the most prominent data mining techniques. Their fame is twofold: famous for astonishing good prediction results, unfamous for their black box behavior and lack of reproducibility of achievements. Neural networks are a classical method for predictive modeling. They have been used for classification and prediction to a similar extent. The basic idea of neural networks is the perceptron (see Fig. 28.3), a feedforward neural network with an input layer and an output layer.

The nodes of the input layer serve to introduce the values of the input variables. In a supervised learning situation with p explanatory variables and n cases, we hence get a network with np input nodes and n output nodes. For each output node $k = 1, \ldots, n$ the model output is calculated as a weighted sum of transformed input values

$$o_k = f(a_k) = f(\sum_{j=1}^{np} w_{jk} x_j).$$

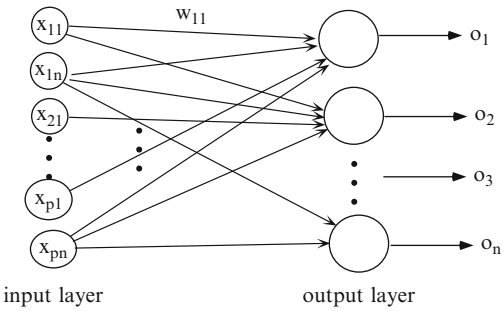


Fig. 28.3 Model of a perceptron with n input nodes and n output nodes

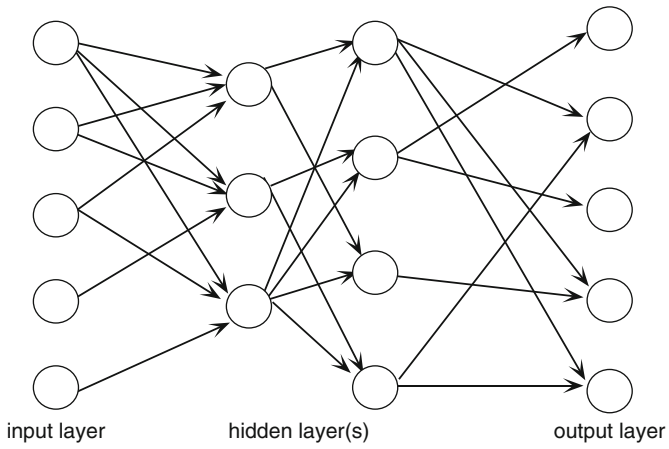


Fig. 28.4 Multi-Layer Perceptron

The transformation function f is usually called the activation function. The basic principle of learning a perceptron is the iterative adaptation of the weights w_{jk} in such a way that the error between the observed target values y_i and the model output o_i is a s small as possible. The most common delta rule is a gradient method with a tuning parameter η also known as "learning rate." The choice of η compromises between run time and convergence considerations.

The simple perceptron only allows to solve linearly separable problems. To address more complex problems a multi-layer perceptron, also called feedforward network must be used. A multi-layer perceptron (see Fig. 28.4) introduces additional layers, so-called hidden layers, into the network topology. Each node in the hidden and output layers operates in the same way as an output node in the perceptron. The lack of original target values for the nodes in the hidden layers is remedied by the backpropagation strategy. This means that the present network topology is used in two ways: as a feedforward network to propagate the observed values of the explanatory variables to the output layer and in reverse order to propagate the errors of fitted values back to the input layer.

A huge variety of different models can now be achieved by using different activation functions or different network architectures, i.e. by specifying the links between nodes. Usually, one uses the same activation function for all nodes in the hidden layers. The standard way of network architecture is to fully connect all neurons to all of the units in the preceding layer. However, it is possible to define networks that are partially-connected to only some units in the preceding layer. Typically, the resulting fitted values will be the same (or at least very similar) for different network architectures, summarizing in a nutshell the characteristics of a neural net: often very good in predicting and fitting values but giving very few insight into the functional relationship between explanatory variables and target variable. Putting it the other way round: neural nets are the method of choice, if you

have no idea about the functional relationship between explanatory variables and dependent variable. If you have a strong hypothesis on the functional relationship it is usually preferable to include this knowledge in the modeling process.

Support Vector Machines

Support vector machines are by now a must try for almost any large scale classification or regression task. Going back to an idea by Vapnik (1979), the support vector machine constitutes a linear classifier and became popular in the second half of the 1990s (Vapnik 1995). By combining the linear classifier with a problem specific kernel function (performing the so-called kernel trick), the support vector machine operates in reproducing Kernel-Hilbert-spaces and hence can perform non-linear classification and regression tasks. The modular set-up allows the construction of different support vector machines that are characterized by different non-linear decision surfaces. A detailed treatment and discussion of support vector machines is given in Chap. III.15.

Memory Based Reasoning

Memory-Based Reasoning (MBR) tries to mimic human behavior in an automatic way. Memories of specific events are used directly to make decisions, rather than indirectly (as in systems which use experience to infer rules). MBR is a two step procedure: first, identifying similar cases from experience, secondly, applying the information from these cases to new cases. MBR is specifically well suited to non-numerical data. MBR needs a distance measure to assign dissimilarity of two observations and a combination function to combine the results from the neighboring points to achieve an answer. Generating examples is much easier than generating rules which makes MBR so attractive. However, applying rules to new observations is much easier and faster than comparing new cases to a bulk of memorized objects.

Association Rules

Rule induction methods are widely applied tools for mining large data bases. They are often used as a starting point in undirected data mining, i.e. when you do not know what specific patterns to look for. One form of rule induction methods are association rules well-known in market basket analysis. They were proposed by Agrawal et al. (1993) with the intention to provide an automated process, which could find connections among items, that were not known before, especially to answer questions like: “which items are likely to be bought together?”. Many other areas of applications have been named from customer-tailored packages in the telecommunication or insurance business to analyzing web links.

In general, an *association rule* is an implication of the form $X \rightarrow Y$, where X and Y are mutually exclusive item sets. The quality of an association rule $X \rightarrow Y$ is measured by two criteria: confidence and support . A rule holds with *confidence* $c = c(X \rightarrow Y)$, if $c\%$ of transactions in D that contain X also contain Y. The rule $X \rightarrow Y$ has *support* s in the database D, if $s\%$ of transactions in D contain $X \cup Y$.

Most data mining software offers a procedure to generate all association rules with confidence and support that exceed some user-specified minimum thresholds for support (*minsup*) and confidence (*minconf*). The procedures are typically based on the a priori algorithm introduced by Agrawal and Srikant (1994).

There are several problems related to this procedure: the setting of the thresholds of minimal support and confidence is crucial; choosing high support and confidence may lead to "uninteresting" results – insofar as the resulting rules are often trivial or well known beforehand by domain experts (Weber 1998). Lowering the minimal thresholds can lead to a vast increase of the number of results. The standard approach is hence to use low thresholds for generating a large number of rules and then prune these rules to a manageable number of patterns. Pruning methods are based on objective or subjective interestingness measures. Strength and weaknesses of a broad class of interestingness measures are analyzed and discussed in McGarry (2005). Confidence, support, information gain, Gini-coefficient, entropy, and lift are some of the widely used objective measures (Bayardo and Agrawal 1999). Subjective measures try to incorporate user and domain knowledge to adapt the resulting rules to the current situation: a pattern which is of interest to one user may be of no interest to another user (Silberschatz and Tuzhilin 1995). However, it is typically not easy for the user to put his/her expectations and knowledge into automatic procedures of rule pruning. Klemettinen et al. (1994) use rule templates to model the user's knowledge, while Silberschatz and Tuzhilin (1995) introduce belief systems to describe users' expectations.

Another source of problems stems from the inability to estimate the quality of a rule merely from the two keys *confidence* and *support*. Graphical solutions showing a rule in the background of the corresponding contingency table have been proposed by Hofmann et al. (2000). Interactive methods further enhance these displays to powerful tools in the exploration of association rules, see Hofmann and Wilhelm (2001).

28.5.2 Visual Data Mining

Data visualization can contribute to the Data Mining process in many different ways, primarily because the human visual system is excellent in discovering unexpected patterns. Visual data mining does not replace other methods, but it complements analytic approaches. Standard numerical methods to detect outliers and erroneous observations are easier to automate and they may uncover most of the problems in a data set but graphic displays are excellent at identifying and understanding new and unusual problems in the quality of the data. Visualization techniques have met interest in the data base and data mining community since the early 90s, where

they have been extensively used to represent results of dynamic data base queries (Keim 1995; Rogowitz and Treinish 1996; Rogowitz et al. 1996; Shneiderman 1994). The parameters of the query are visualized by sliders each representing the range of one query parameter. The user can change the sliders interactively and the query results are shown in a linked graphic. Different methods to represent the distances between the query and the data items have been proposed in the literature: pixel-oriented techniques (Keim 1997) different intensities of the highlighting color (Tweedie and Spence 1998), or the standard linked views approach using a $\{0, 1\}$-distance (Derthick et al. 1997). More recent discussions of visual approaches to data mining tend to use complex static graphics more suited for presentation than for analysis. (For examples, take a look at websites concerned with Data Mining.) This may be for two reasons. Graphics research in computer science has been concerned with sophisticated, computing-intensive displays (for instance in scientific visualisation) and so it is natural to develop methods of this kind. On the statistical side, commercial software lags substantially behind graphical research. Few packages provide the flexibility and interactivity in graphics that is essential for exploratory work and, of those, even fewer have made provision for the display and graphical investigation of large data sets. Looking at the scatterplots produced by software for large numbers of data points can reveal more about the software than about the data. The capabilities of graphic displays for initial explorations of a data set are past comparison. Graphic exploration to grasp the main structural features and to get to know the data before beginning formal analysis can be carried out swiftly and informatively. It is not just the graphic displays themselves that are necessary, but the ability to directly work with them: to query points, symbols or axes; to select and link cases across displays; to sort and group data; to rescale and zoom. Interactivity not only means that the user can interact with the data, but also that the results from the changes made by the user can be seen instantaneously. A rapid and responsive interaction facilitates active exploration in a manner that is inconceivable with static displays. Users can start to pose "What if" queries spontaneously as they work through a task. Therefore, interactive displays not only offer the possibility of comparing resulting static views of different aspects of the data, they even encourage to draw conclusions from the way things are changing. Visualisation is also valuable for checking, filtering and comparing results from analytical procedures, and communication of the final outcome to the data base owner and the decision makers is indispensable without charts. At all these stages of the knowledge discovery process, at which contact with domain specialists is important to turn data into knowledge, the advantages of graphical presentation of ideas to enhance communication are considerable.

Research on interactive principles for statistical graphics can be categorized into two classes: firstly, development of innovative tools that help making a single display flexible and dynamic, and secondly, development of tools that operate on the underlying data and therefore have impacts to all displays showing the same data. Common tools of the first class are for example interactive modifiers of the bar width of a histogram, zooming in dot or scatter plots as well as slider-controlled dynamic changes in a graphic. The core of the second class are selection mechanisms and linking. Various selection modes that can even be combined to a sequence help in

choosing a specific set of data points to assign interest to them. Linking is the basic concept giving graphical methods the capability of multivariate analysis. Linking builds up a relation between different observations and between different displays. It propagates the changes and selections made in one plot to all other connected plots that deal with the same database.

Visualising Large Numbers of Cases

Data Mining is statistics at scale and speed. Large data sets can be large in two different aspects. First, if the size of the sample investigated is fairly large it will result in a large quantity of observations, the number of data points possibly going towards the billions. Secondly, a large data set can also arise from investigations with a large number of variables. For graphical as well as for analytical procedures both these issues pose problems and require new approaches for solution.

Graphical displays are often used to provide an easy overview of the data from which the global structure can be rapidly deduced while it is still possible at the same time to spot individual features. For small data sets one can represent each data point by a single graphical element, as for example in scatter plots in the form of small circles, to ensure that all the information in the data is also represented in the display. However, the use of point based displays reaches its limits as the number of observations increases. A computer screen with about 1280×1024 pixels screen size will possibly offer about one million pixels to be used for displaying points the others needed for frames, scroll bars, legends, and scales. Assuming that we need about five pixels to make a point clearly visible a natural limit for point based displays would lie at about 200,000 observations. This number decreases even more if we take into account that the more structure a data set shows the less spread out are the points over the display's range. Thus, most realistic data sets will only use about a quarter of the available area in a graphic and thus induce that we only can show about 50,000 observations in one point-based plot.

Analytical methods reduce the dimensions and in the extreme case condense the data into one single number, like the mean or a regression coefficient. The graphical analogue is to use one single graphical element to represent the data. A single smooth density curve for example can be used to display the univariate distributional properties of data. Note here, that as in this example, a single graphical element can still convey much more information than a single number. The use of smooth density curves is not only marked by a reduction in terms of graphical elements but also by a transition from using position to encode information to using area instead. Histograms have a long tradition in statistics for being used to display distributional information. They are computationally simpler than smooth density curves and indicate their local constructional properties more clearly. So, for large data sets, area based graphics are to be preferred.

Graphical elements overlap when the values they represent fall too close together on the scale used in the current display. Brightness of the graphical elements can be used to visualize the extent of overplotting. The more graphical elements overlap

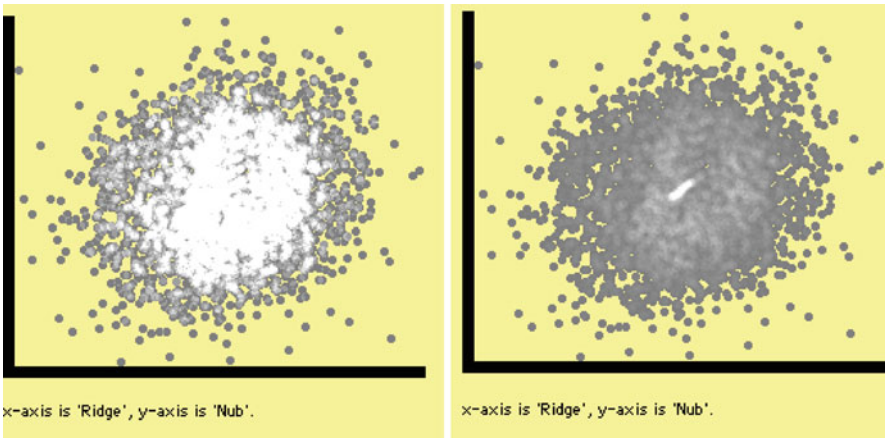


Fig. 28.5 Tonal highlighting is used to show the amount of overplotting. Increasing the brightness parameter with the overall density of the plot points towards areas with relatively high overplotting. The left plot shows the default setting. The plot on the right with increased brightness parameter clearly shows a pencil shape region in the center with a very high density

the brighter they are drawn. One implementation of this is tonal highlighting as provided for dotplots and scatterplots in MANET (Hofmann 2000). This procedure can be seen as a special kernel density estimation using a uniform kernel. Tonal highlighting and its visual effect is based on two parameters that can be interactively varied: the size of the graphical points used for displaying data and the maximum brightness parameter that is the number of data points that have to overlap to yield the maximum brightness.

For sparse plots brightness should change already with little overplotting, for dense plots a stronger overplotting is needed to cause a change in brightness, see Fig. 28.5.

In contrast to point oriented displays in which dense areas cause problems due to overplotting, it is more likely that area based displays run into difficulties when regions have too few data points. The counts for such regions might be too small to result in an area that is perceptible to the human eye. Especially, histograms for highly unequally dense regions will cause representational problems: according to the scale used either the high density regions are visible or the low density regions.

Linking leads to a technical difficulty associated with large data sets and particularly relevant to data quality. Even with a high-resolution screen it may be necessary for each pixel to represent many points. If the number of points to be drawn or highlighted is too small to be represented, then those points will be invisible. This can lead to false judgments, for instance in masking unusual cases, and needs to be avoided. One solution, which works well in practice, has been implemented in the software MANET. Red-marking is used to indicate that some information may be hidden. For histograms and barcharts a red line is drawn under any bar where information may be hidden. It could be that the bar itself is too small

to be drawn (which can happen with extreme values in histogram displays); it could be that too few are highlighted in a bar for any highlighting to show; it could be that too few are not highlighted in a bar so that the whole bar is, misleadingly, completely highlighted. A related problem arises in the display of spatial data: very small areas (often the most populous!) may not be shown and special red-marking techniques are needed to take account of this.

Direct manipulation graphics can alleviate this problem further when there is the possibility of easily changing the scale of a display. Here it is by no means sufficient to have a parameter tool box in which we can insert our desired scaling. We must be able to grab the vertical axis in a histogram to shrink or expand a scale by simply dragging it as well as breaking a scale and using different scalings at different parts of the range. Such a feature can be thought of as a particular variant of logical zooming. While standard zooming enlarges the displayed graphical elements, logical zooming works on the underlying model and changes it to display more details. Logical zooming is quite natural when working with maps, starting with a country map, we zoom into a regional map, a city map and finally a street map that shows us the neighborhood in every detail.

Logical zooming for the vertical axis aims at balancing out the needs for a good overview and the need for focussing on individual features. Area based displays like histograms aggregate the data and lose the ability to show anomalous behavior of single points. For large data sets this is not an unwelcome feature since huge data sets might contain too many features to investigate. So the researcher will only focus on phenomena that occur frequently enough.

Logical zooming is even more important when used on the x-axis to change the level of aggregation. Again, there is not only the need to zoom in a homogeneous way such that all bins of a histogram are treated in the same manner and use the same scaling for all of them. Rather often, the analyst might be interested in particular regions, for example in the center of a distribution to see whether there are any gaps that just might be smoothed away by the graphical representation. Thus, a very common interest is zooming into one region while still keeping the general overview.

Logical zooming is valuable for all statistical displays that aggregate the data: for boxplots, for example, logical zooming should result in displaying the dotplot for the selected section or a boxplot based on the selected points only. Logical zooming in mosaic plots would help investigating how additional variables could split up a large cell. Hot selection as provided in DataDesk (Velleman 2000) can be seen as an elementary form of logical zooming.

Visualising Large Numbers of Variables

Common orthogonal coordinate systems have at most three orthogonal axes for visualization at hand. To display more variables and typically most data sets will comprise a much larger number of variables projection techniques are widely used to reduce the dimensionality to a manageable size. Other approaches are using matrix

layouts for scatterplots and providing brushing techniques to visually connect points that belong together, but this only works for a low number of dimensions. Another approach is to use parallel coordinate systems that can deal with a larger number of variables simultaneously. The parallel coordinate display (Inselberg 1985; Wegman 1990) sacrifices orthogonal axes by drawing the axes parallel to each other resulting in a planar diagram where each d–dimensional point $(x_1, \ldots, x_d)$ is uniquely represented by a broken line. Current screens limit the numbers of variables that can be simultaneously shown in parallel coordinate plots to about 30 – but, of course, scrolling windows offer in principle unlimited capabilities.

The individual parallel coordinate axes represent one–dimensional projections of the data. Usually, different variables will be based on different units of measurement. Using the original scale might make inefficient use of the screen space. Using standardized variables will ameliorate that. In certain instances the original scale will, however, display valuable information. An easy interactive change of this scaling is thus a particularly welcome feature. Pairwise relationships for adjacent variables are much more easily seen than for nonadjacent variables. Since a complete parallel coordinate investigation would require running through all possible permutations, interactive facilities for manually or automatically changing the order of the variables are needed.

Due to the point – line duality between a parallel coordinate system and a cartesian coordinate system the correlation of two adjacent variables is depicted by the mutual position of the line segments: parallel line segments indicate a strong positive correlation, a line crossing in a single point means strong negative correlation. Since it is much easier to recognize an intersection point than almost parallel lines, negative correlation is simpler to detect. It is helpful to have an interactive tool to invert the range of a single variable to turn positive correlation into a negative one. CASSATT is a JAVA application that offers a wide variety of interactive features for parallel coordinate plots, see Winkler (2000) . Linking and highlighting interactive statistical graphics increases the dimensionality of data that can be explored. For highly multivariate data (i.e., more than ten to twenty variables) insight into the data by linking low-dimensional plots can be limited. Thus the need for high-dimensional plots arises. These plots–for example, rotating plots (grand tour, projection pursuit, see Chap. II.12), parallel coordinate plots, or mosaic plots–can incorporate up to ten or more variables in a single plot. Linked highlighting and alterations inside these plots (e.g., zooming, reordering, or sorting) offer high-dimensional insights into data sets. Multiple selections via selection sequences offer a convenient way of interacting with high-dimensional subsets of the data using low-dimensional plots.

Visualizing Association Rules

Visualizing association rules aims at solving some major problems that come with association rules. First of all the rules found by automatic procedures must be filtered. Depending on what minimum confidence and what support is specified a vast amount of rules may be generated. Among those, however, not only “interesting

and new" results – according to the principle of KDD – are found. In supermarket data for instance most purchases will also contain plastic or paper bags. The naive filtering approach that searches for the rules with highest confidence and/or highest support fails because it yields rules that are typically already known before and are thus not of interest. So, in many cases rules that just do not pass the thresholds can be economically better exploited and are therefore higher rated. Association rules tend to prefer frequent events as their response (right hand side of a rule). If $P(Y)$ is large, then it is very likely for any small event X, that $P(Y|X)$ is higher than the minimal confidence threshold. The meaning of rules found in this way, on the other hand, is more than questionable. When dealing with different association rules which refer to the same response, it is of interest to examine, whether the populations described by the explanatory variables (left hand side of a rule) differ from each other. It well could be one and the same population, and different rules providing only different descriptions for it. – Another goal therefore is, to examine the impact of rules on one another and to find intersections amongst them. Quite often, slight variations of items are listed as different association rules: either the same items alternatively take right and left hand sides of rules or sequences occur.

Besides finding new and interesting results, data mining tools are to find *explicable* results. The interpretation of results therefore should be of great interest – if one can not explain a rule at first, we could use methods of cross-classification in the hope of finding clues within the data, which allow us to decide, whether a result has to be considered as a peculiarity of this particular data set or can be generalized.

Since support and confidence are equally important for the conclusions drawn from association rules any approach should visualize support and confidence of competing rules within one display. Figure 28.6 shows a standard visualization of association rules as a matrix of all left and right hand sides of rules. Left hand sides are the rows, right hand sides the columns of the matrix. Each rule, which fulfills $minsupp$ and $minconf$ is drawn as a square. The size of which is given by the actual support of the rule.

This approach is rather unsatisfactory since it reaches the space limits already for a small number of association rules. It also uses two different visual attributes to encode support and confidence of a rule: color for the confidence and size for the support. The ordering of colors is difficult and in no way unambiguous and depends heavily on the used scale of the color scheme. The encoding of the support by the size of the squares is visually problematic, since length of the square is used instead of area. For instance the rules

$$\textit{turkey \& hering \& corned beef} \rightarrow \textit{olives}$$
$$\textit{ham \& corned beef \& apples} \rightarrow \textit{olives}$$

have support 11.19% and 3.1%, respectively. The factor is approx. 4, whereas the areas differ with factor 16 which yields a lie factor as defined by Tufte (1983) of 400% .

Mosaic plots as introduced by Hartigan and Kleiner (1981) are a graphical analogue to multivariate contingency tables. They show the frequencies in a contingency table as a collection of rectangles whose areas represent the cell frequencies. Large areas therefore represent large cells. The shape of a tile is

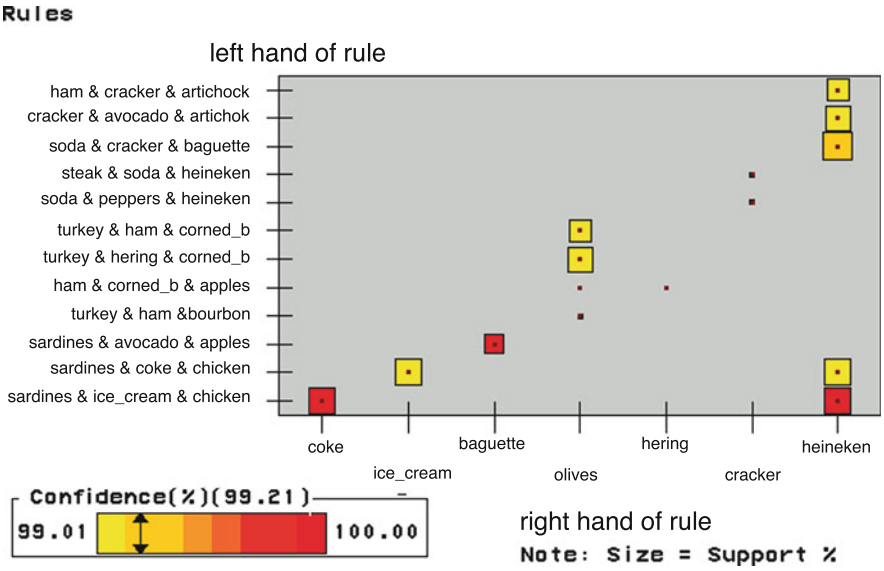


Fig. 28.6 SAS Enterprise Miner: Visualisation of all association rules with a minimum confidence of 99%. Each rule is represented by a square, the area of which corresponds to the support of the rule. Color is used to encode the confidence of the rule

calculated during the (strictly hierarchical) construction. In classical mosaic plots alternately width and height of each bin is split according to the categories of the next variable included, in such a way, that the area of each bin is proportional to the number of cases falling into this specific intersection of variables. Thus, viewed statically the mosaic plot gives a graphical estimation of the joint distribution of the variables contained. Interactive features are necessary to turn the mosaic plot into a powerful exploration tool. In Hofmann (1999) these interactive features have been explained with an example, the most essentials are *linked highlighting*, *rotating*, *navigating through dimensions*, *grouping categories*, and *variation of display*.

Linked highlighting provides the transition to conditional distributions and is essential for visualizing association rules. The basic approach is as follows: Combine all variables involved in the left-hand-side X of a rule as *explanatory* variables and draw them within one mosaicplot. Visualize the *response* Y, the right-hand-side of the rule, in a bar chart and then by highlighting a category in the bar chart every tile in the mosaic plot corresponds to an association rule. The amount of highlighting in a bin in the mosaic plot gives a visual measure for the support of a rule, the highlighting heights relative to the bin's height give the rule's confidence.

Figure 28.7 shows an overview of all possible association rules involving three binary variables X_1, X_2 and Y.

The bin in the bottom right corner ($x_1 \cap x_2 \cap x_3$) gives an example for a rule that passes most common thresholds. It has very high confidence (the highlighting almost fills this bin entirely), and also the support is relatively high (the bin itself,

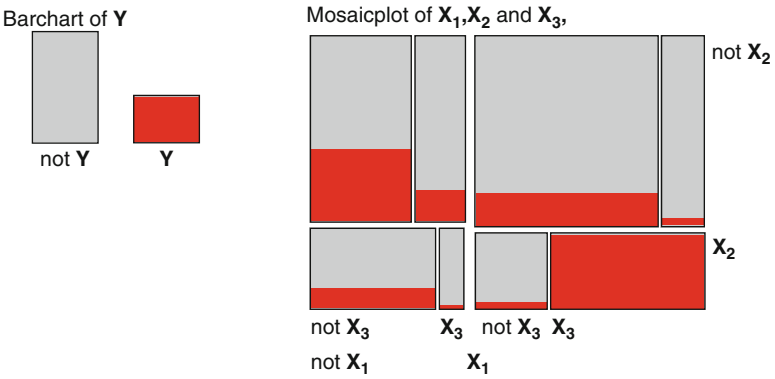


Fig. 28.7 Mosaic plot of all possible association rules of X_1, X_2, X_3 and Y. The amount of highlighting in a bin represents the support of a rule, while its confidence is measured by the proportion of highlighting within a tile

and therefore the amount of highlighting, is large). The leftmost bin in the upper row, (not $x_1 \cap x_2 \cap$ not x_3), represents a rule with a comparable support (this bin contains approximately the same amount of highlighting as the bin in the bottom right corner), yet the confidence of the corresponding rule is rather low (highlighting fills this bin to only one third, approximately). All the other possible rules have even lower confidence and their supports are also too small to be of interest.

Rotated mosaic plots:

To make comparisons of proportions of highlighting heights more easily, it is much better to align all bins. This yields mosaics of the following form: Starting from the basic rectangle, the width is divided according to the first variable's categories and their numbers of entries. Each of these bins is divided again and again horizontally in the same way according to the other variables. In MANET standard mosaic plots can be interactively rotated in this form by a simple mouse click and therefore we call these plots *rotated mosaic plots*. Highlighting splits the bins still vertically. Thus highlighting heights in a p dimensional mosaic plot of variables $X_1, \ldots, X_p$ show the conditional probabilities $P(h|X_1, \ldots, X_p)$.

In addition, to determine the exact combination that a bin represents more easily labels can be added underneath the mosaics (see Fig. 28.8). Each row of the labels corresponds to one variable, the white rectangles stand for "0"s, the blacks for "1"s. The first bin in Fig. 28.8 therefore represents the combination of "no item bought" (all "0"s), the second bin contains all transactions, where only *corned beef* has been bought, and so on. This form of diagrams is also known as doubledecker plots introduced in Hofmann et al. (2000).

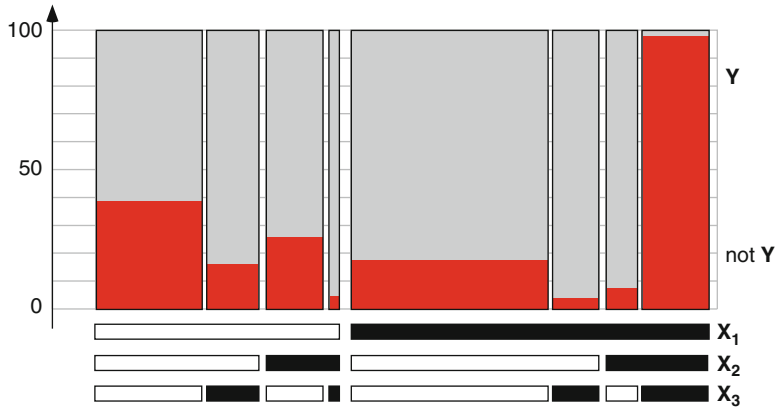


Fig. 28.8 Doubledecker plot of three variables. The rightmost bin corresponds to the rule *heineken & coke & chicken → sardines*

28.6 Trends and Controversies

Data mining and knowledge discovery is surely an important research area with a lot of potential, but it is not a universal panacea. The ever increasing availability of data and the increasing speed in which this data is collected and fed into the analysis pipeline is not asking for even more sophisticated modeling and analysis techniques with a high component of human involvement, but it calls for flexible, fast and stable search heuristics. So, a major challenge is to incorporate as much of the already available sophistications of models and the flexibility of the human mind into the automated data analysis processes. Progress has been made in this respect, but we are still a long way to go.

Surely, the amount of data to be analyzed will grow, presumably at a similar rate as the speed of the analysis tools will grow. What has been a large data set ten years ago, is now a toy class-room example.What is massive now, will fit in the main memory in a few years time. Streaming data is to be analyzed to make decisions in real-time and to keep up with important security issues in many fields (cf Gaber et al. 2005). Data handling facilities will improve and speed optimization of algorithms will help, but the fundamental race between data growth and decreasing reaction time will remain.

Also remain will the problem of data quality and the enormous amount of time and effort that is needed for proper data cleaning. A small data set which is collected with care, due diligence and a proper sampling scheme, will typically yield more reliable results than fast search heuristics in massive data sets. A further discussion on the use of elaborated sampling techniques within data mining might be a worthwhile endeavour.

With all successes in finding patterns and trends and with a shift away from analysing the mean behaviour towards exploring the unusual cases, data protection

and privacy issues need to be incorporated in an appropriate way. Data mining has matured as a discipline and will enjoy its adulthood.

References

Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. Technical Report RJ9839, IBM (1994); IBM Research Report RJ9839.

Agrawal, R., Imielinski, T., Swami, A.: Mining associations between sets of items in massive databases. In Proceedings of the ACM-SIGMOD 1993 International Conference on Management of Data, pp. 207–216, Washington, DC (1993)

Bayardo, R.J., Agrawal, R.: Mining the most interesting rules. In Proceedings of the 5th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 145–154 (1999)

Brachman, R., Anand, T.: The process of knowledge discovery in databases. In Advances in Knowledge Discovery and Data Mining, pp. 37–57. AAAI Press/The MIT Press, Cambridge, MA (1996)

Breiman, L.: Bagging predictors. Mach. Lear. **24**(2), 123–140 (1996); URL citeseer.nj.nec.com/breiman96bagging.html.

Breiman, L., Friedman, J.H., Olshen, R., Stone, C.: Classification and Regression Trees. Wadsworth, Belmont, CA (1984)

Chau, M., Xu, J.: Using web mining and social network analysis to study the emergence of cyber communities in blogs. In: Chen, H., Reid, E., Sinai, J., Silke, A., Ganoz, B. (eds.) Terrorism Informatics: Knowledge Management and Data Mining for Homeland Security, pp. 197–220. Springer, Berlin (2008)

Chen, H., Reid, E., Sinai, J., Silke, A., Ganoz, B.: Terrorism informatics: Knowledge management and data mining for homeland security. Springer (2008)

Derthick, M., Kolojejchick, J., Roth, S.F.: An interactive visualization environment for data exploration. Technical report, Carnegie Mellon University, Pittsburgh, PA (1997)

Ding, Q., Ding, Q., Perrizo, W.: Association rule mining on remotely sensed images using p-trees. In Cheng, M.-S., Yu, P.S., Liu, B. (eds) PAKDD, Lecture Notes in Computer Science, vol. 2336, pp. 66–79. Springer, Berlin (2002)

Fayyad, U., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.) Advances in Knowledge Discovery and Data Mining. AAAI/MIT Press, Cambridge, MA (1996)

Fienberg, S.: Homeland insecurity: Data mining, privacy, disclosure limitation, and the hunt for terroroists. In: Chen, H., Reid, E., Sinai, J., Silke, A., Ganoz, B. (eds.) Terrorism Informatics: Knowledge Management and Data Mining for Homeland Security, pp. 197–220. Springer, Berlin (2008)

Freund, Y., Schapire, R.: A short introduction to boosting (1999); URL citeseer.nj.nec.com/freund99short.html.

Friedman, J.H.: Data mining and statistics: What's the connection? In Computing Science and Statistics: Proceedings of the 29th Symposium on the Interface. Interface Foundation of North America (1998)

Gaber, M.M., Zaslavsky, A., Krishnaswamy, S.: Mining data streams: a review. SIGMOD Rec. **34**(2), 18–26 (2005); ISSN 0163-5808. doi: http://doi.acm.org/10.1145/1083784.1083789.

Hand, D.J.: Construction and Assessment of Classification Rules. Wiley, Chichester (1997)

Hand, D.J.: Data mining: statistics and more? Am. Stat. **52**, 112–118 (1998)

Hand, D.J., Mannila, H., Smyth, P.: Principles of Data Mining. MIT Press, Cambridge, MA (2001)

Härdle, W.: *Smoothing Techniques with implementation in S*. Springer, New York (1991)

Hartigan, J., Kleiner, B.: Mosaics for contingency tables. In Computer Science and Statistics: Proceedings of the 13th Symposium on the Interface, pp. 268–273, Springer, New York (1981)

Hofmann, H.: Simpson on board the Titanic? Stat. Comput. Graphics Newsletter **9**, 16–19 (1999)

Hofmann, H.: MANET. http://stats.math.uni-augsburg.de/manet (2000)

Hofmann, H., Wilhelm, A.: Visual comparison of association rules. Comput. Stat. **16**, 399–415 (2001)
Hofmann, H., Siebes, A., Wilhelm, A.: Visualizing association rules with interactive mosaic plots. In Proceedings of the 6th International Conference on Knowledge Discovery and Data Mining, pp. 227–235, Boston, MA (2000)
Inselberg, A.: The plane with parallel coordinates. Visual Comput. **1**, 69–91 (1985)
Keim, D.A.: Enhancing the visual clustering of query-dependent data visualization techniques using screen-filling curves. In Proceedings International Workshop on Database Issues in Data Visualization, Atlanta, GA (1995)
Keim, D.A.: Visual techniques for exploring databases. Tutorial Notes: Third International Conference on Knowledge Discovery and Data Mining, pp. 1–121 (1997)
Klemettinen, M., Mannila, H., Ronkainen, P., Toivonen, H., Verkamo, A.: Finding interesting rules from large sets of discovered association rules. In Proceedings of the Third International Conference on Information and Knowledge Management CIKM-94, pp. 401–407 (1994)
Kloesgen, W., Zytkow, J.M.: Knowledge discovery in database terminology. In: Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., Uthurusamy, R. (eds.) Advances in Knowledge Discovery and Data Mining, pp. 573–592. AAAI/MIT Press, Cambridge, MA (1996)
Lowe, D.G.: Object recognition from local scale-invariant features. In ICCV '99: Proceedings of the International Conference on Computer Vision- Volume 2, pages 1150–1157, Washington, DC, USA, 1999. IEEE Computer Society.
McGarry, K.: A survey of interestingness measures for knowledge discovery. Know. Eng. Rev. **20**(01), 39–61 (2005)
Ordonez, C., Omiecinski, E.: Discovering association rules based on image content. In ADL, pp. 38–49 (1999)
Rogowitz, B.E., Treinish, L.A.: How not to lie with visualization. Comput. Phys. **10**, 268–273 (1996)
Rogowitz, B.E., Rabenhorst, D.A., Gerth, J.A., Kalin, E.B.: Visual cues for data mining. Technical report, IBM Research Division, Yorktown Heights, NY (1996)
Schober, J.-P., Hermes, T., Herzog, O.: Picturefinder: Description logics for semantic image retrieval. In Smeulders, A., Ip, H., Smeaton, Smith, J.R. (eds.), IEEE International Conference on Multimedia and Expo, 2005. ICME 2005, pages 1571–1574, Amsterdam, July 2005. IEEE.
Scott, D.W.: Multivariate Density Estimation. Wiley, New York (1992)
Shneiderman, B.: Dynamic queries for visual information seeking. IEEE Software **11**(6), 70–77 (1994)
Silberschatz, A., Tuzhilin, A.: On subjective measures of interestingness in knowledge discovery. In Proceedings of the First International Conference on Knowledge Discovery and Data Mining, pp. 275–281 (1995)
Sivic J., Zisserman, A.: Video google: A text retrieval approach to object matching in videos. In 9th IEEE International Conference on Computer Vision (ICCV), Proceedings of ICCV, pages 1470–1477, Nice, France, October 14–17 2003. IEE Computer Society.
Tešić, J.: Managing Large-scale Multimedia Repositories. PhD thesis (2004); URL http://vision.ece.ucsb.edu/publications/04ThesisTesic.pdf.
Tufte, E.R.: The Visual Display of Quantitative Information. Graphics Press, Cheshire, CT (1983)
Tukey, J.W.: Analyzing data: Sanctification or detective work? Am. Psychol. **24**, 83–91 (1969)
Tweedie, L., Spence, R.: The prosection matrix: A tool to support the interactive exploration of statistcal models and data. Comput. Stat. **13**, 65–76 (1998)
Unwin, A., Hofmann, H., Wilhelm, A.: Direct manipulation graphics for data mining. Int. J. Image Graph. **2**, 49–65 (2002)
Vapnik, V.: Estimation of dependences based on empirical data (in russian) (1979)
Vapnik, V.: The Nature of Statistical Learning Theory. Springer, New York (1995)
Velleman, P.: DataDesk. http://www.datadesk.com (2000)
Weber, I.: On pruning strategies for discovery of generalized and quantitative association rules. In Proceedings of Knowledge Discovery and Data Mining Workshop, Singapore (1998)

Wegman, E.J.: Hyperdimensional data analysis using parallel coordinates. J. Am. Stat. Assoc. **85**, 664–675 (1990)

Wilhelm, A., Jacobs, A., Hermes, T.: Association rule mining of multimedia content. In: Lauro, C., Palumbo, F., Greenacre, M. (eds) Data Analysis and Classification: from the exploratory to the confirmatory approach, pp. 179–186, Springer, Berlin (2009)

Winkler, S.: Parallele Koordinaten: Entwicklung einer interaktiven Software – CASSATT. Technical report, University of Augsburg, German (2000)

Zhang, H., Spiliopoulou, M., Mobasher, B., Giles, C., McCallum, A., Nasraoui, O., Srivastava, J., Yen, J.: Advances in Web Mining and Web Usage Analysis. Lect. Notes Comput. Sci. **5439** (2009)

Chapter 29
Recursive Partitioning and Tree-based Methods

Heping Zhang

29.1 Introduction

Tree-based methods have become one of the most flexible, intuitive, and powerful data analytic tools for exploring complex data structures. The applications of these methods are far reaching. They include financial firms (credit cards Altman 2002; Frydman et al. 2002 and investments Brennan et al. 2001; Pace 1995), manufacturing and marketing companies Levin et al. (1995), and pharmaceutical companies. In the past decades, there have been many applications in genomics and bioinformatics Zhang et al. (2001).

The best documented, and arguably most popular uses of tree-based methods are in biomedical research for which classification is a central issue. For example, a clinician or health scientist may be very interested in the following question Goldman et al. (1996), Goldman et al. (1982), Zhang et al. (2001): Is this patient with chest pain suffering a heart attack, or does he simply have a strained muscle? To answer this question, information on this patient must be collected, and a good diagnostic test utilizing such information must be in place. Tree-based methods provide one solution for constructing the diagnostic test.

Classification problems also frequently arise from engineering research. Bahl et al. (1989) introduced a tree-based language model for natural language speech recognition. Desilva and Hull (1994) used the idea of decision trees to detect proper nouns in document images. Geman and Jedynak (1996) used a related idea to form an active testing model for tracking roads in satellite images. In addition, decision trees have been used in scientific and social studies including astronomy Owens

H. Zhang (✉)
Yale University School of Medicine, New Haven, CT, USA

Sun Yat-Sen University, Guangzhou, China
e-mail: heping.zhang@yale.edu; http://c2s2.yale.edu

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_29,

et al. (1996), chemistry Chen et al. (1998) and politics (http://www.dtreg.com/housevotes.htm). I will revisit some of these applications later in detail.

Most commercial applications of tree-based methods have not been well-documented through peer reviewed publications. In 1999 I helped the CLARITAS, a marketing company, apply a tree-based method as described in Sect. 29.6 for marketing segmentation analysis Zhang (1998). Tree-based methods have also been frequently used in the drug development process. I have personally provided consultations to Aventis, Inc. and Eisai Inc. for drug approvals.

The purpose of this article is to provide an overview for the construction of the decision trees, and, particularly, the recursive partitioning technique, which is the thrust of this methodology. In their early applications, tree-based methods were developed primarily to facilitate the automation of classifications as an expert system Breiman et al. (1984), Friedman (1977), Wasson et al. (1985), although Morgan and Sonquist (1963) were motivated by the need to analyze survey data to identify interactions, particularly in the presence of non-numerical predictors. More recently, classification trees have not only been used for automated disease diagnosis, but also for selecting important variables that are associated with a disease or any response of interest Zhang and Bracken (1995), Zhang and Bracken (1996) Zhang and Singer (2010), Zhang et al. (2003), Zhang et al. (2001).

There are different approaches to classification. First, it can be done intuitively. For example, a physician or a group of physicians may use their experience in caring for patients with chest pain to form a subjective opinion or an empirical decision as to whether a new patient with chest pain is likely to suffer a heart attack, and consequently, decide what treatment is most appropriate. Secondly, methods in both statistical and machine learning literature have been developed, such as Fisher linear discriminant analysis Fisher (1936) and support vector machine Cristianini and Shawe-Taylor (2000). These methods have the parametric flavor in the sense that the classification rule has an explicit form with only a few parameters to be determined from a given sample that is usually referred to as learning sample.

Classification trees belong to the third type of methods for which we allow a very general structure, e.g., the binary tree as displayed in Fig. 29.1, but the number of "parameters" also needs to be determined from the data, and this number varies. For this reason, classification trees are regarded as nonparametric methods. They are adaptive to the data and are flexible, although the large number of quantities (or parameters) to be estimated from the data makes the classification rule more vulnerable to noise in the data.

To be more precise about the statistical problem, let us define the data structure and introduce some notations. Suppose that we have observed p covariates, denoted by a p-vector $\mathbf{x}$, and a response y for n individuals. For the ith individual, the measurements are

$$\mathbf{x}_i = (x_{i1}, \cdots, x_{ip})' \text{ and } y_i, \ i = 1, \cdots, n.$$

The objective is to model the probability distribution of $P(y \mid \mathbf{x})$ or a functional of this conditional distribution.

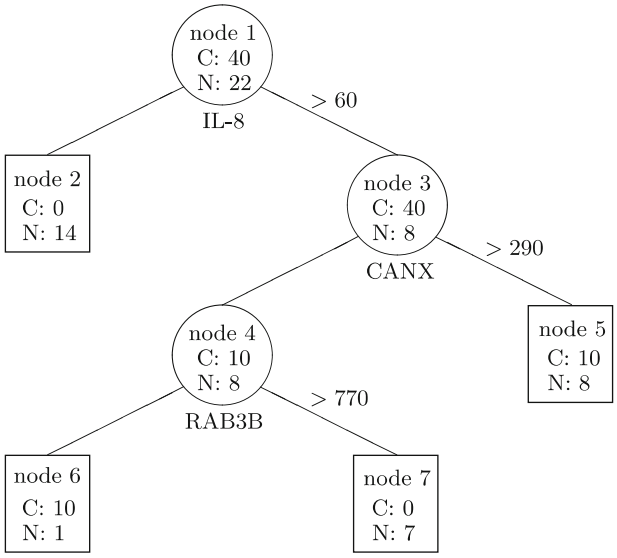


Fig. 29.1 Classification Tree for Colon Cancer Diagnosis Based on Gene Expression Data. Inside each node are the number of tumor (C) and normal (N) tissues. See Zhang et al. (2001) for more details

To appreciate how these variables are characterized in real applications, let me examine some of the published applications.

Example 1. Levin et al. (1995) described a probability-driven, customer-oriented decision support system for the marketing decisions of the Franklin Mint, a leading Philadelphia-based worldwide direct response marketer of quality collectibles and luxury home decor products. The purpose of the system is to target the "right" audience for each promotion from among a very large marketing database, based on the customers' attributes and characteristics. In this case, the customers' attributes and characteristics constitute the **x** variables. Whether the targeted client is desirable or not forms the basis for the response y.

Example 2. To screen large chemical databases in corporate collections and chemical libraries, Chen et al. (1998) used recursive partitioning to develop three-dimensional pharmacophores that can guide database screening, chemical library design, and lead optimization. Their idea was to encode the three-dimensional features of chemical compounds into bit strings, which we will refer to as the **x** variables. Then, those features are selected in relation to the biological activities (i.e., y) of the compounds. Here, each compound contributes an observation. Using this idea, the authors successfully retrieved three-dimensional structure-activity relationships from a large heterogeneous dataset of 1,644 monoamine oxidase inhibitors. I will revisit this example in detail in Sect. 29.4.

As we can see from the above examples, like any multivariate regression model, the covariates or predictors in $\mathbf{x}$ may contain variables that can be categorical (nominal or ordinal) or continuous. For example, ethnicity is usually treated as categorical data and age as continuous. Some of the covariates may have missing values. I will discuss how these missing values are handled in the tree framework. In a nutshell, unlike what is usually done in a simple linear regression, observations with missing information are not omitted from classification trees.

Not only can we have mixed types of predictors, but also the response variable can be discrete (binary or multiclass), continuous, and sometimes censored. The characteristics of the response, y, determines the method for estimating $P(y\ |\mathbf{x})$. I will review a variety of tree-based methods that are adaptable to the distribution of y. In Sect. 29.2, I will introduce the basic idea of classification trees using a dichotomous response. Section 29.2 is followed by some in-depth discussion of computational challenges and implementations in Sect. 29.3 and by examples in Sect. 29.4 to illustrate how we can interpret results from tree-based analyses. One of the most popular uses of tree-based methods is in the analysis of censored data in which y is the time to an event and is subject to censoring. As described in Sect. 29.5, such trees are referred to as survival trees Bacchetti and Segal (1995), Carmelli et al. (1991), Carmelli et al. (1997), Gordon and Olshen (1985), Zhang (1995). In Sect. 29.6, I will present an extension of the tree methodology to the classification of a response consisting of multiple components such as an array of respiratory symptoms Zhang (1998). Finally, I will conclude in Sect. 29.9 with some remarks on relatively recent developments such as forests and Bayesian trees. To illustrate the methods and their applications, some examples will be presented along with the methods.

29.2 Basic Classification Trees

I have highlighted some applications of decision trees. Here, I will explain how they are constructed. There has been a surge of interest lately in using decision trees to identify genes underlying complex diseases. For this reason, I will begin the explanation of the basic idea with a genomic example, and then will also discuss other examples.

Zhang and his colleagues Zhang et al. (2001) analyzed a data set from the expression profiles of 2,000 genes in 22 normal and 40 colon cancer tissues Alon et al. (1999). In this data set, the response y equals 0 or 1 according to whether the tissue is normal or with cancer. Each element of $\mathbf{x}$ is the expression profile for one of the 2,000 genes. The objective is to identify genes and to use them to construct a tree so that we can classify the tumor type according to the selected gene expression profiles. Figure 29.1 is a classification tree constructed from this data set. In what follows, I will explain how such a tree is constructed and how it can be interpreted.

29.2.1 Tree Growing and Recursive Partitioning

Tree construction usually comprises of two steps: growing and pruning. The growing step begins with the root node, which is the entire learning sample. In the present example, the root node contains the 62 tissues and it is labeled as node 1 on the top of Fig. 29.1. The most fundamental step in tree growing is to partition the root node into two subgroups, referred to as daughter nodes, such that one daughter node contains mostly cancer tissue and the other daughter node mostly normal tissue. Such a partition is chosen from all possible binary splits based on the 2,000 gene expression profiles via questions like "Is the expression level of gene 1 greater than 200?" A tissue is assigned to the right or left daughter according to whether the answer is yes or no. When all of the 62 tissues are assigned to either the left or right daughter nodes, the distribution in terms of the number of cancer tissues is assessed for both the left and right nodes using typically a node impurity. One of such criteria is the entropy function

$$i_t = -p_t \log(p_t) - (1 - p_t) \log(1 - p_t),$$

where p_t is the proportion of cancer tissue in a specified node t. This function is at its lowest level when $p_t = 0$ or 1. In other words, there is the least impurity when the node is perfect. On the other hand, it reaches the maximum when $p_t = \frac{1}{2}$, that is, the node is equally mixed with the cancer and normal tissues.

Let L and R denote the left and right nodes, respectively. The quality of the split s, resulting from the question "Is the expression level of gene 1 greater than 200?" is measured by weighing i_{L} and i_{R} as follows:

$$g_s = 1 - P(\mathrm{L})i_{\mathrm{L}} - P(\mathrm{R})i_{\mathrm{R}}, \tag{29.1}$$

where $P(\mathrm{L})$ and $P(\mathrm{R})$ are probabilities of tissues falling into the left and right nodes, respectively. The split with the lowest g_s is ultimately chosen to split the root node. This very same procedure can be applied to split the two daughter nodes, leading to the so-called recursive partitioning process. This process dies out as the sizes of the offspring nodes become smaller and smaller and the distribution of the tissue type becomes more and more homogeneous. The splitting stops when the node contains only one type of tissues.

The objective of the tree growing step is to produce a tree by executing the recursive partitioning process as far as possible. A natural concern is that such a saturated tree is generally too big and prone to noise. This calls for the second step to prune the saturated tree in order to obtain a reasonably sized tree that is still discriminative of the response whereas parsimonious for interpretation and robust with respect to the noise.

29.2.2 Tree Pruning and Cost Complexity

For the purpose of tree pruning, Breiman et al. (1984) introduced misclassification cost to penalize the errors of classification such as classifying a cancer tissue as a normal one, and vice versa. The unit of misclassification cost is chosen to reflect the seriousness of the errors because the consequence of classifying a cancer tissue as a normal one is usually more severe than classifying a normal tissue as a cancer one. A common practice is to assign a unit cost for classifying a normal tissue as a cancer one and a cost, c, for classifying a cancer tissue as a normal one. Once c is chosen, the class membership for any node can be determined to minimize the misclassification cost. For example, the root node of Fig. 29.1 is classified as a cancer node for any c chosen to be greater than $\frac{22}{40}$. While c is usually chosen to be greater than 1, for the purpose of illustration here, if it is chosen to be 0.5, the root node is classified as a normal node because it gives rise to a lower misclassification cost.

When the class memberships and misclassification costs are determined for all nodes, the misclassification cost for a tree can be computed easily by summing all costs in the terminal nodes. A node is terminal when it is not further divided, and other nodes are referred to as internal nodes. Precisely, the quality of a tree, denoted by T, is reflected by the quality of its terminal nodes as follows:

$$R(T) = \sum_{t \in \tilde{T}} P(t)R(t), \tag{29.2}$$

where $\tilde{T}$ is the set of terminal nodes of tree T and $R(t)$ the within-node misclassification cost of node t.

The ultimate objective of tree pruning is to select a subtree of the saturated tree so that the misclassification cost of the selected subtree is the lowest on an independent, identically distributed sample, called a test sample. In practice, we rarely have a test sample. Breiman et al. (1984) proposed to use cross validation based on cost-complexity. They defined the number of the terminal nodes of T, denoted by $|\tilde{T}|$, as the complexity of T. A penalizing cost, the so-called complexity parameter, is assigned to one unit increase in complexity, i.e., one extra terminal node. The sum of all costs becomes the penalty for the tree complexity, and the cost-complexity of a tree is:

$$R_\alpha(T) = R(T) + \alpha|\tilde{T}|, \tag{29.3}$$

where $\alpha(> 0)$ is the complexity parameter.

A useful and interesting result from Breiman et al. (1984) is that, for a given complexity parameter, there is a unique smallest subtree of the saturated tree that minimizes the cost-complexity measure (29.3). Furthermore, if $\alpha_1 > \alpha_2$ the optimally pruned subtree corresponding to α_1 is a subtree of the one corresponding to α_2. Therefore, increasing the complexity parameter produces a finite sequence of nested optimally pruned subtrees, which makes the selection of the desirably-sized subtree feasible.

Although the introduction of misclassification cost and cost complexity provides a solution to tree pruning, it is usually a subjective and difficult decision to choose the misclassification costs for different errors. Moreover, the final tree can be heavily dependent on such a subjective choice. From a methodological point of view, generalizing the concept of misclassification cost is difficult when we have to deal with more complicated responses, which I will discuss in detail later. For these reasons, I prefer a simpler way for pruning as described by Segal (1988) and Zhang and Singer (2010).

Let us now return to the example. In Fig. 29.1, the 62 tissues are divided into four terminal nodes 2, 5, 6, and 7. Two of them (Nodes 2 and 7) contain 21 normal tissues and no cancer tissue. The other two nodes (Node 5 and 6) contain 40 cancer tissues and 1 normal tissue. Because this tree is relatively small and has nearly perfect classification, pruning is almost unnecessary. Interestingly, this is not accidental for analyses of many microarray data for which there are many genes and relatively few samples.

The construction of Fig. 29.1 follows the growing procedure as described above. First, node 1 is split into nodes 2 and 3 after examining all allowable splits from the 2000 gene expression profiles, and the expression level of gene IL-8 and its threshold at 60 are chosen because they result in the lowest weighted impurity of nodes 2 and 3. A tissue is sent to the left (node 2) or right (node 3) daughter node according to whether or not the IL-8 level is below 60. Because node 2 is pure, no further split is necessary and it becomes a terminal node. Node 3 is split into nodes 4 and 5 through recursive partitioning and according to whether or not the expression of gene CANX is greater than 290, while the partition is restricted to the 40 tissues in node 3 only. Furthermore, node 4 is subsequently partitioned into nodes 6 and 7 according to whether or not the expression of gene RAB3B exceeds 770.

There are also many interesting applications of simple classification trees. For example, Goldman et al. (1982) used classification trees to predict heart attacks based on information from 482 patients. After a tree is constructed, the prediction is made from a series of questions such as "Is the pain in the neck only?" and/or "Is the pain in the neck and shoulder?" An appealing feature of tree-based classification is that the classification rule is based on the answers to simple and intuitive questions as posed here.

Although I present classification trees for a binary response, the method is similar for a mult-level response. The impurity function can be defined as

$$i_t = -\sum_{j=1}^{J} P(y = j) \log\{P(y = j)\},$$

for a J-level y. Everything else in the tree growing step as described above is applicable. For tree pruning, the only change to be made is to define the misclassification cost $c(j|k)$ from level k to level j, $j, k = 1, \ldots, J$.

29.3 Computational Issues

In Sects. 29.2.1 and 29.2.2, I have explained the basic steps and concepts for tree construction. For most users of decision trees, the implementation aspect does not really affect the application. For methodological and software developments, however, it is imperative to understand the computational issues. The most critical issue is to find the optimal split efficiently for any given node. The overall strategy is to identify the optimal split from each of the predictors and then choose the overall best one. Choosing the overall best one is straightforward, but identifying the optimal split from a predictor takes some efforts. The algorithm must take into account the nature of the predictor. Although I will use a dichotomous response to explain the ideas, the algorithm is also applicable for the other types of responses.

29.3.1 Splits Based on An Ordinal Predictor

Let us first consider a predictor with an ordinal scale such as gene expression in Fig. 29.1 or the ratio of cash flow to total debt in Fig. 29.2. Under the tree framework,

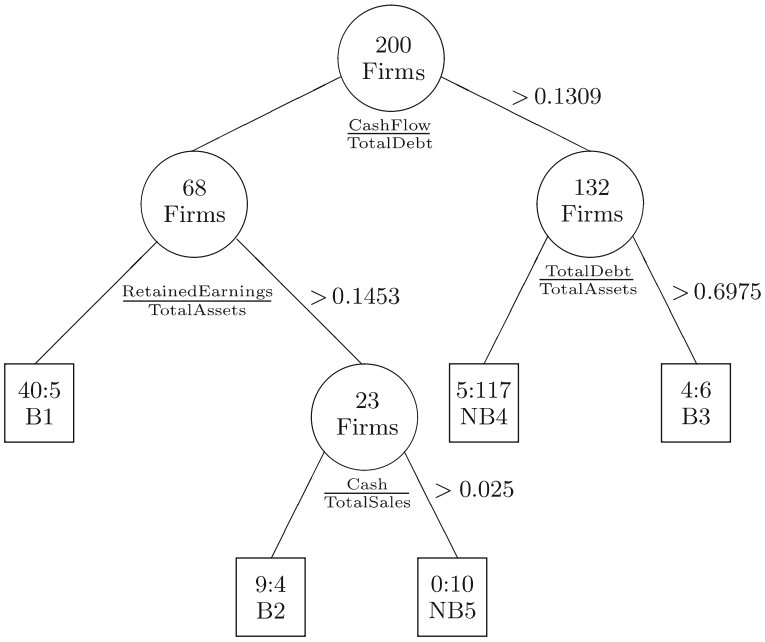


Fig. 29.2 Classification tree for bankruptcy. B1, B2, and B3 are three groups of relatively high risk of bankruptcy, and NB1 and NB2 are two groups of likely non-bankrupt companies. Inside the terminal nodes (boxes) are the numbers of bankrupt and non-bankrupt companies. See Frydman et al. (2002) for more details

Table 29.1 Expression level of gene IL-8 in 22 normal and 40 colon cancer tissues used in Fig. 29.1

Expression Level	Colon Cancer	Expression Level	Colon Cancer	Expression Level	Colon Cancer	Expression Level	Colon Cancer
23.74	N	35.95875	N	33.9725	N	45.1	N
56.91875	N	28.7675	N	28.00875	N	39.7575	N
11.37625	N	31.6975	N	30.57875	N	171.4525	N
36.8675	N	40.33875	N	76.9875	N	97.92	N
55.2	N	238.58625	N	645.99375	N	117.6025	N
113.91375	N	567.13125	N	1528.4062	Y	306.30875	Y
76.125	Y	169.1375	Y	213.6275	Y	326.42625	Y
370.04	Y	114.92375	Y	311.4375	Y	186.2775	Y
131.65875	Y	412.135	Y	284.14625	Y	1178.9188	Y
75.81375	Y	1007.5262	Y	120.72	Y	227.70625	Y
80.73875	Y	2076.9025	Y	93.3575	Y	1813.4562	Y
170.11875	Y	737.695	Y	270.19625	Y	75.95	Y
62.7375	Y	148.04125	Y	599.6975	Y	247.52625	Y
390.31125	Y	222.55875	Y	391.355	Y	249.15125	Y
117.185	Y	104.78125	Y	124.91875	Y	210.90125	Y
519.08125	Y	175.55125	Y				

as long as a predictor is ordinal, we will soon see that it does not matter whether the predictor is on a continuous or discrete scale.

Table 29.1 displays the expression levels of gene IL-8 in 22 normal and 40 colon cancer tissues. Our objective for the time being is to split these 62 tissues into two subsamples according to whether the expression level of gene IL-8 is greater than a given threshold. In theory, this threshold can be anything, but practically, there is only a finite number of them that make a difference. In other words, it takes a finite number of steps to find an optimal threshold, although the solution is not unique.

The first step in finding an optimal threshold is to sort all expression levels, say, in an ascending order as displayed in Table 29.2. If the threshold is below the minimum (11.37625) or above the maximum (2076.9025), it produces an empty subsample. Thus, the threshold should be between 11.37625 and 2076.9025. If we take a look at the two lowest levels, 11.37625 and 23.74, it is clear that any threshold between these two levels produces the same two subsamples (or daughter nodes). In this example, there are 62 distinct levels of expression. Thus, we have $62 - 1 = 61$ distinct ways to split the 62 samples into two daughter nodes. It is noteworthy that, unlike this example, the number of unique levels of a predictor is usually lower than the number of samples.

The second step in finding an optimal threshold is to move along the intervals defined by two adjacent, distinct levels of the sorted predictor values. In Table 29.2, we move along as follows:

$$[11.37625, 23.74), \ [23.74, 28.00875), \ldots, [56.91875, 62.7375),$$
$$\ldots, [1528.4062, 1813.4562), [1813.4562, 2076.9025).$$

Table 29.2 Sorted expression level of gene IL-8 in 22 normal and 40 colon cancer tissues used in Fig. 29.1

Colon Level	Expression Cancer	Colon Level	Expression Cancer	Colon Level	Expression Cancer	Colon Level	Cancer
11.37625	N	23.74	N	28.00875	N	28.7675	N
30.57875	N	31.6975	N	33.9725	N	35.95875	N
36.8675	N	39.7575	N	40.33875	N	45.1	N
55.2	N	56.91875	N	62.7375	Y	75.81375	Y
75.95	Y	76.125	Y	76.9875	N	80.73875	Y
93.3575	Y	97.92	N	104.78125	Y	113.91375	N
114.92375	Y	117.185	Y	117.6025	N	120.72	Y
124.91875	Y	131.65875	Y	148.04125	Y	169.1375	Y
170.11875	Y	171.4525	N	175.55125	Y	186.2775	Y
210.90125	Y	213.6275	Y	222.55875	Y	227.70625	Y
238.58625	N	247.52625	Y	249.15125	Y	270.19625	Y
284.14625	Y	645.99375	N	306.30875	Y	311.4375	Y
326.42625	Y	370.04	Y	390.31125	Y	391.355	Y
412.135	Y	519.08125	Y	567.13125	N	599.6975	Y
737.695	Y	1007.5262	Y	1178.9188	Y	1528.4062	Y
1813.4562	Y	2076.9025	Y				

Table 29.3 Search for the optimal split

	Left node			Right node			Split
Interval	No. of Sample	No. of Cancer	Node Impurity	No. of Sample	No. of Cancer	Node Impurity	Quality g_s
[11.37625, 23.74)	1	0	0	61	40	0.6438	0.3666
[23.74, 28.00875)	2	0	0	60	40	0.6365	0.3849
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
[56.91875, 62.7375)	14	0	0	48	40	0.4506	0.6512
[62.7375, 75.81375)	15	1	0.1030	47	39	0.4562	0.6292
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
[1528.4062, 1813.4562)	60	38	0.6572	2	2	0	0.3640
[1813.4562, 2076.9025)	61	39	0.6538	1	1	0	0.3568

For computation, the threshold can be chosen as the middle point of the above intervals. For interpretation, the threshold can be rounded-off as is done to the first split in Fig. 29.1.

We have determined the pool of the potential thresholds, which is sometimes referred to as the allowable splits. Obviously, we can examine each threshold one at a time and assess its quality according to (29.1).

For a large data set, this means a lot of wasted computing time. To reduce the computation to a minimal level, let us take a careful look as to what happens when we move the threshold from one interval to the next. In Table 29.3, as the threshold is moved up to the next interval, the samples that were already assigned to the left daughter stay on the left side because their expression levels are still below the new

threshold. Most of the samples that were assigned to the right daughter stay on the right side, except those samples whose expression levels are equal to the lower limit of the new interval. In this particular case, there is only one sample that we need to move from the right side to the left every time we move the threshold by one interval. This observation implies that the node impurities and the split quality can be computed by updating the information slightly for the small set of the samples that are affected. Each sample in this small set is affected only once in the entire search of the predictor. In summary, after the values of a predictor are sorted, we can find an optimal threshold to split a node in the number of steps proportional to the number of distinct values of the predictor, which is at most the number of samples in the node. For the present example, any value in $[56.91875, 62.7375)$ is an optimal split. Intuitively from Table 29.3, we push the threshold as high as possible to maintain the perfect purity of the left daughter node. In the meantime, if we look bottom-up from the table, we also push the threshold as low as possible to maximize the purity of the right daughter node. The interval $[56.91875, 62.7375)$ offers the best balance. In Fig. 29.1, the split is chosen at 60, although any number in this interval is a legitimate choice.

Overall, if we have n samples in a node and p predictors, excluding the sorting time, the final threshold for the node can be identified in at most $O(np)$ steps.

29.3.2 Splits Based on A Nominal Predictor

For a nominal variable, we cannot sort the values of the variable as we did in Table 29.2. For a predictor of k levels, there are a total of $2^{k-1} - 1$ ways to split a node. To explain the algorithm, let me use an artificial example as summarized in Table 29.4.

In Table 29.4, the predictor has 4 levels, giving rise to 7 possible ways to split a node. A naive way is to assess every allowable split on an individual basis. This could be an extensive computation when the number of levels is 10 or higher. Thus, it is important to find a way to compute the quality of all splits in a gradual manner as in Sect. 29.3.1. If we focus on the levels of the predictor for the left daughter node, we can travel all 7 possible splits as follows: $\{A\}$, $\{AB\}$, $\{B\}$, $\{BC\}$, $\{C\}$, $\{AC\}$, and $\{ABC\}$. The key is that every move requires either the deletion or addition of a single level, which keeps the computation at the minimal level. Such a path of traveling through all $2^{k-1} - 1$ splits can be defined for any k.

Table 29.4 An artificial data set

Predictor Value	No. of Normal	No. of Cancer	Rate of Cancer
A	5	10	0.67
B	10	5	0.33
C	20	30	0.60
D	35	25	0.42

There is actually a simple and quick solution for a dichotomous response. As shown in Table 29.4, we can compute the cancer rate for every level of the nominal predictor. During the splitting, the rates can substitute for the corresponding nominal levels. Because the rates are ordinal, the method described in Sect. 29.3.1 can be applied. After the optimal split is determined, we can map the rate back to the original nominal level. For example, for the data in Table 29.4, the optimal threshold based on the rate is in the interval $[0.42, 0.6)$, which means that the left daughter node contains samples with levels B and D, and the right daughter node with levels A and C. For a multiclass response, there is no apparent way to form an ordinal surrogate for a nominal predictor.

29.3.3 Missing Values

An important feature of decision trees is their ability to deal with missing predictor values. There are several solutions. Although there have been limited attempts Quinlan (1989) to compare some of them, the performance of the various solutions is largely unexplored. The choice mostly depends on the objective of the study.

The easiest approach is to treat the missing attribute as a distinct value and to assign all samples with missing values to the same node Zhang et al. (1996). This approach is not only simple, but also provides a clear path as to where the samples with missing attributes end up in the tree structure.

Breiman et al. (1984) introduced and advocated surrogate splits to deal with missing attributes. The idea is very intuitive. For example, in Table 29.2, we considered using expression levels from gene IL-8 to split the 62 samples. What happens if the expression level from one of the samples, say, the first one, was not recorded? This happens in microarray experiments. Because IL-8 level is missing for the first sample, we cannot determine whether the level is below or above 60 and hence cannot decide whether the first sample should be assigned to the left or right daughter node. To resolve this ambiguity, Breiman et al. (1984) proposed to seek help from other genes that act "similarly" to IL-8. Since there are many other genes, we can use the one that is most similar to IL-8, which leads to a surrogate for IL-8.

What we need to clarify is the meaning of similarity. To illustrate this concept, let us consider gene CANX. Using the method described in Sect. 29.3.1, we can find an optimal split from gene CANX. The similarity between CANX and IL-8 is the probability that the optimal splits from these two genes assign a sample with complete information in these two genes into the same node. This strategy is similar to replacing a missing value in one variable in linear regression by regressing on the non-missing value most highly correlated with it. Then, why can't we use the same strategy as in the linear regression? According to Breiman et al. (1984), their strategy is more robust. The main reason is that their strategy is more specific to the particular sample with missing attributes, and does not result in a potential catastrophic impact for other samples with missing attributes.

The surrogate splits have some advantages over the simpler approach as described earlier. It makes use of other potentially useful information. Breiman et al. (1984) also proposed to rank the importance of variables through surrogate splits. The surrogate splits also have some limitations. First, it is uncommon, if at all, that surrogate splits are provided in published applications. Thus, it is unrealistic to know what the surrogate splits are and how we assign a sample with a missing attribute. Second, there is no guarantee in a data set that we can find a satisfactory surrogate split. Lastly, while it is a sensible idea to rank the variable importance based on surrogate splits, there is no assurance that a predictor ranked relatively high is necessarily predictive of the outcome, which can create a dilemma for interpretation. More recently, the importance of a variable tends to be evaluated on the basis of its performance in forests Breiman (1994), Zhang et al. (2003) rather than on a single tree.

In the construction of random forests, Breiman proposed another way of replacing missing values through an iterative process. A similar idea can be applied for tree construction. To initialize the process, we can fill in the missing values by the median of an ordered variable or by the category of a nominal variable with the highest frequency. An initial tree can be constructed once all missing data are imputed. In the next step, suppose again that in Table 29.2, the expression of gene IL-8 is missing for the first sample. The unobserved level is estimated by a weighted average over the samples with observed expressions for gene IL-8. Here, the weight is the so-called proximity, which is a similarity measure between a pair of samples. Intuitively, if the second sample is more similar to the first sample than to the third one, we give more weight to the second sample than to the third one if the first sample is not observed. How is the proximity defined for a pair of samples? We can set it to zero before the initial tree is grown. Then, whenever a tree is grown, if two samples end up in the same terminal nodes, its promixity is increased by one unit. After the missing data are updated, a new tree is grown. Breiman recommends to continue this process at most five times in the random forest construction. For tree construction, it may take longer for the process to "converge," especially when the number of predictors is large. Nonetheless, it may still be worthwhile to repeat a few iterations. In addition to this convergence issue, it is also difficult to track where the samples with missing values are assigned as with the use of surrogate splits.

29.4 Interpretation

Interpretation of results from trees is usually straightforward. In Fig. 29.1, we identified 3 genes IL-8, CANX, and RAB3B whose expression levels are highly predictive of colon cancer. However, this does not necessarily mean that these genes cause colon cancer. Such a conclusion requires a thorough search of the literature and further experiments. For example, after reviewing the literature, Zhang et al. (2001) found evidence that associates IL-8 with the stage of colon cancer Fox et al. (1998), the migration of human clonic epithelial cell lines Toshina et al. (2000),

and metastasis of bladder cancer Inoue et al. (2000). In addition, the expression of the molecular chaperone CANX was found to be down-regulated in HT-29 human colon adenocarcinoma cells Yeates and Powis (1997) and to be involved in apoptosis in human prostate epithelial tumor cells Nagata et al. (1997). Lastly, RAB3B is a member of the RAS oncogene family. Therefore, these existing studies provide independent support that the three genes identified in Fig. 29.1 may be in the pathways of colon cancer. If this hypothesis could be confirmed from further experiments, Fig. 29.1 would have another important implication. Pathologically speaking, the 40 colon cancer samples are indistinguishable. Fig. 29.1 indicates that those 40 samples are not homogeneous in terms of gene expression levels. If confirmed, such a finding could be useful in cancer diagnosis and treatment.

As I stated earlier, there are numerous applications of decision trees in biomedical research, including the example above. To have a glimpse of the diverse applications of decision trees, let me review two different examples.

Example 3. Frydman and colleagues introduced recursive partitioning for financial classification Frydman et al. (2002). They considered a financial dataset of 58 bankrupt ($y = 1$) industrial companies that failed during 1971-81, and 142 non-bankrupt ($y = 0$) manufacturing and retailing companies randomly selected from COMPUSTAT universe. Each company forms an observational unit or the so-called sample. Twenty financial variables with prior evidence of predicting business failure are considered. They include the ratio of cash to total assets, the ratio of cash to total sales, the ratio of cash flow to total debt, the ratio of current assets to current liabilities, the ratio of current assets to total assets, the ratio of current assets to total sales, the ratio of earnings before interest and taxes to total assets, interest coverage, the ratio of market value of equity to total capitalization, the ratio of net income to total assets, the ratio of quick assets to current liabilities, the ratio of quick assets to total assets, the ratio of quick assets to total sales, the ratio of retained earnings to total assets, the ratio of total debt to total assets, the ratio of total sales to total assets, and the ratio of working capital to total sales.

We can see from Fig. 29.2 that the risk of bankruptcy is relatively high if the ratio of cash flow to total debt is below 0.1309, unless both the ratio of retained earnings to total assets and the ratio of cash to total sales are above certain levels, i.e., 0.1453 and 0.025, respectively. Even if the ratio of cash flow to total debt is above 0.1309, there can be elevated risk of bankruptcy if the ratio of total debt to total assets is high (above 0.6975). A tree diagram as in Fig. 29.2 offers a very clear and simple assessment of the financial state of a company.

Example 2 (continued) I indicated earlier what the predictors and response are for Example 2. Let us revisit this example. Unlike the other examples that I have introduced so far, this example uses a continuous response y – the compound potency. Because of this difference, the resulting tree is called a regression tree. To utilize the information from the 3-dimensional structures of compounds, Chen et al. (1998) used atom pair descriptors that are composed of the atom types of the two component atoms and the "binned" Euclidean distance between these two atoms. The width of each distance bin was chosen as 1.0 Å. To define predictors $\mathbf{x}$ from the atom pair descriptors, the authors characterized the atom pair descriptors

in 17 types including negative charge center (e.g., sulfinic group), positive charge center (e.g., the nitrogen in primary, secondary, and tertiary amines), hydrogen bond acceptor (e.g., oxygen with at least one available lone pair electron), triple bond center, aromatic ring center, and H-bond donor hydrogen.

Figure 29.3 presents part of the regression tree that is constructed by Chen et al. (1998). I trimmed the left hand side to fit into the space here; however, we can get the idea from the right hand side of tree. Generally speaking, a node of size 3 or 6 such as nodes 6 and 8 is too small to be reliable. Since I do not have the data to re-grow the tree, let us pretend that the node sizes are adequate, and concentrate on the interpretation instead. Since the main objective of Chen et al. appears to identify active nodes (i.e., those with high potencies), a small, inactive node is not of great concern.

First, there is one highly active node (node 7 with potency greater than 2) in Fig. 29.3. There are also two highly active nodes on the left hand side which are not shown in Fig. 29.3. Supported by the literature, Chen et al. (1998) postulated that

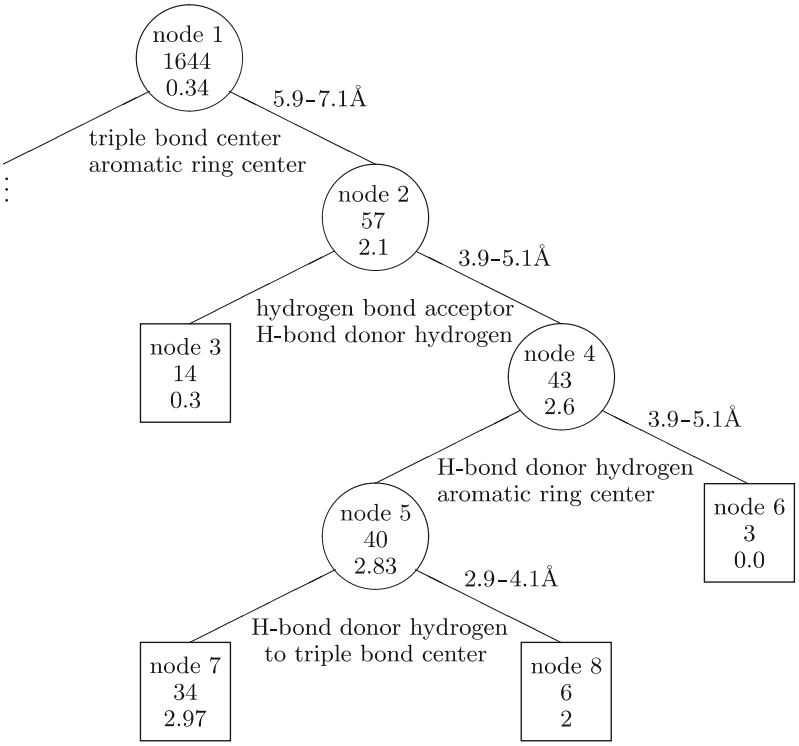


Fig. 29.3 Regression Tree for Predicting Potencies of Compounds. Inside each node are the number of compounds (middle) and the average potency of all compounds within the node (bottom). Underneath each node is the selected atom pair descriptor. Above the arm is the interval for the distance between the selected atom pair descriptor that assigns the compounds to the right daughter node. See Chen et al. (1998) for more details

there might be different mechanisms of action because the active nodes contain compounds of very different characteristics. This is similar to the hypothesis suggested by Fig. 29.1 that the 40 colon cancer tissues might be biologically heterogeneous. Chen et al. concluded further that their tree demonstrates the ability to detect multiple mechanisms of action coexisting in a large three-dimensional chemical data set. In addition, the selected atom pair descriptors also reveal interesting features of the monoamine oxidase (MAO) inhibitors. For instance, the "aromatic ring center–triple bond center" pair in the first split is the structural characteristic of pargyline, a well known MAO inhibitor.

We can see from these examples that tree-based methods tend to unravel integrated, intuitive results whose pieces are consistent with prior findings. Not only can we use trees for prediction, but also we may use them to identify potentially important mechanisms or pathways for further investigation.

29.5 Survival Trees

The most popular use of tree-based methods is arguably in survival analysis for censored time, particularly in biomedical applications. The general goal of such applications is to identify prognostic factors that are predictive of survival outcome and time to an event of interest. For example, Banerjee et al. (2000) reported a tree-based analysis that enables the natural identification of prognostic groups among patients in the perioperative phase, using information available regarding several clinicopathologic variables. Such groupings are important because patients treated with radical prostatectomy for clinically localized prostate carcinoma present considerable heterogeneity in terms of disease-free survival outcome, and the groupings allow physicians to make early yet prudent decisions regarding adjuvant combination therapies. See, e.g., Bacchetti and Segal (1995), Carmelli et al. (1991), Carmelli et al. (1997), Kwak et al. (1990) for additional examples.

Before pointing out the methodological challenge in extending the basic classification trees to survival trees, let me quickly introduce the censored data. Let z denote the time to an event, which can be death or the occurrence of a disease. For a variety of reasons including losts to follow-up and the limited period of a study, we may not be able to observe z until the event occurs for everyone in the study. Thus, what we actually observe is a censored time y which is smaller than or equal to z. When z is observed, $y = z$. Otherwise, z is censored and $y < z$. Let $\delta = 1$ or 0 denote whether z is censored or observed.

The question is how to facilitate the censored time y in the tree-based methods. As in Sect. 29.2, we need to define a splitting criterion to divide a node into two, and also to find a way to choose a "right-sized" tree. Many authors have proposed different methods to address these needs. Here, I describe some of the methods. See Crowley et al. (1995), Intrator and Kooperberg (1995), LeBlanc and Crowley (1995), Segal (1988), Segal (1995), Zhang et al. (2001), Zhang and Singer (2010) for more details.

29.5.1 Maximizing Difference between Nodes

Gordon and Olshen (1985) are among the earliest to have developed survival trees. Earlier, we focused on reducing the impurity within a node by splitting. When two daughter nodes have low impurities, the distributions of the response tend to differ between the two nodes. In other words, we could have achieved the same goal by maximizing the difference between the distributions of the response in the two daughter nodes. There are well established statistics that measure the difference in distribution. In survival analysis, we can compute the Kaplan-Meier curves (see, e.g., Miller 1981) separately for each node. Gordon and Olshen used the so-called L^p Wasserstein metrics, $d_p(\cdot, \cdot)$, as the measure of discrepancy between the two survival functions. Specifically, for $p = 1$, the Wasserstein distance, $d_1(S_L, S_R)$, between two Kaplan-Meier curves, S_L and S_R, is illustrated in Fig. 29.4.

A desirable split maximizes the distance, $d_1(S_L, S_R)$, where S_L and S_R are the Kaplan-Meier curves for the left and right daughter nodes, respectively. Replacing g_s in (29.1) with $-d_1(S_L, S_R)$ we can split the root node into two daughter nodes and use the same recursive partitioning process as before to produce a saturated tree.

To prune a saturated survival tree, T, Gordon and Olshen (1985) generalized the tree cost-complexity for censored data. The complexity remains the same as before, but we need to redefine the cost $R(t)$, which now is measured by how far node t deviates from a desirable node in lieu of a pure node in the binary response case. In the present situation, a replacement for a pure node is a node τ in which all observed times are the same, and hence its Kaplan-Meier curve, δ_τ, is a piecewise constant survival function that has at most one point of discontinuity. Then, the within-node cost, $R(t)$, is defined as $d_1(S_t, \delta_\tau)$. Combining this newly defined cost-complexity with the previously described pruning step serves as a method for pruning survival trees.

Another, perhaps more commonly used way to measure the difference in survival distributions is to make use of the log-rank statistic. Indeed, the procedures proposed by Ciampi et al. (1986) and Segal (1988) maximize the log-rank statistic by comparing the survival distributions between the two daughter nodes. The authors did not define the cost-complexity using the log-rank statistic. However, LeBlanc and Crowley (1993) introduced the notion of "goodness-of-split" complexity as a substitute for cost-complexity in pruning survival trees. Let $G(t)$ be the value of the

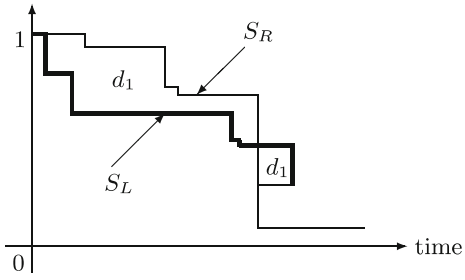


Fig. 29.4 The L^1 wasserstein distance between Two Kaplan-Meier curves as measured by the area marked with d_1. Note that one curve (S_L) is thicker than the other (S_R)

log-rank test at node t. Then the split-complexity measure is

$$G(T) = \sum_{t \notin \tilde{T}} G(t) - \alpha(|\tilde{T}| - 1).$$

Therneau et al. (1990) proposed another way to define $R(t)$ that makes use of the so-called martingale residuals by assuming within-node proportional hazard models and then the least squares are computed as the cost.

In my experience, I found that Segal (1988) bottom-up procedure is practical and easy to use. That is, for each internal node (including the root node) of a saturated tree, we assign it a value that equals the maximum of the log-rank statistics over all splits starting from the internal node of interest. Then, we plot the values for all internal nodes in an increasing order and decide a threshold from the graph. If an internal node corresponds to a smaller value than the threshold, we prune all of its offspring. Zhang and Singer (2010) pointed out that this practical procedure can be modified in a broad context by replacing the log-rank statistic with a test statistic that is appropriate for comparing two samples with a defined outcome.

29.5.2 Use of Likelihood Functions

Although the concept of node impurity is very useful in the development of tree-based methodology, that concept is closely related to the concept of likelihood as pointed out by Zhang et al. (2001). In fact, the adoption of likelihood makes it much easier to extend the tree-based methodology to analysis of complex dependent variables including censored time. For example, Davis and Anderson (1989) assume that the survival function within any given node is an exponential function with a constant hazard. LeBlanc and Crowley (1992) and Ciampi et al. (1988) assume different within-node hazard functions. Specifically, the hazard functions in two daughter nodes are assumed proportional, but are unknown. In terms of estimation, LeBlanc and Crowley (1992) use the full or partial likelihood function in the Cox proportional hazard model whereas Ciampi et al. (1988) use a partial likelihood function.

The most critical idea in using the likelihood is that within-node survival functions are temporarily assumed to serve as a vehicle of finding a split instead of believing them to be the true ones. For example, we cannot have a constant hazard function in the left daughter node, and then another constant hazard function in the right daughter node while assuming that the parent node also has a constant hazard function. Here, the constant hazard function plays the role of the "sample average." However, after a tree is constructed, it is both reasonable and possible that the hazard functions within the terminal nodes may become approximately constant.

29.5.3 A Straightforward Extension

Zhang (1995) examined a straightforward tree-based approach to censored survival data by observing the fact that the response variable involves two dimensions: a binary censoring indicator and the observed time. If we can split a node so that the node impurity is "minimized" in both dimensions, the within-node survival distribution is expected to be homogeneous. Based on this intuitive idea, Zhang (1995) proposed to compute the within-node impurity in terms of both the censoring indicator and the observed time first separately, and then together through weighting. Empirically, this simple approach tends to produce trees similar to those produced from using the log-rank test. More interestingly, empirical evidence also suggests that this simple approach outperforms its more sophisticated counterparts in discovering the underlying structures of data. Unfortunately, there need to be more comparative studies to scrutinize these different methods, even though limited simulations comparing some of the methods have been reported in the literature Crowley et al. (1995), Crowley et al. (1997), Zhang (1995).

29.5.4 Other Developments

The methods that I described above are not designed to deal with time-dependent covariates. Bacchetti and Segal (1995) and Huang et al. (1998) proposed similar approaches to accommodate the time-dependent covariates in survival trees. The main concern with these existing approaches is that the same subject can be assigned to both the left and right daughter nodes, which is distinct from any other tree-based methods and is potentially confusing in interpretation.

It is common in survival tree analysis that we want to stratify our sample into a few groups that define the grades for the survival. To this end, it is useful to combine some terminal nodes into one group, which is loosely called "amalgamation." Ciampi et al. (1986) used the log-rank statistic for this purpose. LeBlanc and Crowley (1993) proposed constructing an ordinal variable that describes the terminal nodes. Often, we can simply examine the Kaplan-Meier curves for all terminal nodes to determine the group membership Carmelli et al. (1997).

29.6 Tree-Based Methods for Multiple Correlated Outcomes

As pointed out by Zhang (1998), multiple binary responses arise from many applications for which an array of health-related symptoms are of primary interest. Most of the existing methods are parametric; see, e.g., Diggle et al. (1994) for an excellent overview. In this section, I will describe a tree-based alternative to the parametric methods.

Motivated by both the broad application as well as by the need to analyze building-related occupant complaint syndrome (BROCS), I proposed a tree-based method to model and classify multiple binary responses Zhang (1998). Let me use the BROCS study to explain the method.

To understand the nature of BROCS, data were collected in 1989 from 6,800 employees of the Library of Congress (LOC) and the headquarters of the Environmental Protection Agency (EPA) in the United States. The data contain many explanatory variables, but I extracted a subset of 22 putative risk factors, most of which are answers to "yes or no" or frequency (never, rarely, sometimes, etc.) questions. For example, is working space an enclosed office with door, a cubicle without door, stacks, etc? See Table 1 of Zhang (1998) for a detailed list. In this data set, BROCS is represented by six binary responses that cover respiratory symptoms in the central nervous system, upper airway, pain, flu-like symptoms, eyes, and lower airway. The primary purpose with this extracted data set is to evaluate the effect of the important risk factors on the six responses by constructing trees.

In terms of notation, the primary distinction is that the response y for each subject is a 6-vector. Consequently, we need to generalize the node-splitting criterion and cost-complexity to this vector-response. As I indicated earlier, one solution is to assume a certain type of within-node distribution for the vector-response and then maximize the within-node likelihood for splitting. One such distribution is

$$f(y; \Psi, \boldsymbol{\theta}) = \exp(\Psi' y + \boldsymbol{\theta}' w - A(\Psi, \boldsymbol{\theta})), \tag{29.4}$$

where Ψ and $\boldsymbol{\theta}$ are node-dependent parameters, $A(\Psi, \boldsymbol{\theta})$ is the normalization function depending on Ψ and $\boldsymbol{\theta}$, and $w = \sum_{i<j} y_i y_j$. I chose this distribution because it is commonly used in the parametric models for multiple binary responses, e.g., Cox (1972), Fitzmaurice and Laird (1993), Zhao and Prentice (1990). The negative of the likelihood based on (29.4) now serves as the impurity function, and the rest of the recursive partitioning as described before applies.

A naive approach is to treat y as a numerical vector and use a function such as the determinant of the within-node covariance matrix of y as a measure of impurity. If y were continuous, this approach is what Segal (1992) proposed to construct regression trees for repeatedly measured continuous y. For binary outcomes, however, this approach appears to suffer the well-known end-cut preference problem in the sense that it gives preference to the splits that result in two unbalanced daughter nodes in terms of their sizes.

One advantage of the likelihood based method is that the negative of the within-node likelihood can also be used as the within-node cost $R(t)$ for tree pruning. The main difficulty with this method is the computational burden, because every allowable split calls for a maximization of the likelihood derived from (29.4). Some strategies for reducing the computational time are discussed in Zhang (1998).

The criterion based on (29.4) ultimately leads to a 9 terminal nodes tree as displayed in Fig. 29.5, which suggests that respondents belonging to terminal nodes 7 and 17 have high incidence of respiratory symptoms. This is because the working

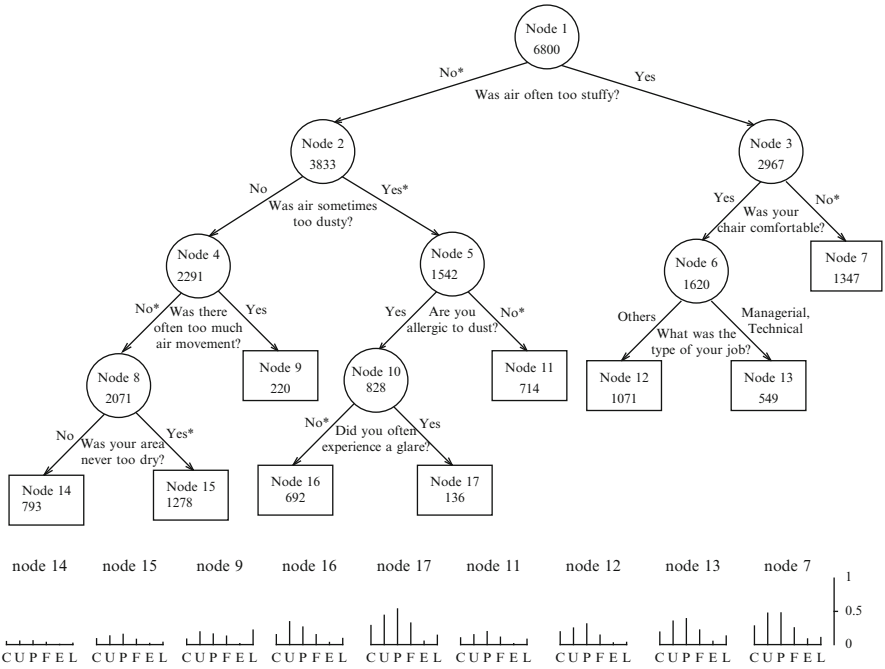


Fig. 29.5 Tree Structure for the risk factors of BROCS based on (29.4). Inside each node (a circle or a box) are the node number and the number of subjects. The splitting question is given under the node. The asterisks indicate where the subjects with missing information are assigned. The pin diagrams under the tree show the incidence rates of the six clusters (C: CNS; U: upper airway; P: pain; F: flu-like; E: eyes; and L: lower airway) in the nine terminal nodes. The side bar on the right end indicates the range of 0 and 1 for the rates of all symptoms

area air quality of the people within these terminal nodes was poor, namely, often too stuffy or sometimes dusty. On the other hand, for example, subjects in terminal node 14 experienced the least discomfort because they had the best air quality. The basic message from this example is that tree-based analyses often reveal findings that are readily interpretable.

For ordinal responses, we describe the method proposed by Zhang and Ye (2008). Let z_{ij} be the jth ordinal response in the ith subject, taking a value from $1, \cdots, K$. Note here that K is the same for all response variables, although in principle we can create extra levels with zero frequency to accommodate different K's. We define $K-1$ indicator variables $y_{ijk} = I(z_{ij} > k)$, for $k = 1, \cdots, K-1$. Recall $I(\cdot)$ is the indicator function. Let

$$\begin{aligned} y_{ij} &= (y_{ij,1}, \cdots, y_{ij,K-1})', \\ \mathbf{y}_i &= (y'_{i1}, \cdots, y'_{in})', \end{aligned} \tag{29.5}$$

Then, the observed responses from the ith unit can be rewritten as

$$\mathbf{y}_i = (y_{i1,1}, \cdots, y_{i1,K-1}, \cdots, y_{in,1}, \cdots, y_{in,K-1})'.$$

Now, the components of the $\mathbf{y}_i$ are binary, and hence we can use the same procedure described above.

29.7 Tree for Treatment Effectiveness

Although this chapter is mainly concerned with classifying a defined response, the method can be modified for other important applications. For example, in a typical randomized clinical trial, different treatments (say two treatments) are compared in a study population, and the effectiveness of the treatments is assessed by averaging the effects over the treatment arms. However, it is possible that the on-average inferior treatment is superior in some of the patients. Because each patient is generally assigned to one treatment, it is possible to define a response variable for the treatment effectiveness for the single patient, and we do not know what would happen if the patient receives the alternative medication. The trees provide a useful framework to explore this possibility by identifying patient groups within which the treatment effectiveness varies the greatest among the treatment arms. Even though we cannot directly use the impurity function as defined in Sect. 29.2.1, we can replace it with the Kullback-Leibler divergence Kullback and Leibler (1951). To do so, let $p_{y,i}(t) = P(Y = y|t, \text{Trt} = i)$ be the probability that the response is y when a patient in node t received the i-th treatment. Then, the Kullback-Leibler divergence within node t is $\sum_y p_{y,1} \log(p_{y,1}/p_{y,2})$. Note that the Kullback-Leibler divergence is not symmetric with respect to the role of $p_{y,1}$ and $p_{y,2}$, but it is easy to symmetrize it as follows:

$$D_{KL}(t) = \sum_y p_{y,1} \log(p_{y,1}/p_{y,2}) + \sum_y p_{y,2} \log(p_{y,2}/p_{y,1}).$$

A simpler and more direct measure is the difference

$$DIFF(t) = \sum_y (p_{y,1} - p_{y,2})^2.$$

It is noteworthy that neither D_{KL} nor $DIFF$ is a distance metric and hence does not possess the property of triangle inequality. Consequently, the result does not necessarily improve as we split a parent node into offspring nodes. However, this does not prevent us from splitting a node or building a tree.

We should point out that the current goal is not to find a pure node, but to find a node within which the treatments have different effects. In other words, we want to identify groups so that the clinicians may have the easiest time to make a clinical

recommendation. Once the splitting criterion is chosen, the rest of the recursive partitioning can proceed similarly. We applied this method to a clinical trial on ovulation in a study of women with polycystic ovary syndrome, and the results of analysis shall be reported elsewhere.

For pruning, we could also incorporate clinical information. For example, we can merge any pair of terminal offspring nodes if the same treatment is preferred in both of them, because the splitting does not change the clinical decision and hence is uncalled for.

Later, we will introduce survival trees where the outcome variable is censored. A similar issue of treatment effectiveness in terms of survival time may arise from clinical trials. Again, one solution is to define a splitting criterion that compares the survivorship in different treatment arms such as by examining the ratio of the hazard ratios.

29.8 Forests

It should be clear now that tree-based data analyses are intuitive to interpret, but there are also caveats. As discussed by Zhang and Singer (2010), first, tree structure is prone to instability even with minor data perturbations. Second, as illustrated in Fig. 29.1, we have data sets that include hundreds of thousands of variables. We have to broaden the classic statistical view of "one parsimonious model" for a given data set. Third, due to the adaptive nature of the tree construction, theoretical inference based on a tree is usually not feasible. Generating more trees may provide an empirical solution to statistical inference Zhang (1998). These lead to the idea of forest.

A forest is a constellation of any number of tree models Breiman (1996), Breiman (2001). In computer science, a forest is an *ensemble*. While each individual tree is not a good model, combining them into a committee improves the overall performance. Having many trees also maximize the utilization of the information in the data set, potentially unraveling alternative pathways to disease etiologies.

29.8.1 Random Forest

There are different ways of constructing a forest. Here is how a random forest is formed.

1 Draw a bootstrap sample from the original data set.
2 Use the bootstrap to grow a tree. At each node, randomly select a fixed (much smaller) number of the predictors and use this subset of the variables to split the node.
3 Let the recursive partition run to the end and generate a tree.
4 Repeat Steps 1 to 3 to form a forest.

The forest-based classification is made by the majority vote from all trees. If Step 2 is skipped, the above algorithm is called *bagging* (bootstraping and aggregating) Breiman (1994).

To form and use a forest, we need to be aware of several issues. First, how many variables should we select when splitting a node? The common recommendations include the logarithm or square root of the original number of predictors. This could be problematic when the number of the original predictors is huge or there are a particularly large number of some types of variables (such as genetic markers) Zhang and Singer (2010). Second, how many trees do we need in a forest? In general, a forest contains hundreds or thousands of trees Breiman (2001). In a recent study, we found that a few representive trees may be sufficient to maintain the performance of a forest Zhang and Wang (2009). Finally, how do we identify important variables in a forest? Breiman (2001) introduced the concept of variable importance.

The most cited importance index is the permutation importance or often referred to as the variable importance. For each tree in the forest, we count the number of votes cast for the correct class. Then, we randomly permute the values of variable k in the out-of-the-bag (oob) cases and recount the number of votes cast for the correct class in the oob cases with the permuted values of variable k. The permutation importance of the average of the differences between the number of votes for the correct class in the variable-k-permuted oob data from the number of votes for the correct class in the original oob data, over all trees in the forest.

Despite its popularity, this permutation importance index has some undesirable properties. First, it is not necessarily positive, and does not have an upper limit. Secondly, both the magnitudes and relative rankings of the permutation importance for predictors are not stable if the number of predictors is much greater than the sample size. They are also sensitive to the number of the subset variables chosen for node splitting and to the level of correlation among the predictors Genuer et al. (2008), Wang et al. (2010). To overcome some of these issues, Wang et al. (2010) recently introduced a maximal conditional chi-square (MCC) importance by taking the maximum chi-square statistic resulting from all splits in the forest that use the same predictor. Chen et al. (2007) introduced an even simpler measure called the depth importance by considering the location of the splitting variable as well as its impact. Specifically, whenever node t is split based on variable k, let $L(t)$ be the depth of the node and $S(k,t)$ be the χ^2 test statistic from the variable, then $2^{-L(t)}S(k,t)$ is added up for variable k over all trees in the forest. Here, the depth is 1 for the root node, 2 for the offspring of the root node, so on and so forth.

29.8.2 Random Forests for Uncertain Predictors

In other sections of this chapter, we assume that predictors are observed with certainty. However, what can we do if this is not the case. Chen et al. (2007) considered

an important application. Specifically, to identify genetic variants for complex diseases, haplotypes are considered for association analysis. In genomewide association studies, a haplotype is a set of single nucleotide polymorphisms (SNPs) on a chromatid. In general, haplotypes must be statistically inferred from the SNPs Lin et al. (2002). As a result, haplotypes are only available in frequencies. The idea in Chen et al. (2007) is to generate data according to the distribution of the haplotypes, which replaces Step 1 in the random forest algorithm. Then, they proceed with the rest of the steps, except that they utilized all predictors in Step 2, although one can certainly consider the use of a subset.

29.8.3 Deterministic Forests

The main reason that Wang et al. (2010) was able to find a small number of trees to represent a large random forest is that there tend to be trees with comparable structures that have similar classification performance when the number of features is large relative to the number of samples. This observation motivated Zhang et al. (2003) to assemble a forest with trees of similar structures and similar performance. They called such a forest a deterministic forest. The advantage of this forest construction is that it is a deterministic process, and hence reproducible. Through numerical examples, they found that the predictive performance of the deterministic forest and random forest is comparable.

29.8.4 Survival Forests

It is noteworthy that after we construct a survival tree using any of the methods described above, we can use the same method described in this section to construct a random survival forest. See, for example, Ishwaran et al. (2008).

29.9 Remarks

In Breiman et al. (1984), tree-based methods are presented primarily as an automated machine learning technique. There is now growing interest in applying tree-based methods in biomedical applications, partly due to the rising challenges in analyzing genomic data in which we have a large number of predictors and a far smaller number of observations Zhang et al. (2001). In biomedical applications, scientific understanding and interpretation of a fitted model are an integral part of the learning process. In most situations, an automated tree as a whole has characteristics that are difficult or awkward to interpret. Thus, the most effective and productive way of conducting tree-based analyses is to transform this machine

learning technique into a human learning technology. This requires the users to review the computer-generated trees carefully and revise the trees using their knowledge, which not only often simplifies the trees, but also may improve the predictive precision of the trees, because recursive partitioning is not a forward looking process and does not guarantee any optimality of the overall tree. Zhang et al. (1996) called this step tree repairing. Also, by expanding a tree to a forest, we are no longer short-sighted by the forward stepwise partitioning algorithm.

While the full potential of tree-based applications remains to be seen and exploited, it must be made crystal clear that parametric methods such as logistic regression and Cox models will continue to be useful statistical tools. We will see more applications that use tree-based methods together with parametric methods to take advantages of various types of methods. The main advantage of tree-based methods is their flexibility and intuitive structures. However, because of their adaptive nature, statistical inference based on tree-based methodology is generally difficult. Despite the difficulty, some progress has been made to understand the asymptotic behavior of tree-based inference Breiman (1994), Buhlmann and Yu (2003), Donoho (1997), Gordon and Olshen (1978), Gordon and Olshen (1980), Gordon and Olshen (1984), Lugosi and Nobel (1996), Nobel (1996), Nobel and Olshen (1996).

Some attempts have been made to compare the tree-structured methods with other methods Long et al. (1993), Segal and Bloch (1989), Selker et al. (1995). More comparisons are still warranted, particularly in the context of genomic applications where data reduction is necessary and statistical inference is also desirable.

One exciting development in recent years is the expansion of trees into forests. In a typical application such as Banerjee et al. (2000), Carmelli et al. (1997), constructing one or several trees is usually sufficient to unravel relationships between predictors and a response. Nowadays, many studies produce massive information such as recognizing spam mail from numerous characteristics and identifying disease genes. One or even several trees are no longer adequate to convey all of the critical information in the data. Construction of forests enables us to discover data structures further and in the meantime improves classification and predictive precision Breiman (1994), Zhang et al. (2003). The emergence of genomic and proteomic data afford us the opportunity to construct deterministic forest Zhang et al. (2003) by collecting a series of trees that have a similarly high predictive quality. Not only do forests reveal more information from large data sets, but they also outperform single trees Breiman (1994), Buhlmann and Yu (2003) Buhlmann and Yu (2002), Zhang et al. (2003).

A by-product of forests is a collection of variables that are frequently used in the forests, and the frequent uses are indicative of the importance of those variables. Zhang et al. (2003) examined the frequencies of the variables in a forest and used them to rank the variables. In this chapter, we described several measures of variable importance.

Bayesian approaches may offer another way to construct forests by including trees with a certain level of posterior probability. These approaches may also help us

understand the theoretical properties of tree-based methods. However, the existing Bayesian tree framework focuses on providing an alternative method to those that exist. We would make important progress if we could take full advantage of the Bayesian approach to improve our tree-based inference.

Classification and regression trees assign a subject to a particular node following a series of boolean statements. Ciampi et al. (2002) considered a "soft" splitting algorithm that at each node an individual goes to the right daughter node with a certain probability, which is a function of a predictor. This approach has the spirit of random forests. In fact, we can construct a random forest by repeating this classification scheme.

Several companies including DTREG.com, Insightful, Palisade Corporation, Salford Systems, and SAS market different variants of decision trees. In addition, there are many versions of free-ware including my own version, which is distributed from my website (http://c2s2.yale.edu).

Acknowledgements This research is supported in part by grant R01DA016750 from the National Institutes on Drug Abuse.

References

Altman, E.I.: Bankruptcy, Credit Risk and High Yield Junk Bonds. Blackwell Publishers, Malden, MA (2002)

Alon, U., Barkai, N., Notterman, D.A., Gish, K., Ybarra, S., Mack, D., Levine, A.J.: Broad patterns of gene expression revealed by clustering analysis of tumor and normal colon tissues probed by oligonucleotide arrays. Proc. Natl. Acad. Sci. **96**, 6745–6750 (1999)

Bacchetti, P., Segal, M.R.: Survival trees with time-dependent covariates: application to estimating changes in the incubation period of AIDS. Lifetime Data Anal. **1**, 35–47 (1995)

Bahl, L.R., Brown, P.F., de Sousa, P.V., Mercer R.L.: A tree-based language model for natural language speech recognition. IEEE Trans. AS and SP **37**, 1001–1008 (1989)

Banerjee, M., Biswas, D., Sakr, W., Wood, D.P. Jr.: Recursive partitioning for prognostic grouping of patients with clinically localized prostate carcinoma. Cancer **89**, 404–411 (2000)

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees, Wadsworth, Belmont, California (1984)

Breiman, L.: Bagging predictors. Mach. Learn. **26**, 123–140 (1994)

Breiman, L.: Bagging predictors, Mach. Learn., **26**, 123–140 (1996)

Breiman, L.: Random Forests, Mach. Learn., **45**, 5–32 (2001)

Brennan, N., Parameswaran, P. et al.: A Method for Selecting Stocks within Sectors. Schroder Salomon Smith Barney (2001)

Buhlmann, P., Yu, B.: Boosting with the L-2 loss: Regression and classification. J. Am. Stat. Assoc. **98**, 324–339 (2003)

Buhlmann, P., Yu, B.: Analyzing bagging. Ann. Stat. **30**, 927–961 (2002)

Carmelli, D., Halpern, J., Swan, G.E., Dame, A., McElroy, M., Gelb, A.B., Rosenman, R.H.: 27-year mortality in the western collaborative group study: construction of risk groups by recursive partitioning. J. Clin. Epidemiol. **44**, 1341–1351 (1991)

Carmelli, D., Zhang, H.P., Swan, G.E.: Obesity and 33 years of coronary heart disease and cancer mortality in the western collaborative group study. Epidemiology **8**, 378–383 (1997)

Chen, X., Liu, CT., Zhang, M., Zhang, H.: A forest-based approach to identifying gene and gene gene interactions. Proc Natl Acad Sci USA, **104**, 19199–19203 (2007)
Chen, X., Rusinko, A., Young, S.S.: Recursive partitioning analysis of a large structure-activity data set using three-dimensional descriptors. J. Chem. Inform. Comput. Sci. **38**, 1054–1062 (1998)
Ciampi, A., Couturier, A., Li, S.L.: Prediction trees with soft nodes for binary outcomes. Stat. Med. **21**, 1145–1165 (2002)
Ciampi, A., Hogg, S., McKinney, S., Thiffault, J.: A computer program for recursive partition and amalgamation for censored survival data. Comput Meth. Programs Biomed. **26**, 239–256 (1988)
Ciampi, A., Thiffault, J., Nakache J.-P., Asselain, B.: Stratification by stepwise regression, correspondence analysis and recursive partition: A comparison of three methods of analysis for survival data with covariates. Comput. Stat. Data Anal. **4**, 185–204 (1986)
Cox, D.R.: Regression models and life-tables (with discussion), Journal of the Royal Statistical Society, Series B, **34**, 187–220 (1972)
Cox, D.R.: The analysis of multivariate binary data. Appl. Stat. **21**, 113–120 (1972)
Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, Cambridge University Press, Cambridge (2000)
Crowley, J., LeBlanc, M., Gentleman, R., Salmon, S.: Exploratory methods in survival analysis. In: Koul, H.L., Deshpande, J.V.: (eds.) IMS Lecture Notes – Monograph Series 27, pp. 55–77. IMS, Hayward, CA (1995)
Crowley, J., LeBlanc, M., Jacobson, J., Salmon S.: Some exploratory methods for survival data. In: Lin, D.Y., Fleming, T.R. (eds.) Proceedings of the First Seattle Symposium in Biostatistics, Springer, New York (1997)
Davis, R., Anderson, J.: Exponential survival trees. Stat. Med. **8**, 947–962 (1989)
Desilva, G.L., Hull, J.J.: Proper noun detection in document images. Pattern Recogn. **27**, 311–320 (1994)
Diggle, P.J., Liang, K.Y., Zeger, S.L.: Analysis of Longitudinal Data, Oxford Science Publications, New York (1994)
Donoho, D.L.: CART and best-ortho-basis: A connection. Ann. Stat. **25**, 1870–1911 (1997)
Fisher, R.A.: The use of multiple measurements in taxonomic problems. Ann. Eugenics **7**, 179–188 (1936)
Fitzmaurice, G., Laird, N.M.: A likelihood-based method for analyzing longitudinal binary responses. Biometrika **80**, 141–151 (1993)
Fox, S.H., Whalen, G.F., Sanders, M.M., Burleson, J.A., Jennings, K., Kurtzman, S., Kreutzer, D.: Angiogenesis in normal tissue adjacent to colon cancer. J. Surg. Oncol. **69**, 230–234 (1998)
Friedman, J.H.: A recursive partitioning decision rule for nonparametric classification. IEEE Trans. Comput. **C-26**, 404–407 (1977)
Frydman, H., Altman, E.I., Kao, D.-I.: Introducing Recursive Partitioning for Financial Classification: The Case of Financial Distress. In: Altman ed. Bankruptcy, pp. 37–59. Credit Risk and High Yield Junk Bonds (2002)
Geman, D., Jedynak, B.: An active testing model for tracking roads in satellite images. IEEE Trans. Pattern Anal. Mach. Intell. **18**, 1–14 (1996)
Genuer, R., Poggi, J.M., Tuleau, C.: Random Forests: some methodological insights, Rapport de Recherche, Institut National de Recherche en Informatique et en Automatique (2008)
Goldman, L., Cook, F., Johnson, P., Brand, D., Rouan, G., Lee, T.: Prediction of the need for intensive care in patients who come to emergency departments with acute chest pain. New Engl. J. Med. **334**, 1498–504 (1996)
Goldman, L., Weinberg, M., Olshen, R.A., Cook, F., Sargent, R. et al.: A computer protocol to predict myocardial infarction in emergency department patients with chest pain. New Engl. J. Med. **307**, 588–597 (1982)
Gordon, L., Olshen, R.A.: Asymptotically efficient solutions to the classification problem. Ann. Stat. **6**, 515–533 (1978)
Gordon, L., Olshen, R.A.: Consistent nonparametric regression from recursive partitioning schemes. J. Multivariate Anal. **10**, 611–627 (1980)

Gordon, L., Olshen, R.A.: Almost surely consistent nonparametric regression from recursive partitioning schemes. J. Multivariate Anal. **15**, 147–163 (1984)
Gordon, L., Olshen, R.A.: Tree-structured survival analysis. Canc. Treat. Rep. **69**, 1065–1069 (1985)
Huang, X., Chen, S.D., Soong, S.J.: Piecewise exponential survival trees with time-dependent covariates. Biometrics **54**, 1420–14333 (1998)
Inoue, K., Slaton, J.W., Karashima, T., Shuin, T., Sweeney, P., Millikan, R., Dinney, C.P.: The prognostic value of angiogenesis factor expression for predicting recurrence and metastasis of bladder cancer after neoadjuvant chemotherapy and radical cystectomy. Clin. Canc. Res. **6**, 4866–4873 (2000)
Intrator, O., Kooperberg, C.: Trees and splines in survival analysis. Stat. Meth. Med. Res. **4**, 237–262 (1995)
Ishwaran, H., Kogalur, U.B., Blackstone, E.H., Lauer, M.S.: Random Survival Forests, the Annals of Applied Statistics, **2**, 841–860 (2008)
Kullback, S., Leibler, R.A.: On information and sufficiency, The Annals of Mathematical Statistics, **22**, 79–86 (1951)
Kwak, L.W., Halpern, J., Olshen, R.A., Horning, S.J.: Prognostic significance of actual dose intensity in diffuse large-cell lymphoma: results of a tree-structured survival analysis. J. Clin. Oncol. **8**, 963–977 (1990)
LeBlanc, M., Crowley, J.: Relative risk trees for censored survival data. Biometrics **48**, 411–425 (1992)
LeBlanc, M., Crowley, J.: Survival trees by goodness-of-split. J. Am. Stat. Assoc. **88**, 457–467 (1993)
LeBlanc, M., Crowley, J.: A review of tree-based prognostic models. In: Thall, P.F. (eds/) Recent Advances in Clinical Trial Design and Analysis, pp. 113–124. Kluwer, New York (1995)
Levin, N., Zahavi, J., Olitsky, M.: Amos – A probability-driven, customer-oriented decision support system for target marketing of solo mailings. Eur. J. Oper. Res. **87**, 708–721 (1995)
Lin, S., Cutler, D.J., Zwick, M.E., Chakravarti, A.: Haplotype inference in random population samples, American Journal of Human Genetics, **71**, 1129–1137 (2002)
Long, W.L., Griffith, J.L., Selker, H.P., D'Agostino, R.B.: A comparison of logistic regression to decision tree induction in a medical domain. Comput. Biomed. Res. **26**, 74–97 (1993)
Lugosi, G., Nobel, A.B.: Consistency of data-driven histogram methods for density estimation and classification. Ann. Stat. **24**, 687–706 (1996)
Miller, R.G.: Survival Analysis, Wiley, New York (1981)
Morgan, J.N., Sonquist, J.A.: Problems in the analysis of survey data and a proposal. J. Am. Stat. Assoc. **58**, 415–434 (1963)
Nagata, K., Okano, Y., Nozawa, Y.: Differential expression of low Mr GTP-binding proteins in human megakaryoblastic leukemia cell line, MEG-01 and their possible involvement in the differentiation process. Thromb. Haemostasis **77**, 368–375 (1997)
Nobel, A.B.: Histogram regression estimation using data-dependent partitions. Ann. Stat. **24**, 1084–1105 (1996)
Nobel, A.B., Olshen, R.A.: Termination and continuity of greedy growing for tree structured vector quantizers. IEEE Trans. Inform. Theor. **42**, 191–206 (1996)
Owens, E.A., Griffiths, R.E., Ratnatunga, K.U.: Using oblique decision trees for the morphological classification of galaxies. Mon. Not. Roy. Astron. Soc. **281**, 153–157 (1996)
Pace, R.K.: Parametric, semiparametric and nonparametric estimation of characteristic values within mass assessment and hedonic pricing models. J. R. Estate Finance Econ. **11**, 195–217 (1995)
Quinlan, J.R.: Unknown attribute values in induction. In: Proceedings of the Sixth International Machine Learning Workshop, Morgan Kaufmann, Cornell, New York (1989)
Segal, M.R.: Regression trees for censored data. Biometrics **44**, 35–48 (1988)
Segal, M.R.: Tree-structured methods for longitudinal data. J. Am. Stat. Assoc. **87**, 407–418 (1992)
Segal, M.R.: Extending the elements of tree-structured regression. Stat. Meth. Med. Res. **4**, 219–236 (1995)

Segal, M.R., Bloch, D.A.: A comparison of estimated proportional hazards models and regression trees. Stat. Med. **8**, 539–550 (1989)

Selker, H.P., Griffith, J.L., Patil, S., Long, W.L., D'Agostino, R.B.: A comparison of performance of mathematical predictive methods for medical diagnosis: Identifying acute cardiac ischemia among emergency department patients. J. Investig. Med. **43**, 468–476 (1995)

Therneau, T.M., Grambsch, P.M., Fleming, T.R.: Martingale-based residuals for survival models. Biometrika **77**, 147–160 (1990)

Toshina, K., Hirata, I., Maemura, K., Sasaki, S., Murano, M., Nitta, M., Yamauchi, H., Nishikawa, T., Hamamoto, N., Katsu, K.: Enprostil, a prostaglandin-E-2 analogue, inhibits interleukin-8 production of human colonic epithelial cell lines. Scand. J. Immunol. **52**, 570–575 (2000)

Wang, M., Chen, X., Zhang, H.: Maximal conditional chi-square importance in random forests. Bioinformatics **26**, 831–837 (2010)

Wasson, J.H., Sox, H.C., Neff, R.K., Goldman, L.: Clinical prediction rules: Applications and methodologic standards. New Engl. J. Med. **313**, 793–799 (1985)

Yeates, L.C., Powis, G.: The expression of the molecular chaperone calnexin is decreased in cancer cells grown as colonies compared to monolayer. Biochem. Biophys. Res. Comm. **238**, 66–70 (1997)

Zhang, H.P.: Splitting criteria in survival trees. In: Statistical Modelling: Proceedings of the 10th International Workshop on Statistical Modeling, pp. 305–314. Springer (1995)

Zhang, H.P.: Classification trees for multiple binary responses. J. Am. Stat. Assoc. **93**, 180–193 (1998)

Zhang, H.P., Bracken, M.B.: Tree-based risk factor analysis of preterm delivery and small-for-gestational-age birth. Am. J. Epidemiol. **141**, 70–78 (1995)

Zhang, H.P., Bracken, M.B.: Tree-based, two-stage risk factor analysis for spontaneous abortion. Am. J. Epidemiol. **144**, 989–996 (1996)

Zhang, H.P., Crowley, J., Sox, H., Olshen, R.A.: Tree structural statistical methods. Encyclopedia of Biostatistics, 6: pp. 4561–4573. Wiley, Chichester, England (2001)

Zhang, H.P., Holford, T., Bracken, M.B.: A tree-based methods of analysis for prospective studies. Stat. Med. **15**, 37–49 (1996)

Zhang, H.P., Singer, B.: Recursive Partitioning and Its Applications. Springer, New York (2010)

Zhang, H.P., Yu, C.Y., Singer, B.: Cell and tumor classification using gene expression data: Construction of forests. Proc. Natl. Acad. Sci. **100**, 4168–4172 (2003)

Zhang, H.P., Yu, C.Y., Singer, B., Xiong, M.M.: Recursive partitioning for tumor classification with gene expression microarray data. Proc. Natl. Acad. Sci. **98**, 6730–6735 (2001)

Zhao, L.P., Prentice, R.L.: Correlated binary regression using a quadratic exponential model. Biometrika **77**, 642–648 (1990)

Zhang, H., Wang, M.: Search for the smallest random forest. Statistics and its Interface **2**, 381–388 (2009)

Chapter 30
Support Vector Machines

Konrad Rieck, Sören Sonnenburg, Sebastian Mika, Christin Schäfer, Pavel Laskov, David Tax, and Klaus-Robert Müller

30.1 Introduction

In this chapter we introduce basic concepts and ideas of the Support Vector Machines (SVM). In the first section we formulate the learning problem in a statistical framework. A special focus is put on the concept of consistency, which leads to the principle of structural risk minimization (SRM). Application of these ideas to classification problems brings us to the basic, linear formulation of the SVM, described in Sect. 30.3. We then introduce the so called "kernel trick" as a tool for building a non-linear SVM as well as applying an SVM to non-vectorial data (Sect. 30.4). The practical issues of implementation of the SVM training algorithms and the related optimization problems are the topic of Sect. 30.5. Extensions of the SVM algorithms for the problems of non-linear regression and novelty detection are presented in Sect. 30.6. A brief description of the most successful applications

K. Rieck (✉) • S. Sonnenburg • K.-R. Müller
Berlin Institute of Technology, Berlin, Germany
e-mail: konrad.rieck@tu-berlin.de; klaus-robert.mueller@tu-berlin.de

S. Mika
idalab GmbH, Berlin, Germany
e-mail: mika@idalab.de; mika@first.fhg.de

C. Schäfer
Fraunhofer Institute FIRST, Berlin, Germany
e-mail: christin@first.fhg.de

P. Laskov
University of Tübingen, Tübingen, Germany
e-mail: laskov@first.fhg.de

D. Tax
Delft University of Technology, Delft, The Netherlands
e-mail: ldavidt@first.fhg.de

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_30,

of the SVM is given in Sect. 30.7. Finally, in the last Sect. 30.8 we summarize the main ideas of the chapter.

30.2 Learning from Examples

30.2.1 General Setting of Statistical Learning

The main objective of statistical learning is to find a description of an unknown dependency between measurements of objects and certain properties of these objects. The measurements, to be also called "input variables", are assumed to be observable in all objects of interest. On the contrary, the objects' properties, or "output variables", are in general available only for a small subset of objects known as *examples*. The purpose of estimating the dependency between the input and output variables is to be able to determine the values of output variables for any object of interest.

The problem of estimating an unknown dependency occurs in various practical applications. For example, the input variables can be the prices for a set of stocks and the output variable the direction of change in a certain stock price. As another example, the input can be some medical parameters and the output the probability of a patient having a certain disease. An essential feature of statistical learning is that the information is assumed to be contained in a limited set of examples (the sample), and the estimated dependency should be as accurate as possible for *all* objects of interest.

To proceed with a formal description of main properties of statistical learning, let us fix some notation. Let $\mathcal{X}$ denote the space of input variables representing the objects, and let $\mathcal{Y}$ be the space of output variables. The structure of $\mathcal{Y}$ defines the learning task. For example, if $\mathcal{Y} = \mathbb{R}$, the learning amounts to a regression problem, for $\mathcal{Y} = \{1, 2, 3\}$, the task is a classification problem with three classes, etc.

Let $\mathcal{Z} = \{(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y} \mid i = 1, \ldots, M\}$ be a given sample. We assume that there exists some unknown but fixed probability distribution $P(X, Y)$ over the space $\mathcal{X} \times \mathcal{Y}$ generating our data; that is, $(\boldsymbol{x}_i, y_i) \in \mathcal{Z}$ are drawn identically and independently from $P(X, Y)$.

The dependency to be estimated takes the form of a function $f : \mathcal{X} \to \mathcal{Y}$. To decide which of many possible functions best describes the dependency observed in the training sample, we introduce the concept of a ***loss function***:

$$\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}. \tag{30.1}$$

Such a loss function should be bounded from below and should measure the cost $\ell(f(\boldsymbol{x}), y)$ of discrepancy between the predicted value $f(\boldsymbol{x}) \in \mathcal{Y}$ and the true value $y \in \mathcal{Y}$. Then the ***risk***, i.e. the expected loss incurred from using a particular prediction function f, can be defined as:

$$\mathrm{R}(f) = \mathbb{E}_P[\ell(f(\boldsymbol{x}), y)], \tag{30.2}$$

where $\mathbb{E}_P$ denotes the expectation with respect to the joint distribution $P(X, Y)$ of input and output variables.

Notice that, if we would know the joint distribution $P(X, Y)$, the learning problem can be easily solved. For example, in the classification case one could calculate the conditional probability $P(Y|X)$ and compute the so called "**Bayes-optimal solution**":

$$f^*(\boldsymbol{x}) = \underset{y_1 \in \mathcal{Y}}{\operatorname{argmax}} \int_{y_2 \in \mathcal{Y}} \ell(y_1, y_2)\ P(Y = y_2 | X = \boldsymbol{x}). \tag{30.3}$$

However, in our setup $P(X, Y)$ is unknown, and only a sample $\mathcal{Z}$ is available. One possible solution would be to estimate $P(X, Y)$ or $P(Y|X)$ from the sample $\mathcal{Z}$. In many theoretical and practical approaches the inference is carried out exactly in this way (Bishop 1995; Devroye et al. 1996; Duda et al. 2001). But it is also well known that estimating a density from empirical data is a hard problem, especially in the multi-dimensional case. The number of examples one needs in order to get a reliable estimate of a density in N dimensions grows exponentially with N – a well-known difficulty denoted as *curse of dimensionality*.

In the approach to be followed in this chapter we shall attempt to estimate the function f directly from $\mathcal{Z}$ without using $P(X, Y)$ or $P(Y|X)$. For this, the following three steps are necessary. First, a class of functions $\mathcal{F}$ needs to be defined. Second, a suitable loss ℓ is to be fixed. Finally, a method has to be provided to find the function f which minimizes the risk $\mathrm{R}(f)$ among all $f \in \mathcal{F}$. Such method is called an "induction principle". Desirable properties of such an induction principle are discussed in the next section.

30.2.2 Desirable Properties for Induction Principles

The most commonly used induction principle is the one of minimizing the **empirical risk**

$$\mathrm{R}_{\mathrm{emp}}(f) = \frac{1}{M} \sum_{i=1}^{M} \ell(f(\boldsymbol{x}_i), y_i), \tag{30.4}$$

which is the empirical counterpart of the **expected risk** (30.2). The goal of learning in our setup is to find an algorithm that, given a training sample $\mathcal{Z}$, finds a function $f \in \mathcal{F}$ that minimizes (30.4). Notice that this will not necessarily result in a unique solution. As one can see in Fig. 30.1 more than one function can have the same (e.g. zero) empirical risk on the same data sample.

However, these functions can take arbitrary values at other points in $\mathcal{X}$; hence the solution that minimizes the empirical risk is not guaranteed to minimize the true risk (30.2).

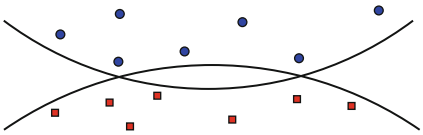


Fig. 30.1 Two functions that separate two classes of data points with zero empirical risk. Without further information it is impossible to decide for one of them

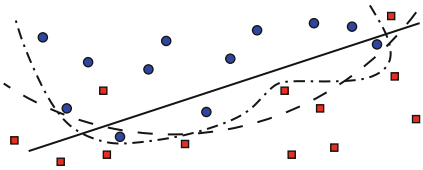


Fig. 30.2 An illustration of underfitting and overfitting on a small sample. The simple linear function (*solid line*) underfits the data and already makes training errors. The complex one (*dash-dotted line*) has no training error but may not generalize well on unseen data. The function with intermediate complexity (*dashed line*) seems to capture the decision boundary best

The other two phenomena arising in relation with the minimization of the empirical risk (30.4) are ***overfitting*** and ***underfitting***. An overly complex function f might describe the training data well but does not generalize to unseen examples. The converse could also happen. Assume the function class $\mathcal{F}$ we can choose from is very small, e.g. it contains only a single, fixed function. Then our learning machine would trivially be consistent, since $\mathrm{R}(f) = const$ for all $f \in \mathcal{F}$. But if this single $f \in \mathcal{F}$ is not by accident the rule that generates our data, the decisions are unrelated to the concept generating our data. This phenomenon is called underfitting (cf. Fig. 30.2).

Apparently we need some way of controlling how large the class of functions $\mathcal{F}$ is, such that we avoid overfitting and underfitting and obtain solutions that generalize well (i.e. with reasonable complexity). The questions of **consistency**, overfitting and underfitting are closely related and will lead us to a concept known as regularization (e.g. Morozov 1984; Tikhonov and Arsenin 1977) and to the principle of Structural Risk Minimization (Vapnik 1998).

Regularization

In the previous paragraphs we have shown that for successful learning it is not enough to find a function with minimal empirical risk. If we are interested in a good estimation of the true risk on all possible data points, we need to introduce a complexity control and choose our solution by minimizing the following objective function:

$$\mathrm{R}_{\mathrm{emp}}(f, \mathcal{Z}) + \lambda \Omega(f). \tag{30.5}$$

This equation shows a regularization approach. We add a penalty term to make the trade-off between the complexity of the function class and the empirical error. Using such a regularization, a bound for the true risk can be derived.

There are several possibilities to choose λ and Ω in order to derive a consistent inductive principle. In the following sections we will describe the choice inspired by the work of Vapnik. Other possible choices are for example Akaike information criterion (Akaike 1974) or Mallows Cp (Mallows 1973), used in classical statistics, as well as spline-regularization (Wahba 1980), wavelet regularization (Donoho et al. 1996), CART (Breiman et al. 1984) and many other modern approaches. A general foundation for regularization in model selection is given in (Barron et al. 1999). Bartlett and Mendelson (2002) investigate regularization in the context of SVM.

Consistency

Let us define more closely what consistency means and how it can be characterized. Let us denote by f^M the function $f \in \mathcal{F}$ that minimizes (30.4) for a given training sample $\mathcal{Z}$ of size M. The notion of ***consistency*** implies that, as $M \to \infty$, $|\mathrm{R}(f^M) - \mathrm{R}_{\mathrm{emp}}(f^M)| \to 0$ in probability. We have already seen in a previous example that such convergence may not be the case in general, the reason being that f^M now depends on the sample $\mathcal{Z}$. One can show that a necessary and sufficient condition for consistency is *uniform* convergence, over all functions in $\mathcal{F}$, of the difference between the expected and the empirical risk to zero. This insight is summarized in the following theorem:

Theorem 1 (Vapnik and Chervonenkis 1991). *One-sided uniform convergence in probability, i.e.*

$$\lim_{M\to\infty} \mathbb{P}\left[\sup_{f\in\mathcal{F}} \left(\mathrm{R}(f) - \mathrm{R}_{\mathrm{emp}}(f)\right) > \epsilon\right] = 0, \tag{30.6}$$

for all $\epsilon > 0$, is a necessary and sufficient condition for (nontrivial) consistency of empirical risk minimization.

Since the condition in the theorem is not only sufficient but also necessary it seems reasonable that any "good" learning machine implementing a specific function class should satisfy condition (30.6).

30.2.3 Structural Risk Minimization

Consequently, the question arises how one can choose function classes that satisfy Theorem 1 in practice? It will turn out that this is possible and it crucially depends on the question how complex the functions in the class $\mathcal{F}$ are, a question we have

already seen to be equally important when talking about overfitting and underfitting. But what does complexity mean and how can one *control* the size of a function class?

The complexity of a function class can be measured by the number of different possible combinations of outcome assignments when choosing functions from this class. This quantity is usually difficult to obtain theoretically for useful classes of functions. Popular approximations of this measure are covering numbers (Shawe-Taylor et al. 1998), annealed entropy and fat-shattering dimension (Bartlett et al. 1996), VC-entropy and **VC-dimension** (Vapnik 1998), or Rademacher and Gaussian complexity (Bartlett and Mendelson 2002). We will not go into detail about these quantities here.

A specific way of controlling the complexity of a function class is given by VC-theory and the principle of ***Structural Risk Minimization*** (Vapnik 1998). Here the concept of complexity is captured by the VC-dimension h of the function class $\mathcal{F}$. Roughly speaking, the VC-dimension measures how many (training) points can be shattered (i.e. separated for all possible labellings) using functions of the class. This quantity can be used to bound the probability that the expected error deviates much from the empirical error for any function from the class, that is VC-style bounds usually take the form

$$\left[\sup_{f\in\mathcal{F}}\left(\mathrm{R}(f)-\mathrm{R}_{\mathrm{emp}}(f,\mathcal{Z})\right)>\epsilon\right]\leq H(\mathcal{F},M,\epsilon), \tag{30.7}$$

where H is some function that depends on properties of the function class $\mathcal{F}$, e.g. the VC-dimension, the size M of the training set and the desired closeness ϵ. By equating the right-hand side of (30.7) to $\delta>0$ and solving $H=\delta$ for ϵ one can turn these bounds into expressions of the following form: with probability at least $1-\delta$ over the random draw of the training sample $\mathcal{Z}$,

$$\mathrm{R}(f)\leq\mathrm{R}_{\mathrm{emp}}(f,\mathcal{Z})+\widetilde{H}(\mathcal{F},M,\delta), \tag{30.8}$$

where $\widetilde{H}$ is the penalty term that measures our degree of uncertainty. If the function class is simple then $\widetilde{H}$ is small. This penalty term usually increases if we require a higher precision (e.g. with $\log(\frac{1}{\delta})$) and decreases if we observe more examples (e.g. with $\frac{1}{M}$ or $\frac{1}{\sqrt{M}}$). Note that this prototypical bound is structurally identical to the regularized risk functional in (30.5). The practical implication of bounds like (30.8) is that our learning machine should be constructed such that:

1. It finds a function with a small empirical error, and
2. At the same time keeps the penalty term $\widetilde{H}$ small.

Only if our learning principle can control both quantities we have a guarantee that the expected error of our estimate will be small (cf. Fig. 30.3).

One of the most famous of these VC-style bounds is due to Vapnik and Chervonenkis:

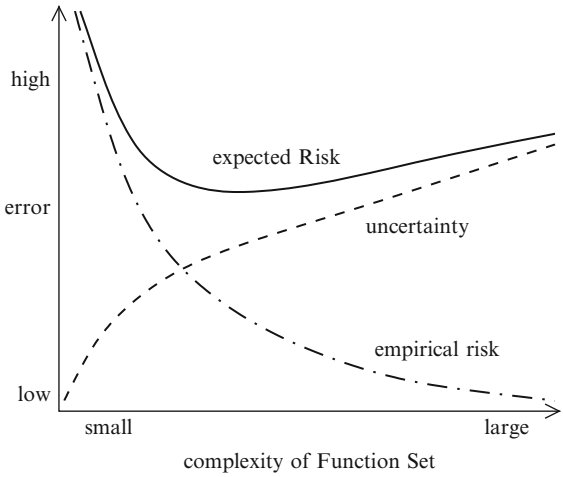


Fig. 30.3 Schematic illustration of (30.8). The dash-dotted line represents the training error (*empirical risk*), the dashed line the upper bound on the complexity term. With higher complexity the empirical error decreases but the upper bound on the risk uncertainty becomes worse. For a certain complexity of the function class the best expected risk (*solid line*) is obtained. Thus, in practice the goal is to find the best trade-off between empirical error and complexity

Theorem 2 (Vapnik and Chervonenkis 1991). *Let h denote the VC-dimension of the function class $\mathcal{F}$ and let $\mathrm{R}_{\mathrm{emp}}$ be defined by (30.4) using the $0/1$-loss. For all $\delta > 0$ and $f \in \mathcal{F}$ the inequality bounding the risk*

$$\mathrm{R}(f) \leq \mathrm{R}_{\mathrm{emp}}(f, \mathcal{Z}) + \sqrt{\frac{h\left(\ln \frac{2M}{h} + 1\right) - \ln(\delta/4)}{M}} \tag{30.9}$$

holds with probability of at least $1 - \delta$ for $M > h$ over the random draw of the training sample $\mathcal{Z}$.

This theorem lays the ground for the SVM algorithm that we will consider in more detail in Sect. 30.3.

Based on Theorem 2 the principle of Structural Risk Minimization (SRM) has been derived (e.g. Cortes and Vapnik 1995; Vapnik 1998). According to this principle a nested family of function classes $\mathcal{F}_1 \subseteq \cdots \subseteq \mathcal{F}_k$ with non-decreasing VC-dimension $h_1 \leq \cdots \leq h_k$ is constructed. After the solutions $f_1, \ldots, f_k$ of the empirical risk minimization (30.4) in the function classes $\mathcal{F}_1, \ldots, \mathcal{F}_k$ have been found, the principle chooses the function class $\mathcal{F}_i$ (and the function f_i) such that an upper bound on the generalization error like (30.9) is minimized.

30.3 Linear SVM: Learning Theory in Practice

Having summarized the prerequisites from statistical learning theory, we now give an example of a particular learning machine that builds upon these insights. The **Support Vector Machine** algorithm (SVM) developed by Vapnik and others (e.g. Boser et al. 1992; Cortes and Vapnik 1995; Cristianini and Shawe-Taylor 2000; Müller et al. 2001; Schölkopf and Smola 2002; Vapnik 1998, and numerous

others) is one of the most successful classification techniques over the last decade, especially after being combined with the kernel idea which we shall discuss in Sect. 30.4.

30.3.1 Linear Separation Planes

We are now going to discuss how one could possibly control the size of a function class and how to select the empirical risk minimizer in this class.

In the following, let us assume that we are dealing with a two class classification problem (i.e. $\mathcal{Y} = \{-1, +1\}$) in a real-valued vector space, e.g. $\mathcal{X} = \mathbb{R}^N$. Further, we assume that the distribution of these two classes is such that they are linearly separable, i.e. one can find a linear function of the inputs $\boldsymbol{x} \in \mathcal{X}$ such that $f(\boldsymbol{x}) < 0$ whenever the label $y = -1$ and $f(\boldsymbol{x}) \geq 0$ otherwise. This can be conveniently expressed by a **hyperplane** in the space $\mathcal{X}$, i.e. we are looking for a function f of the form

$$f(\boldsymbol{x}) = (\boldsymbol{w}^\top \boldsymbol{x}) + b. \tag{30.10}$$

Assume that the function class $\mathcal{F}$ we choose our solution from is the one containing all possible hyperplanes, i.e. $\mathcal{F} = \{f : \mathcal{X} \to \mathbb{R} \mid f(\boldsymbol{x}) = (\boldsymbol{w}^\top \boldsymbol{x}) + b\}$. For $\mathcal{X} = \mathbb{R}^N$ it is rather straightforward to show that the VC-dimension of this class of functions will be $h = N + 1$, that is, in an N-dimensional space the maximal number of points that can be separated for an arbitrary labelling using a hyperplane is $N + 1$.

30.3.2 Canonical Hyperplanes and Margins

To apply the SRM principle in practice, not only must the VC-dimension of the class of hyperplanes be finite, rather a nested structure of function classes must be defined. To this end we define the function classes

$$\mathcal{F}_\Lambda = \{f : \mathbb{R}^N \to \mathbb{R} \mid f(x) = (\boldsymbol{w}^\top \boldsymbol{x}) + b, \|\boldsymbol{w}\| \leq \Lambda\}. \tag{30.11}$$

Clearly $\mathcal{F}_{\Lambda_1} \subseteq \mathcal{F}_{\Lambda_2}$ whenever $\Lambda_1 \leq \Lambda_2$. But what effect does constraining the norm of the weight vector have on the corresponding VC-dimensions of $\mathcal{F}_\Lambda$? It turns out that we also get $h(\mathcal{F}_{\Lambda_1}) \leq h(\mathcal{F}_{\Lambda_2})$ for $\Lambda_1 \leq \Lambda_2$, as we will see shortly in (30.12).

The crucial ingredient in making the function classes $\mathcal{F}_\Lambda$ nested is to define a unique representation for each hyperplane. We introduce the concept of canonical hyperplanes and the notion of margins. If the data are separable by $(\boldsymbol{w}, b)$ then they are also separable by any (positive) multiple of $(\boldsymbol{w}, b)$ and hence there exist an infinite number of representations for the same separating hyperplane. In particular,

all function classes $\mathcal{F}_\Lambda$ would have the same VC-dimension as they would contain the same functions in different representations.

A **canonical hyperplane** with respect to a sample $\mathcal{Z}$ of M points is defined as a function

$$f(\boldsymbol{x}) = (\boldsymbol{w}^\top \boldsymbol{x}) + b,$$

where $\boldsymbol{w}$ is normalized such that

$$\min_{i=1,\ldots,M} |f(\boldsymbol{x}_i)| = 1.$$

The notion of a canonical hyperplane is illustrated in Fig. 30.4. Notice that none of the training examples produces an absolute output that is smaller than one and the examples closest the hyperplane have exactly an output of one, i.e. $(\boldsymbol{w}^\top \boldsymbol{x})+b = \pm 1$. In Sect. 30.5, we will see that the latter objects will be used in the description of the hyperplane, and they are therefore called the ***support vectors***. In Fig. 30.4 these are the objects which are connected to the decision boundary by the dashed lines. Since we assumed the sample $\mathcal{Z}$ to be linearly separable, we can turn any f that separates the data into a canonical hyperplane by suitably normalizing the weight vector $\boldsymbol{w}$ and adjusting the threshold b correspondingly.

The ***margin*** is defined to be the minimal Euclidean distance between any training example $\boldsymbol{x}_i$ and the separating hyperplane. Intuitively, the margin measures how good the separation between the two classes by a hyperplane is. If this hyperplane is in the canonical form, the margin can be measured by the length of the weight vector $\boldsymbol{w}$. Consider two support vectors $\boldsymbol{x}_1$ and $\boldsymbol{x}_2$ from different classes. The margin is given by the projection of the distance between these two points on the direction perpendicular to the hyperplane. This distance can be computed as (e.g. Vapnik 1998)

$$\left(\frac{\boldsymbol{w}^\top}{\|\boldsymbol{w}\|} (\boldsymbol{x}_1 - \boldsymbol{x}_2) \right) = \frac{2}{\|\boldsymbol{w}\|}.$$

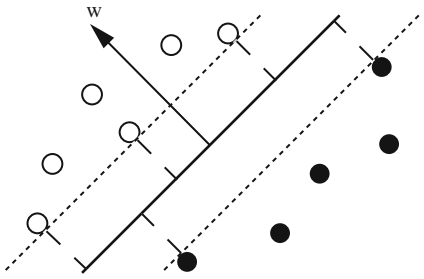


Fig. 30.4 Linear SVM and margins. A linear SVM classifier is defined by the normal vector $\boldsymbol{w}$ of a hyperplane and an offset b. The decision boundary is $\{\boldsymbol{x} | (\boldsymbol{w}^\top \boldsymbol{x})+b = 0\}$ (*solid line*). Each of the two half spaces induced by this hyperplane corresponds to one class, i.e. $f(\boldsymbol{x}) = \text{sgn}((\boldsymbol{w}^\top \boldsymbol{x})+b)$. The margin of a linear classifier is the minimal distance of any training point to the hyperplane. For the case shown in the picture it is the distance between the dotted lines and the solid line

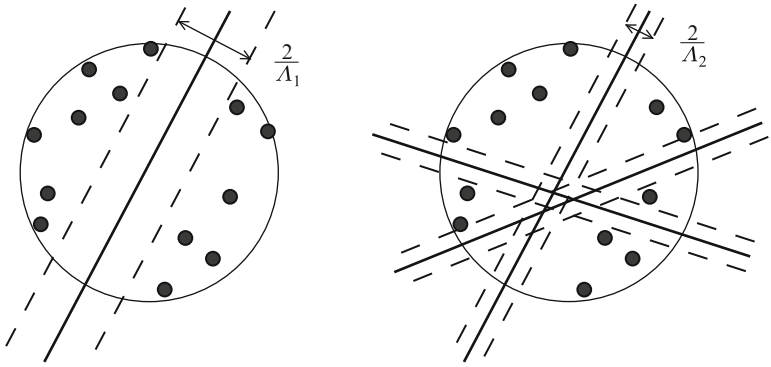


Fig. 30.5 Illustration of why a large margin reduces the complexity of a linear hyperplane classifier. If we choose hyperplanes with a large margin, there is only a small number of possibilities to separate the data, i.e. the VC-dimension of $\mathcal{F}_{\Lambda_1}$ is small (*left panel*). On the contrary, if we allow smaller margins there are more separating hyperplanes, i.e. the VC-dimension of $\mathcal{F}_{\Lambda_2}$ is large (*right panel*)

The smaller the norm of the weight vector $\boldsymbol{w}$ in the canonical representation, the larger the margin.

More generally, it was shown (e.g. Vapnik 1998) that if the hyperplane is constructed under the constraint $\|\boldsymbol{w}\|_2 \leq \Lambda$ then the VC-dimension of the class $\mathcal{F}_\Lambda$ is bounded by

$$h \leq \min(\Lambda^2 R^2 + 1, N + 1), \tag{30.12}$$

where R is the radius of the smallest sphere around the data. Thus, if we bound the margin of a function class from below, say by $\frac{2}{\Lambda}$, we can control its VC-dimension and hence apply the SRM principle as shown in Fig. 30.5.

A particularly important insight is that the complexity only indirectly depends on the dimensionality of the data. This is very much in contrast to density estimation, where the problems become more difficult as the dimensionality of the data increases. For SVM classifier, if we can achieve a large margin the problem remains simple.

30.4 Kernel Functions

In the previous section we have seen that by restricting ourselves to linear functions one can control the complexity of a learning machine. We have thus avoided the problem of dealing with too complex functions at the price of being able to solve only linearly separable problems. In the following we show how to extend the linear SVM for constructing a rich set of non-linear decision functions by abstracting the task of learning from the actual data representation. Based on this abstraction we then introduce techniques for learning with structured data, such as strings and trees.

Central to the success of non-linear SVM was the re-discovery of the so called ***Reproducing Kernel Hilbert Spaces*** (RKHS) and ***Mercer's Theorem*** (Boser et al. 1992). There is a large body of literature dealing with kernel functions, their theory and applicability, see e.g. Kolmogorov (1941), Aronszajn (1950), Aizerman et al. (1964), Boser et al. (1992) or Schölkopf and Smola (2002), Shawe-Taylor and Cristianini (2004) for an overview. We only recall the basic definitions and properties necessary for turning our linear, hyperplane based learning technique into a very powerful algorithm capable of finding non-linear decision functions with controllable complexity.

30.4.1 The Kernel Trick

The basic idea of the so called ***kernel methods*** is to first preprocess the data by some non-linear mapping Φ and then to apply the same linear algorithm as before but in the image space of Φ (cf. Fig. 30.6 for an illustration).

More formally we apply the mapping

$$\begin{aligned} \Phi : \quad & \mathbb{R}^N \to \mathcal{E} \\ & \boldsymbol{x} \quad \mapsto \Phi(\boldsymbol{x}) \end{aligned}$$

to the data $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M \in \mathcal{X}$ and consider our algorithm in $\mathcal{E}$ instead of $\mathcal{X}$, i.e. the sample is preprocessed as

$$\{(\Phi(\boldsymbol{x}_1), y_1), \ldots, (\Phi(\boldsymbol{x}_M), y_M)\} \subseteq (\mathcal{E} \times \mathcal{Y})^M.$$

In certain applications we might have sufficient knowledge about our problem such that we can design an appropriate Φ by hand (e.g. Blankertz et al. 2002; Zien et al. 2000). An alternative strategy is to consider a class of mappings and choose

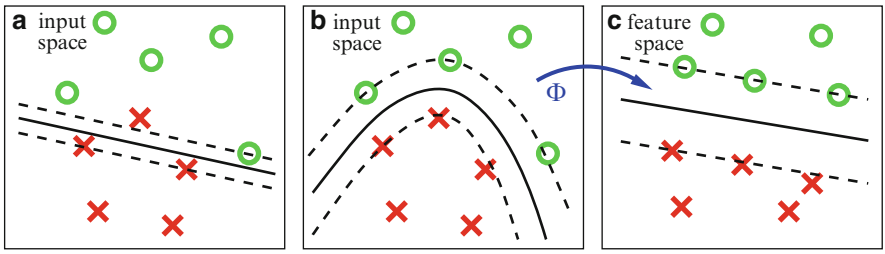


Fig. 30.6 Three views on the same two class separation problem (Zien et al. 2000). (**a**) A linear separation of the input points is not possible without errors. Even allowing misclassification of one data point results in a small margin. (**b**) A better separation is provided by a non-linear surface in the input space. (**c**) This non-linear surface corresponds to a linear surface in a feature space. Data points are mapped from input space to feature space by the function Φ induced by the kernel function k

the Φ providing the best representation for a particular learning task (Braun et al. 2008). If this mapping is not too complex to compute and the space $\mathcal{E}$ is not too high-dimensional, we might just explicitly apply this mapping to our data. Similar transformations are applied in neural networks (Bishop 1995), radial basis networks (e.g. Moody and Darken 1989) or Boosting algorithms (Freund and Schapire 1997), where the input data is mapped to some representation given by the hidden layer, the RBF bumps or the hypotheses space, respectively (Rätsch et al. 2002). The difference with kernel methods, however, is that for a suitably chosen Φ we get an algorithm that has powerful non-linearities but is still very intuitive and retains most of the favorable properties of its linear input space version.

The problem with explicitly using the mapping Φ to construct a feature space is that the resulting space can be extremely high-dimensional. As an example consider the case when the input space $\mathcal{X}$ consists of images of 16×16 pixels, i.e. 256 dimensional vectors, and we choose 5th order monomials as non-linear features. The dimensionality of such space would be

$$\binom{5 + 256 - 1}{5} \approx 10^{10}.$$

Such a mapping would clearly be intractable to carry out explicitly. We are not only facing the technical problem of storing the data and doing the computations, but we are also introducing problems due to the fact that we are now working in an extremely sparsely sampled space.

The problems concerning the storage and the manipulation of the high dimensional data, however, can be alleviated. It turns out that for a certain class of mappings we are well able to compute scalar products in this new space even if it is extremely high dimensional. Simplifying the above example of computing all 5th order products of 256 pixels to that of computing all 2^{nd} order products of two "pixels", i.e.

$$\boldsymbol{x} = (x_1, x_2) \text{ and } \Phi(\boldsymbol{x}) = (x_1^2, \sqrt{2}x_1x_2, x_2^2),$$

the computation of a scalar product between two such feature space vectors can be readily reformulated in terms of a so-called **kernel function** k:

$$\begin{aligned}(\Phi(\boldsymbol{x})^\top \Phi(\boldsymbol{z})) &= (x_1^2, \sqrt{2}\, x_1x_2, x_2^2)(z_1^2, \sqrt{2}\, z_1z_2, z_2^2)^\top \\ &= ((x_1, x_2)(z_1, z_2)^\top)^2 \\ &= (\boldsymbol{x}^\top \boldsymbol{z})^2 \\ &=: \mathrm{k}(\boldsymbol{x}, \boldsymbol{z}).\end{aligned}$$

This finding generalizes: For $\boldsymbol{x}, \boldsymbol{z} \in \mathbb{R}^N$, and $d \in \mathbb{N}$ the kernel function

$$\mathrm{k}(\boldsymbol{x}, \boldsymbol{z}) = (\boldsymbol{x}^\top \boldsymbol{z})^d$$

computes a scalar product in the space of all products of d vector entries (monomials) of $\boldsymbol{x}$ and $\boldsymbol{z}$ (Schölkopf et al. 1998b; Vapnik 1998).

The ***kernel trick*** (Aizerman et al. 1964; Boser et al. 1992; Vapnik 1998) is to take the original algorithm and formulate it such, that we only use $\Phi(\boldsymbol{x})$ in scalar products. Then, if we can efficiently evaluate these scalar products, we do not need to carry out the mapping Φ explicitly and can still solve the problem in the huge feature space $\mathcal{E}$. Furthermore, we do not need to know the mapping Φ but only the kernel function.

Now we can ask two questions:

1. For which mappings Φ does there exist a simple way to evaluate the scalar product?
2. Under which conditions does a function $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ correspond to a scalar product?

The first question is difficult to answer in general. But for the second question there exists an answer which we present in the following.

30.4.2 Feature Spaces

To address the question whether a kernel function $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ equals a scalar product in some feature space, let us first introduce some more notation and definitions. Given a training sample $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\} \subseteq \mathcal{X}$, the $M \times M$ matrix K with elements $K_{ij} = \mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x}_j)$ is called the kernel matrix or the Gram matrix. An $M \times M$ matrix K (and any other symmetric matrix) is said to be positive semi-definite if any quadratic form over K is positive or zero, i.e. for all $r_i \in \mathbb{R}$, $i = 1, \ldots, M$, we have

$$\sum_{i,j=1}^{M} r_i r_j K_{ij} \geq 0. \tag{30.13}$$

Positive semi-definite kernels are exactly those giving rise to a positive semi-definite **kernel matrix** K for all M and all sets $\{\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M\} \subseteq \mathcal{X}$. Note that for a kernel (and a matrix) to be positive semi-definite, it is necessary to be symmetric and non-negative on the diagonal.

For any positive semi-definite kernel k we can construct a mapping Φ into a feature space $\mathcal{E}$, such that k acts as a scalar product over Φ. As a matter of fact, it is possible to construct more than one of these spaces. We will omit many crucial details and only present the central results. For more details see Schölkopf and Smola (2002).

The Feature Map

Given a real-valued, positive semi-definite kernel function k, defined over a non-empty set $\mathcal{X}$, we define the feature space $\mathcal{E}$ as the space of all functions mapping

from $\mathcal{X}$ to $\mathbb{R}$, i.e. as $\mathcal{E} = \mathbb{R}^{\mathcal{X}} = \{f \mid f : \mathcal{X} \to \mathbb{R}\}$. Notice that, unlike the example in Fig. 30.6, this feature space is not a usual Euclidean space but rather a vector space of functions. The mapping Φ is now defined as

$$\Phi : \mathcal{X} \to \mathbb{R}^{\mathcal{X}}, \Phi(\boldsymbol{x}) = \mathrm{k}(\cdot, \boldsymbol{x}), \tag{30.14}$$

i.e. Φ maps each $\boldsymbol{x}$ to the function $\mathrm{k}(\cdot, \boldsymbol{x})$, i.e. the kernel k where the first argument is free and the second is fixed to $\boldsymbol{x}$ (e.g., Schölkopf et al. 1999). One can show that the set of all linear combinations of the form

$$f(\cdot) = \sum_{i=1}^{M} \alpha_i \, \mathrm{k}(\cdot, \boldsymbol{x}_i), \tag{30.15}$$

for arbitrary $M, \alpha_i \in \mathbb{R}$, and $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_M$ forms a vector space. Especially, for all functions of the form (30.15) one gets

$$\langle \mathrm{k}(\cdot, \boldsymbol{x}), f \rangle_{\mathcal{H}} = f(\boldsymbol{x}),$$

where $\langle \cdot, \cdot \rangle_{\mathcal{H}}$ denotes the scalar product in some Hilbert space that will become clearer below. In particular we have

$$\begin{aligned} \langle \mathrm{k}(\cdot, \boldsymbol{x}), \mathrm{k}(\cdot, z) \rangle_{\mathcal{H}} &= \langle \Phi(\boldsymbol{x}), \Phi(z) \rangle_{\mathcal{E}} \\ &= \mathrm{k}(\boldsymbol{x}, z). \end{aligned}$$

The last property is the reason why positive semi-definite kernels are also called reproducing kernels: they reproduce the evaluation of f on $\boldsymbol{x}$. It also shows that k indeed computes, as desired, the scalar product in $\mathcal{E}$ for $\Phi(\boldsymbol{x})$ defined as in (30.14). Hence (30.14) is one possible realization of the mapping associated with a kernel and is called the feature map (for its empirical counterpart see e.g. Mika 2002). The following is a formal definition of a ***Reproducing Kernel Hilbert Space*** (cf. Schölkopf and Smola 2002).

Definition 1 (Reproducing Kernel Hilbert Space (RKHS)). Let $\mathcal{X}$ be a nonempty set and $\mathcal{H}$ a Hilbert space of functions $f : \mathcal{X} \to \mathbb{R}$. Then $\mathcal{H}$ is called a *reproducing kernel Hilbert space* endowed with the dot product $\langle \cdot, \cdot \rangle$ if there exists a function $\mathrm{k} : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ with the properties that:

1. k has the reproducing property $\langle f, \mathrm{k}(\cdot, \boldsymbol{x}) \rangle = f(\boldsymbol{x})$ for all $f \in \mathcal{H}$, in particular $\langle \mathrm{k}(\cdot, \boldsymbol{x}), \mathrm{k}(\cdot, z) \rangle = \mathrm{k}(\boldsymbol{x}, z)$, and
2. k spans $\mathcal{H}$, i.e. $\mathcal{H} = \overline{\mathrm{span}\{\mathrm{k}(\cdot, \boldsymbol{x}) | \boldsymbol{x} \in \mathcal{X}\}}$, where $\overline{A}$ denotes the completion of the set A.

One can show, that the kernel k for such a RKHS is uniquely determined.

Mercer Kernels

As a second way to identify a feature space associated with a kernel k one can use a technique derived from Mercer's Theorem.

The Mercer's Theorem, which we will reproduce in the following, states that if a function k gives rise to a positive integral operator, the evaluation of $\mathrm{k}(\boldsymbol{x},z)$ can be expressed as a finite or infinite, absolute and uniformly convergent series, almost everywhere. This series defines a feature space and an associated mapping connected to the function k.

Let $\mathcal{X}$ be a finite measure space, i.e. a space with a σ-algebra and a measure μ satisfying $\mu(\mathcal{X}) \leq \infty$.

Theorem 3 (Mercer 1909). *Suppose* $\mathrm{k} \in L_\infty(\mathcal{X}^2, \mu)$ *is a symmetric real-valued function such that the integral operator*

$$T_\mathrm{k} : L_2(\mathcal{X}, \mu) \to L_2(\mathcal{X}, \mu),\ (T_\mathrm{k} f)(\boldsymbol{x}) := \int_{\mathcal{X}} \mathrm{k}(\boldsymbol{x}, z) f(z) d\mu(z)$$

is positive semi-definite, i.e. for all $f \in L_2(\mathcal{X}, \mu)$

$$\int_{\mathcal{X}^2} \mathrm{k}(\boldsymbol{x}, z) f(\boldsymbol{x}) f(z) d\mu(\boldsymbol{x}) d\mu(z) \geq 0.$$

Let $\varphi_j \in L_2(\mathcal{X}, \mu)$ *be the normalized orthogonal eigenfunctions of* T_k *associated with the eigenvalues* $\lambda_j \geq 0$*, sorted in non-increasing order. Then:*

1. $(\lambda_j)_j \in l_1$
2. $\mathrm{k}(\boldsymbol{x}, z) = \sum_{j=1}^{N_\mathcal{E}} \lambda_j \varphi_j(\boldsymbol{x}) \varphi_j(z)$ *holds for almost all* $\boldsymbol{x}, z$*. Either* $N_\mathcal{E} \in \mathbb{N}$ *or* $N_\mathcal{E} = \infty$*; in the latter case, the series converges absolutely and uniformly for almost all* $\boldsymbol{x}, z$*.*

If we choose as feature space $\mathcal{E} = l_2^{N_\mathcal{E}}$ and the mapping Φ as

$$\Phi : \mathcal{X} \to l_2^{N_\mathcal{E}},\ \Phi(\boldsymbol{x}) = (\sqrt{\lambda_j} \varphi_j(\boldsymbol{x}))_{j=1,\ldots,N_\mathcal{E}},$$

we see from the second statement in Theorem 3 that the kernel k corresponds to the dot product in $l_2^{N_\mathcal{E}}$, i.e. $\mathrm{k}(\boldsymbol{x}, z) = \langle \Phi(\boldsymbol{x}), \Phi(z) \rangle$.

The kernels satisfying the Mercer's Theorem are called ***Mercer kernels***. It can be shown that, if the set $\mathcal{X}$ on which the kernel is defined, is compact, a kernel is a Mercer kernel if and only if it is a positive semi-definite kernel (cf. Smola et al. 1998). Table 30.1 lists some of the most widely used kernel functions in machine learning applications.

Note that recently Braun et al. (2008) have observed that the excellent generalization that is typically observed when using SVMs in high-dimensional applications with few samples is due to its very economic representation in the feature space $\mathcal{E}$.

Table 30.1 Common kernel functions:

$$\text{Gaussian RBF:} \quad \mathrm{k}(\boldsymbol{x}, z) = \exp\left(\frac{-\|\boldsymbol{x} - z\|^2}{c}\right) \tag{30.16}$$

$$\text{Polynomial:} \quad \mathrm{k}(\boldsymbol{x}, z) = ((\boldsymbol{x}^\top z) + \boldsymbol{\theta})^d \tag{30.17}$$

$$\text{Inverse multi-quadric:} \quad \mathrm{k}(\boldsymbol{x}, z) = \frac{1}{\sqrt{\|\boldsymbol{x} - z\|^2 + c^2}}$$

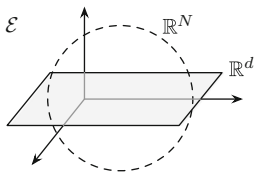


Fig. 30.7 Illustration of the feature space representation. The data is embedded in a high-dimensional feature space $\mathcal{E}$. Since only, say, N data points exist, the data given through a normalized kernel is situated in a N-dimensional ball in $\mathcal{E}$. However, only a small d-dimensional subspace of the N-dimensional ball in $\mathcal{E}$ is task relevant, e.g. for the classification or regression at hand. Thus, if the kernel is well chosen, then kernel methods make very economical use of the data as they map the data into an effectively very low dimensional task relevant subspace of $\mathcal{E}$ (see also Braun et al. 2008 for further discussion and proofs)

Given the appropriate kernel, only a very low dimensional subspace is task relevant (see Fig. 30.7).

30.4.3 Kernels for Structured Data

Another important feature of kernel functions is that they are not restricted to operate on vectorial data. Kernels can be defined over any type of data including discrete and structured representations. Consequently, a large body of research has studied kernel functions for structured data, such as for analysis of strings and sequences (e.g. Lodhi et al. 2002; Sonnenburg et al. 2007a; Watkins 2000), hierarchical representations and trees (e.g. Collins and Duffy 2002; Kashima and Koyanagi 2002; Rieck et al. 2010) as well as network and graph structures (e.g. Gärtner et al. 2004; Kashima et al. 2004; Vishwanathan et al. 2010). We herein provide a brief introduction to kernel functions defined over strings and trees. An extensive discussion of kernels for structured data is provided by Shawe-Taylor and Cristianini (2004).

String Kernels

Strings and sequences are a natural representation of data in many areas of computer science. For example, several applications in bioinformatics are concerned with

studying strings of DNA and many tasks of information retrieval center on analysis of text documents. Before introducing kernels for strings, let us introduce some notation. A ***string*** or ***sequence*** $\boldsymbol{x}$ is a concatenation of symbols from an *alphabet* $\mathcal{A}$, such as the characters in text or the bases of DNA. The set of all possible concatenations of symbols from $\mathcal{A}$ is denoted by $\mathcal{A}^*$ and the set of all concatenations of length n by $\mathcal{A}^n$

For characterizing the content of sequential data, most string kernels make use of a predefined set $L \subseteq \mathcal{A}^*$ of relevant strings. This set L can be interpreted as a language that is used to capture structure contained in strings and may range from a simple selection of interesting terms to complex constructs involving gaps and wildcards. We focus on two basic definitions that are widely used for learning with string kernels: *words* and ***n-grams*** (Fig. 30.8).

In the domain of information retrieval and natural language processing, the set L is often defined as *words* of a natural language, such as English or German. In this setting, L is either given explicitly by providing a dictionary of terms or implicitly by partitioning strings according to a set of delimiter symbols $\mathcal{D} \subset \mathcal{A}$, such that

$$L = (\mathcal{A} \setminus \mathcal{D})^*$$

where L corresponds to all concatenations of non-delimiter symbols. Based on this definition, the content of a string can be described in terms of contained words from the set L – a representation often denoted as a "***bag of words***" (Joachims 1999).

In several applications, however, the structure underlying sequential data is unknown and no set of appropriate words can be defined a priori. An alternative technique for defining the set L is to move a sliding window of length n over a string and to extract *n-grams* (substrings of length n, cf. Damashek 1995). Formally, this set can be defined as

$$L = A^n.$$

Using this definition of L, the content of a string can be described in terms of contained n-grams, even if no prior knowledge about its structure is available, for example as in many applications of bioinformatics involving DNA and protein sequences.

Based on the set L, a feature map Φ can be defined which embeds strings in an $|L|$-dimensional vector space spanned by the strings of L, that is,

$$\Phi : \mathcal{A}^* \to \mathbb{R}^{|L|}, \quad \Phi(\boldsymbol{x}) = \left(\#_w(\boldsymbol{x})\right)_{w \in L}, \tag{30.18}$$

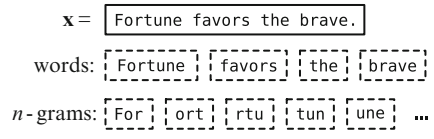


Fig. 30.8 Representations of a string $\boldsymbol{x}$ using words and n-grams

where $\#_w(\boldsymbol{x})$ returns the number of occurrences of the string w in the string $\boldsymbol{x}$. Alternatively, $\#_w(\boldsymbol{x})$ may be defined as frequency, probability or binary flag for the occurrences of w in $\boldsymbol{x}$.

This feature map Φ provides the basis for constructing a *string kernel* which takes the form

$$\mathrm{k} : \mathcal{A}^* \times \mathcal{A}^* \to \mathbb{R}, \quad \mathrm{k}(\boldsymbol{x}, z) = \sum_{w \in L} \#_w(\boldsymbol{x}) \cdot \#_w(z) \tag{30.19}$$

and corresponds to an inner product in the feature space spanned by the strings of L. Depending on the complexity of the set L, however, the dimension of this features space may grow almost arbitrarily. Thus in practice, computation of string kernels is rarely conducted using explicit vectors, but carried out by means of advanced data structures, such as hash maps, Tries and suffix trees (cf. Sonnenburg et al. 2007a). The corresponding realizations of (30.19) for words and n-grams are denoted as *Bag-of-Words Kernel* (Joachims 1999) and *Spectrum Kernel* (Leslie et al. 2002), respectively.

Due to the ease of incorporation with kernel-based learning methods, string kernels have gained considerable attention in research, starting from first realizations of Haussler (1999) and Watkins (2000), and extending to domain-specific kernels for natural language processing (Cancedda et al. 2003; Lodhi et al. 2002) and bioinformatics (Zien et al. 2000). In particular, the challenge of uncovering structure in DNA has influenced several extensions of the feature map in (30.18), for example by incorporating generative models (Jaakkola et al. 2000; Tsuda et al. 2002), inexact and position-dependent matching (Leslie and Kuang 2004; Leslie et al. 2003; Rätsch et al. 2005; Sonnenburg et al. 2006) as well as sequence alignments (Cuturi et al. 2007; Vert et al. 2004). A discussion of several string kernels and their implementations is provided by Sonnenburg et al. (2007a).

Tree Kernels

Besides sequences and strings, several applications of statistics and machine learning involve tree-structured data, for example in form of parse trees in natural language processing or molecule structures in chemistry. Formally, a ***tree*** $\boldsymbol{x}$ is an acyclic graph with a dedicated root, where we additionally require each *node* x to be labeled with a symbol. To navigate in a tree, we address the i-th child of a node x by x_i and denote the number of children by $|x|$. Moreover, we denote the set of all possible trees by T. Similar to strings, we construct a feature map Φ which embeds a tree $\boldsymbol{x}$ in a $|T|$-dimensional vector space spanned by all possible trees, that is

$$\Phi : T \to \mathbb{R}^{|T|}, \quad \Phi(\boldsymbol{x}) = \big(\#_t(\boldsymbol{x})\big)_{t \in T}, \tag{30.20}$$

where $\#_t(\boldsymbol{x})$ counts the occurrences of the (sub)tree t in the tree $\boldsymbol{x}$. In contrast to (30.18), the set T is more involved than a list of strings and thus requires special techniques for constructing a feasible kernel function.

A generic technique for defining kernels over structured data is the convolution of local kernels defined over sub-structures (Haussler 1999). This concept has been applied to tree data by Collins and Duffy (2002) for constructing a ***tree kernel*** which implicitly computes an inner product by counting shared subtrees. Given two trees $\boldsymbol{x}$ and $\boldsymbol{y}$, this kernel is defined as

$$\mathrm{k}(\boldsymbol{x},\boldsymbol{z}) = \langle \Phi(\boldsymbol{x})\Phi(\boldsymbol{z})\rangle = \sum_{x\in\boldsymbol{x}}\sum_{z\in\boldsymbol{z}} c(x,z) \tag{30.21}$$

where the counting function c recursively determines the number of **shared subtrees** rooted in the tree nodes x and z.

The function c is defined as $c(x,z) = 0$ if x and z have different labels and $c(x,z) = 1$ if x and z are leave nodes of the same label. In all other cases, the definition of c follows a recursive rule given by

$$c(x,z) = \prod_{i=1}^{|x|}(1 + c(x_i, z_i)). \tag{30.22}$$

To understand how the counting of subtrees relates to an inner product, let us consider two trees $\boldsymbol{x},\boldsymbol{z}$ and a subtree t, which occurs m times in $\boldsymbol{x}$ and n times in $\boldsymbol{z}$. Both trees share the subtree t and we can count mn distinct pairs of t common to $\boldsymbol{x}$ and $\boldsymbol{z}$. If we consider the feature map Φ given in (30.20), we have $\Phi_t(\boldsymbol{x}) = m$ and $\Phi_t(\boldsymbol{z}) = n$ and also obtain $\Phi_t(\boldsymbol{x})\Phi_t(\boldsymbol{z}) = mn$. Hence, by counting all shared subtrees of $\boldsymbol{x}$ and $\boldsymbol{z}$, we arrive at an inner product over the vectors $\Phi(\boldsymbol{x})$ and $\Phi(\boldsymbol{z})$. As an example, Fig. 30.9 illustrates two simple trees and their shared subtrees.

Several extensions and refinements of the kernel in (30.21) have been proposed to increase its expressiveness for specific learning tasks. For example, there exists several variations of (30.22) which account for different types of subtrees, such as complete trees (Vishwanathan and Smola 2004) and ordered trees (Kashima and Koyanagi 2002). A generic extension by Moschitti (2006) allows for controlling the vertical as well as the horizontal contribution of subtree counts. Furthermore, different techniques have been studied for alleviating the quadratic run-time of

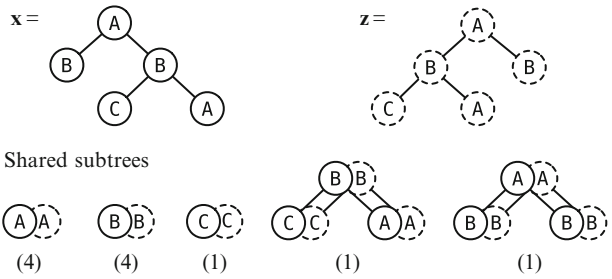


Fig. 30.9 Shared subtrees of two trees. The numbers in brackets indicate the number of occurrences for each shared subtree pair

counting shared subtrees, most notably the approximation framework of Rieck et al. (2010) which enables computing tree kernels in almost linear time.

30.4.4 Properties of Kernels

Besides being useful tools for constructing non-linear classifiers or learning with structured data, kernels possess some additional properties that make them an interesting choice in algorithms. It was shown (Girosi et al. 1993) that using a particular positive semi-definite kernel corresponds to an *implicit* choice of a regularization operator. For translation-invariant kernels, the regularization properties can be expressed conveniently in Fourier space in terms of the frequencies (Girosi 1998; Smola et al. 1998). For example, Gaussian kernels (cf. (30.16)) correspond to a general smoothness assumption in all k-th order derivatives (Smola et al. 1998). Vice versa, using this correspondence kernels matching a certain prior about the frequency content of the data can be constructed so as to reflect our prior problem knowledge.

Furthermore, many algorithms can be formulated using so called ***conditionally positive definite*** **kernels** (cf. Smola et al. 1998) which are a superclass of positive semi-definite kernels considered so far. They can be interpreted as generalized non-linear dissimilarity measures (opposed to just the scalar product) and are applicable e.g. in SVM and kernel PCA.

30.5 Implementation of SVM

30.5.1 Basic Formulations

We are now at the point to merge the ideas of statistical learning, structural risk minimization and reproducing kernels into a single algorithm, Support Vector Machines, suitable for a wide range of practical application. The main goal of this algorithm is to find a weight vector $\boldsymbol{w}$ separating the data $\mathcal{Z}$ with the largest possible margin.

Separable Data

Assume that the data are separable. Our goal is to find the smallest possible $\boldsymbol{w}$ without committing any error. This can be expressed by the following quadratic optimization problem:

$$\min_{\boldsymbol{w},b} \quad \frac{1}{2}\|\boldsymbol{w}\|^2, \tag{30.23}$$

$$\text{subject to} \quad y_i \left((\boldsymbol{w}^\top \boldsymbol{x}_i) + b \right) \geq 1, \quad \forall i = 1, \ldots, M.$$

The constraints in (30.23) assure that $\boldsymbol{w}$ and b are chosen such that no example has a distance to the hyperplane smaller than one. The problem can be solved directly by using a quadratic optimizer. Notice that the optimal solution renders a canonical hyperplane. In contrast to many neural networks (e.g. Bishop 1995) one can always find the *global* minimum. In fact, all minima of (30.23) are global minima, although they might not be unique as e.g. in the case when $M < N$, where N is the dimensionality of the data.

In the formulation (30.23), referred to as the **primal formulation**, we are bound to use the original data $\boldsymbol{x}_i$. In order to apply the **kernel trick** (cf. sect. 30.4.1) we need to transform the problem such that the only operation involving the original data $\boldsymbol{x}$ is an inner product between certain data points. This can be achieved by transforming the problem to the **dual formulation**. The notion of duality is an essential part of non-linear optimization theory, for details one can refer to any standard textbook on mathematical programming (e.g. Bertsekas 1995; Luenberger 1973). For our purposes it is important that for every quadratic optimization problem there exists a dual problem which is also a quadratic problem. If both the primal and the dual problems have an optimal solution then the values of the objective function at the optimal solutions coincide. This implies that by solving the dual problem – which uses the original data $\boldsymbol{x}$ only through inner products – the solution to the primal problem can be reconstructed.

To derive the dual of (30.23), we introduce **Lagrange multipliers** $\alpha_i \geq 0$, $i = 1, \ldots, M$, one for each of the constraints in (30.23). We obtain the following Lagrangian:

$$L(\boldsymbol{w}, b, \boldsymbol{\alpha}) = \frac{1}{2} \|\boldsymbol{w}\|^2 - \sum_{i=1}^{M} \alpha_i \left(y_i \left((\boldsymbol{w}^\top \boldsymbol{x}_i) + b \right) - 1 \right). \tag{30.24}$$

The task is to minimize (30.24) with respect to $\boldsymbol{w}, b$ and to maximize it with respect to α_i. At the optimal point, we have the following saddle point equations:

$$\frac{\partial L}{\partial b} = 0 \quad \text{and} \quad \frac{\partial L}{\partial \boldsymbol{w}} = 0,$$

which translate into

$$\sum_{i=1}^{M} \alpha_i y_i = 0 \quad \text{and} \quad \boldsymbol{w} = \sum_{i=1}^{M} \alpha_i y_i \boldsymbol{x}_i. \tag{30.25}$$

From the right equation of (30.25), we find that $\boldsymbol{w}$ is contained in the subspace spanned by the $\boldsymbol{x}_i$ in the training set. By substituting (30.25) into (30.24), we get the dual quadratic optimization problem:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha_j y_i y_j \left(\boldsymbol{x}_i^\top \boldsymbol{x}_j \right), \tag{30.26}$$

$$\text{subject to} \quad \alpha_i \geq 0, \; i = 1, \ldots, M, \tag{30.27}$$

$$\sum_{i=1}^{M} \alpha_i y_i = 0. \tag{30.28}$$

Thus, by solving the dual optimization problem, one obtains the coefficients α_i, $i = 1, \ldots, M$, which one needs to express the solution $\boldsymbol{w}$. This leads to the decision function:

$$\begin{aligned} f(\boldsymbol{x}) &= \operatorname{sgn}\left((\boldsymbol{w}^\top \boldsymbol{x}_i) + b \right) \\ &= \operatorname{sgn}\left(\sum_{i=1}^{M} y_i \alpha_i \left(\boldsymbol{x}_i^\top \boldsymbol{x} \right) + b \right). \end{aligned} \tag{30.29}$$

Note that the scalar product in this dual formulation can be directly replaced by the kernel mapping $\mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x})$, opening the possibility for the non-linear classifiers. This expression does not directly depend on the dimensionality N of the data but on the number of training examples M. As long as we are able to evaluate the scalar product $(\boldsymbol{x}_i^\top \boldsymbol{x})$ the dimensionality could be arbitrary, even infinite.

Non-separable Data

So far we have only considered the separable case which corresponds to an empirical error of zero (cf. Theorem 2). However for many practical applications this assumption is violated. If the data is not linearly separable then problem (30.23) has no feasible solution. By allowing for some errors we might get better results and avoid overfitting effects (cf. Fig. 30.2).

Therefore a "good" trade-off between the empirical risk and the complexity term in (30.9) needs to be found. Using a technique which was first proposed in (Bennett and Mangasarian 1992) and later used for SVMs in (Cortes and Vapnik 1995), one introduces **slack-variables** to relax the hard-margin constraints:

$$y_i((\boldsymbol{w}^\top \boldsymbol{x}_i) + b) \geq 1 - \xi_i, \;\; \xi_i \geq 0, \quad i = 1, \ldots, M, \tag{30.30}$$

additionally allowing for some classification errors. The SVM solution can then be found by (a) keeping the upper bound on the VC dimension small and (b) by minimizing an upper bound $\sum_{i=1}^{M} \xi_i$ on the empirical risk, i.e. the number of training errors. Thus, one minimizes

$$\min_{\boldsymbol{w}, b, \boldsymbol{\xi}} \quad \frac{1}{2} \|\boldsymbol{w}\|^2 + C \sum_{i=1}^{M} \xi_i,$$

where the regularization constant $C > 0$ determines the trade-off between the empirical error and the complexity term. This leads to the dual problem:

$$\max_{\boldsymbol{\alpha}} \sum_{i=1}^{M} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha_j y_i y_j \left(\boldsymbol{x}_i^\top \boldsymbol{x}_j \right), \tag{30.31}$$

$$\text{subject to} \quad 0 \leq \alpha_i \leq C, \; i = 1, \ldots, M, \tag{30.32}$$

$$\sum_{i=1}^{M} \alpha_i y_i = 0. \tag{30.33}$$

From introducing the slack-variables ξ_i, one gets the *box* constraints that limit the size of the Lagrange multipliers: $\alpha_i \leq C, i = 1, \ldots, M$.

The threshold b can be computed by exploiting the fact that for all support vectors $\boldsymbol{x}_i$ with $0 < \alpha_i < C$, the slack variable ξ_i is zero. This follows from the Karush-Kuhn-Tucker (KKT) conditions (cf. (30.34) below). Thus, for any support vector $\boldsymbol{x}_i$ with $\alpha_i < C$ holds:

$$y_i \left(\sum_{j=1}^{M} y_j \alpha_j \left(\boldsymbol{x}_i^\top \boldsymbol{x}_j \right) + b \right) = 1.$$

Averaging over these patterns I yields a numerically stable solution:

$$b = \frac{1}{|I|} \sum_{i \in I} \left(y_i - \sum_{j=1}^{M} y_j \alpha_j \left(\boldsymbol{x}_i^\top \boldsymbol{x}_j \right) \right).$$

Sparsity

The **Karush-Kuhn-Tucker** (KKT) conditions are the necessary conditions for an optimal solution of a non-linear programming problem (e.g. Bertsekas 1995; Luenberger 1973). The conditions are particularly simple for the dual SVM problem (30.31), (30.32) and (30.33) (Vapnik 1982):

$$\begin{aligned} \alpha_i = 0 &\Rightarrow \; y_i f(\boldsymbol{x}_i) \geq 1 \;\text{ and }\; \xi_i = 0, \\ 0 < \alpha_i < C &\Rightarrow \; y_i f(\boldsymbol{x}_i) = 1 \;\text{ and }\; \xi_i = 0, \\ \alpha_i = C &\Rightarrow \; y_i f(\boldsymbol{x}_i) \leq 1 \;\text{ and }\; \xi_i \geq 0. \end{aligned} \tag{30.34}$$

They reveal one of the most important properties of SVMs: the solution is sparse in $\boldsymbol{\alpha}$. For all examples outside the margin area the optimal α_i's are zero. Specifically, the **KKT conditions** show that only such α_i connected to a training pattern $\boldsymbol{x}_i$, which is either on the edge of (i.e. $0 < \alpha_i < C$ and $y_i f(\boldsymbol{x}_i) = 1$) or inside the margin area (i.e. $\alpha_i = C$ and $y_i f(\boldsymbol{x}_i) < 1$) are non-zero. These are exactly the *support vectors* as mentioned in Sect. 30.3.2.

30.5.2 Decomposition

The practical usefulness of SVM stems from their ability to provide arbitrary non-linear separation boundaries and at the same time to control generalization ability through the parameter C and the kernel parameters. In order to utilize these features it is necessary to work with the dual formulation (30.31)–(30.33) of the SVM training problem. This can be difficult from the computational point of view, for two reasons:

1. One needs to solve the quadratic programming problem with as many variables as the number M of available data points (this can be quite large, up to 10^5–10^6).
2. Merely to define the quadratic problem formally, one needs to store the $M \times M$ kernel matrix, which poses an insurmountable storage problem for large datasets.

Because of these implications, it is usually impossible to use the standard optimization tools (e.g. MINOS, CPLEX, LOQO) for solving the SVM training problems on datasets containing larger than 10,000 examples. In the following sections the decomposition techniques are presented, which use the special structure of the SVM problem to provide efficient training algorithms.

Basic Principles

The key idea of decomposition is to freeze all but a small number of optimization variables and to solve a sequence of constant-size problems. The set of variables optimized at a current iteration is referred to as the ***working set***. Because the working set is re-optimized, the value of the objective function is improved at each iteration provided the working set is not optimal before re-optimization.

The mathematics of the decomposition technique can be best seen in the matrix notation. Let $\boldsymbol{\alpha} = (\alpha_1, \dots \alpha_M)^\top$, let $\boldsymbol{y} = (y_1, \dots, y_M)^\top$, let H be the matrix with the entries $H_{ij} = y_i y_j \, \mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x}_j)$, and let $\mathbf{1}$ denote the vector of length M consisting of all 1s. Then the dual SVM problem (30.31)–(30.33) can be written in the matrix form as:

$$\max_{\boldsymbol{\alpha}} \quad \mathbf{1}^\top \boldsymbol{\alpha} - \frac{1}{2} \boldsymbol{\alpha}^\top H \boldsymbol{\alpha}, \tag{30.35}$$

$$\text{subject to} \quad \boldsymbol{y}^\top \boldsymbol{\alpha} = 0, \tag{30.36}$$

$$\boldsymbol{\alpha} - C\,\mathbf{1} \leq 0, \tag{30.37}$$

$$\boldsymbol{\alpha} \geq 0. \tag{30.38}$$

Let us partition the vector $\boldsymbol{\alpha}$ into $\boldsymbol{\alpha}_B$ of the variables to be included in the working set at a current iteration and the vector $\boldsymbol{\alpha}_N$ of the remaining variables. The matrix H is partitioned as

$$H = \begin{bmatrix} H_{BB} & H_{BN} \\ H_{NB} & H_{NN} \end{bmatrix},$$

with the corresponding parts determined by the index sets B and N. By re-writing the problem (30.35)–(30.38) using the partitioned matrix notation, and observing that only the free variables $\boldsymbol{\alpha}_B$ are to be considered as variables, the following sub-problem, to be solved at each iteration, is obtained:

$$\max_{\boldsymbol{\alpha}} \quad (\mathbf{1}_B^\top - \boldsymbol{\alpha}_N^\top H_{NB})\boldsymbol{\alpha}_B - \frac{1}{2}\boldsymbol{\alpha}_B^\top H_{BB}\boldsymbol{\alpha}_B, \tag{30.39}$$

$$\text{subject to} \quad \boldsymbol{y}_B^\top \boldsymbol{\alpha}_B = -\boldsymbol{y}_N \boldsymbol{\alpha}_N, \tag{30.40}$$

$$\boldsymbol{\alpha}_B - C\mathbf{1}_B \leq 0, \tag{30.41}$$

$$\boldsymbol{\alpha}_B \geq 0. \tag{30.42}$$

Choosing the size q of the working set B relatively small (usually $q \leq 100$) one can ensure that the sub-problem (30.39)–(30.42) is easily solved by any optimization tool.

Iteration of this procedure is carried out until the following termination criteria, derived from Karush-Kuhn-Tucker conditions (30.34), are satisfied to the required precision ϵ:

$$b - \epsilon \leq y_i - \sum_{j=1}^{M} y_j \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) \leq b + \epsilon, \qquad \forall i : \ 0 < \alpha_i < C, \tag{30.43}$$

$$y_i \left(\sum_{j=1}^{M} y_j \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) + b \right) \geq 1 - \epsilon, \qquad \forall i : \ \alpha_i = 0, \tag{30.44}$$

$$y_i \left(\sum_{j=1}^{M} y_j \alpha_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) + b \right) \leq 1 + \epsilon, \qquad \forall i : \ \alpha_i = C. \tag{30.45}$$

Working Set Selection: Feasible Direction Algorithms

The crucial issue in the decomposition technique presented above is the selection of working sets. First, the provision that a working set must be sub-optimal before re-optimization is essential to prevent the algorithm from cycling. Second, the working set selection affects the speed of the algorithm: if sub-optimal working sets are selected more or less at random, the algorithm converges very slowly. Finally, the working set selection is important in theoretical analysis of decomposition algorithms; in particular in the convergence proofs and in the analysis of the convergence rate.

Two main approaches exist to the working set selection in the SVM decomposition algorithms: the heuristic approach and the feasible direction approach. The former has been used in the original paper of Osuna et al. (1997a) on SVM decomposition and has been mainly used in the specific flavor of decomposition algorithms called **Sequential Minimal Optimization** (**SMO**), presented in the

next section. The feasible direction decomposition algorithms root in the SVM$^{\text{light}}$ algorithm of Joachims (1999) for pattern recognition, with the formal connection to the feasible direction methods of non-linear programming established by Laskov (2002).

The notion of a "feasible direction" stems from the classical techniques of non-linear programming subject to linear constraints (Bertsekas 1995; Zoutendijk 1960). It refers to the direction along which any step of the magnitude δ satisfying $0 < \delta \leq \delta^0$, for some fixed δ^0, results in a feasible solution to the non-linear program. For any non-linear program, finding the feasible direction amounts to a solution of a linear programming problem. In particular, for the dual SVM training problem (30.31)–(30.33) the following problem must be solved:

$$\max_{\boldsymbol{d}} \quad \boldsymbol{g}^\top \boldsymbol{d}, \tag{30.46}$$

$$\text{subject to} \quad \boldsymbol{y}^\top \boldsymbol{d} = 0, \tag{30.47}$$

$$d_i \geq 0, \quad \text{for all } \alpha_i = 0, \tag{30.48}$$

$$d_i \leq 0, \quad \text{for all } \alpha_i = C, \tag{30.49}$$

$$||\boldsymbol{d}||_2 \leq 1, \tag{30.50}$$

where $\boldsymbol{g}$ is the gradient of the objective function (30.31). Solving the feasible direction problem exactly at each iteration is inefficient because the linear program (30.46)–(30.50) has all M variables. However, an approximate solution to the feasible direction problem can be efficiently found by using the normalization $d_i \in \{-1, 0, 1\}$ instead of (30.50) and requiring that the number of positive direction components is equal to the number of the negative components. In this case, the solution is obtained by sorting all examples by the quantity $g_i y_i$, and selecting $q/2$ examples with the largest and $q/2$ examples with the smallest values. In fact, by using a Heap data structure, sorting can be avoided, and the entire selection can be executed in $O(q \log M)$ time. The motivation behind the quantity $g_i y_i$ can be traced back to the first-order Karush-Kuhn-Tucker conditions (Laskov 2002), which provides the solid formal background for the feasible direction SVM decomposition.

Convergence of the feasible direction SVM decomposition has been proved by Lin (2001), and the linear convergence rate has been observed experimentally (Laskov 2002).

Sequential Minimal Optimization

The **Sequential Minimal Optimization** (**SMO**) algorithm proposed by Platt (1999) is another popular and efficient algorithm for the SVM training. In this algorithm the idea of decomposition is taken to its extreme by reducing the working sets to two points – the smallest size for which it is possible to maintain the equality constraint (30.33). For two points the optimal solution can be computed analytically without calling an optimization tool.

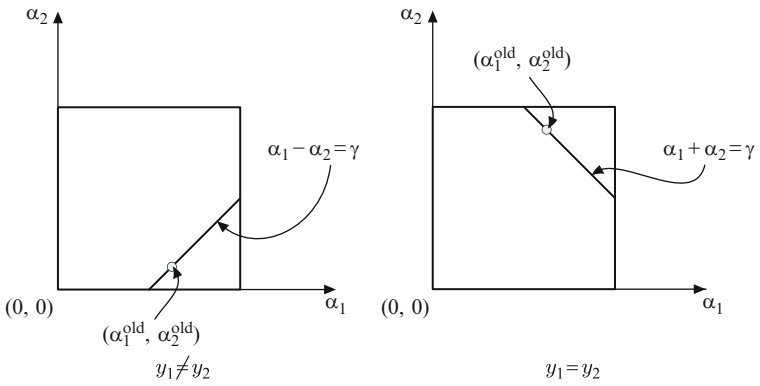


Fig. 30.10 Constraints of the SVM training problem with two examples

The analytical computation of the optimal solution is based on the following idea: given the solution $(\alpha_1^{\text{old}}, \alpha_2^{\text{old}})$, the optimal update is computed to obtain the solution $(\alpha_1^{\text{new}}, \alpha_2^{\text{new}})$. To carry out the update, first the constraints (30.32)–(30.33) have to be obeyed. The geometry of these constraints depends on whether the labels y_1 and y_2 are equal or not. The two possible configurations are shown in Fig. 30.10. If $y_1 \neq y_2$ (left picture) the solution should be sought along the line $\alpha_1 - \alpha_2 = \gamma$, where $\gamma = \alpha_1^{\text{old}} + y_1 y_2 \alpha_2^{\text{old}}$. If $y_1 = y_2$ (right picture) the solution should be sought along the line $\alpha_1 + \alpha_2 = \gamma$. If the solution transgresses the bound of the box, it should be clipped at the bound.

The optimal values of the variables along the line are computed by eliminating one of the variables through the equality constraint and finding the unconstrained minimum of the objective function, for example, eliminating α_1 one obtains the following update rule for α_2:

$$\alpha_2^{\text{new}} = \alpha_2^{\text{old}} - \frac{y_2(E_1 - E_2)}{\eta}, \tag{30.51}$$

where

$$E_1 = \sum_{j=1}^{M} y_j \alpha_j \, \mathrm{k}(\boldsymbol{x}_1, \boldsymbol{x}_j) + b - y_1, \tag{30.52}$$

$$E_2 = \sum_{j=1}^{M} y_j \alpha_j \, \mathrm{k}(\boldsymbol{x}_2, \boldsymbol{x}_j) + b - y_2, \tag{30.53}$$

$$\eta = 2\,\mathrm{k}(\boldsymbol{x}_1, \boldsymbol{x}_2) - \mathrm{k}(\boldsymbol{x}_1, \boldsymbol{x}_1) - \mathrm{k}(\boldsymbol{x}_2, \boldsymbol{x}_2). \tag{30.54}$$

Next, the bound constraints should be taken care of. Depending on the geometry, one computes the following lower and upper bounds on the value of the variable α_2:

$$L = \begin{cases} \max(0, \alpha_2^{\text{old}} - \alpha_1^{\text{old}}), & \text{if } y_1 \neq y_2, \\ \max(0, \alpha_2^{\text{old}} + \alpha_1^{\text{old}} - C), & \text{if } y_1 = y_2, \end{cases}$$

$$H = \begin{cases} \min(C, C + \alpha_2^{\text{old}} - \alpha_1^{\text{old}}), & \text{if } y_1 \neq y_2, \\ \min(C, \alpha_2^{\text{old}} + \alpha_1^{\text{old}}), & \text{if } y_1 = y_2. \end{cases}$$

The constrained optimum is then found by clipping the unconstrained optimum to the ends of the line segment:

$$\alpha_2^{\text{new}} := \begin{cases} H, & \text{if } \alpha_2^{\text{new}} \geq H, \\ L, & \text{if } \alpha_2^{\text{new}} \leq L, \\ \alpha_2^{\text{new}}, & \text{otherwise.} \end{cases}$$

Finally, the value of α_1^{new} is computed:

$$\alpha_1^{\text{new}} = \alpha_1^{\text{old}} + y_1 y_2 (\alpha_2^{\text{old}} - \alpha_2^{\text{new}}). \tag{30.55}$$

The working set selection in the SMO algorithm is carried out by means of two heuristics. The "first choice" heuristic is responsible for the choice of the first example in each pair. Following this heuristic, all examples that violate the KKT condition (30.34) are used in turns as the first example in the pair. More precisely, the algorithm makes repeated passes only through the examples whose α_i is strictly between then bounds, and only when all such examples satisfy the KKT conditions the sweep over the entire data is done to bring in new examples. The "second choice" heuristic is responsible for the selection of the second example in the pair. It is intended to bring in such an example that results in the largest step taken during the joint optimization (30.51). As a cheap approximation of this step the numerator $|E_1 - E_2|$ is taken (the denominator, which involves evaluation of kernels, can be expensive to compute). Following the strategy to maximize $|E_1 - E_2|$, the SMO algorithm chooses the example with the largest E_2, if E_1 is negative, and the example with the smallest E_2, if E_1 is positive. Under unusual circumstances, when no progress can be made using the second heuristic above, a hierarchy of weaker heuristics is applied, the details of which are provided by Platt (1999).

30.5.3 Incremental Support Vector Optimization

Many real-life machine learning problems can be more naturally viewed as online rather than batch learning problems. Indeed, the data is often collected continuously in time, and, more importantly, the concepts to be learned may also evolve in time. Significant effort has been spent in the recent years on the development of online

SVM learning algorithms (e.g. Kivinen et al. 2001; Ralaivola and d'Alché Buc 2001; Rüping 2002). An elegant solution to online SVM learning is the incremental SVM proposed by Cauwenberghs and Poggio (2001), which provides a framework for exact online learning.

The incremental SVM learning algorithm can be best presented using the following abstract form of the SVM optimization problem:

$$\max_{\mu} \min_{\substack{0\le\boldsymbol{\alpha}\le C \\ \boldsymbol{a}^\top\boldsymbol{\alpha}+b=0}} : W = -\boldsymbol{c}^\top\boldsymbol{\alpha} + \frac{1}{2}\boldsymbol{\alpha}^\top K\boldsymbol{\alpha} + \mu(\boldsymbol{a}^\top\boldsymbol{\alpha} + b), \tag{30.56}$$

where $\boldsymbol{c}$ and $\boldsymbol{a}$ are $M \times 1$ vectors, K is the $M \times M$ kernel matrix and b is a scalar. By defining the meaning of the abstract parameters $\boldsymbol{a}$, b and $\boldsymbol{c}$ for the particular SVM problem at hand, one can use the same algorithmic structure for different SVM algorithms. In particular, for the standard support vector classifiers (Vapnik 1998), take $\boldsymbol{c} = \mathbf{1}, \boldsymbol{a} = \boldsymbol{y}, b = 0$ and the given regularization constant C; the same definition applies to the ν-SVM (Schölkopf et al. 2000) except that $C = \frac{1}{N\nu}$.

The incremental SVM provides a procedure for adding one example to an existing optimal solution. When a new point k is added, its weight α_k is initially set to 0. Then the weights of other points and μ should be updated, in order to obtain the optimal solution for the enlarged dataset. Likewise, when a point k is to be removed from the dataset, its weight is forced to 0, while updating the weights of the remaining points and μ so that the solution obtained with $\alpha_k = 0$ is optimal for the reduced dataset. The online learning follows naturally from the incremental learning: the new example is added while some old example is removed from the working set.

The basic principle of the incremental SVM (Cauwenberghs and Poggio 2001) is that *updates to the state of the example k should keep the remaining examples in their optimal state*. In other words, the KKT conditions (30.34) expressed for the gradients g_i:

$$g_i = -c_i + K_{i,:}\boldsymbol{\alpha} + \mu a_i \begin{cases} \ge 0, & \text{if } \alpha_i = 0, \\ = 0, & \text{if } 0 < \alpha_i < C, \\ \le 0, & \text{if } \alpha_i = C, \end{cases} \tag{30.57}$$

$$\frac{\partial W}{\partial \mu} = \boldsymbol{a}^\top\boldsymbol{\alpha} + b = 0, \tag{30.58}$$

must be maintained for all the examples, except possibly for the current example k. Let S denote the set of unbounded support vectors and R the set of other examples. In order to maintain the KKT conditions (30.57), one can express increments to the weights of the unbounded support vectors and to the gradients of other examples as a linear function of the increment of the current example's weight $\Delta\alpha_k$:

$$\Delta\alpha_S = \beta\Delta\alpha_k, \quad \Delta g_R = \gamma\Delta\alpha_k. \tag{30.59}$$

The sensitivity relations (30.59) only make sense for the fixed composition of sets S and R. To proceed with learning, we need to detect the largest increment $\Delta\alpha_k^{\max}$ such that the sets S and R remain intact. After changing the SVM state according to the relations (30.59) evaluated for the increment $\Delta\alpha_k^{\max}$, the sensitivity vectors β and γ must be recomputed accordingly (depending of whether an example enters or leaves the set S). For the details of accounting the reader should refer to (Cauwenberghs and Poggio 2001; Tax and Laskov 2003). The iteration terminates when the current element satisfies the KKT conditions (30.57) as well.

30.5.4 Large-Scale Learning with SVM

The kernel trick has enabled SVMs to be applied to several domains and it has been one of the reasons SVMs are used with great success, often delivering state-of-the-art results. Unfortunately, it is also the kernel trick that limits the speed of applying SVMs and renders them unsuitable in several large-scale applications. The reason is that for predicting the class label of a single example, we need to compare it with all support vectors, that is, computing

$$f(\boldsymbol{x}) = \operatorname{sgn}\left(\sum_{i=1}^{M} \alpha_i \, y_i \, \mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x}) + b\right).$$

Consequently, the speed for application of an SVM decays linearly with the number of support vectors. It is this same operation that slows down training in most SVM implementations and potentially the cause for a shift in interest back to linear SVMs for large-scale applications. In the linear case, the decision of an SVM (without bias term b) takes the form

$$f(\boldsymbol{x}) = \operatorname{sgn}\left(\boldsymbol{w}^T \boldsymbol{x}\right)$$

and can be computed in linear time in the number of dimensions of the feature space, i.e. $\mathcal{O}(\dim(\mathcal{X}))$. Furthermore, if either the vectors $\boldsymbol{x}$ or $\boldsymbol{w}$ are sparse, this run-time can be further reduced to processing non-zero entries only.

The first linear SVM for large-scale application has been proposed Joachims (2006, SVM^{perf}) and builds on solving the underlying optimization problem using the concept of ***cutting planes*** (Kelly 1960). The resulting linear SVM achieves an ε-precise solution in time linear in the number of samples and dimensions, that is, $\mathcal{O}(M \dim(\mathcal{X}))$.

Theorem 4 (Joachims 2006). *For any $\varepsilon > 0, C > 0$, and any training sample $\mathcal{Z} = \{(\boldsymbol{x}_1, y_1), \ldots, (\boldsymbol{x}_M, y_M)\}$, SVM^{perf} terminates after at most*

$$\max\left\{\frac{2}{\varepsilon}, \frac{8\,C R^2}{\epsilon^2}\right\}$$

iterations, where $R = \max_{i=1}^{M} ||\boldsymbol{x}_i||$.

This method has been further refined by Teo et al. (2007, 2010) and later on by Franc and Sonnenburg (2008, 2009) who improve convergence rates to $\mathcal{O}(\frac{1}{\varepsilon})$. Following a different line of research, Fan et al. (2008) have developed a **dual coordinate descent** approach that performs similar, but drastically reduces memory requirements, as no set of cutting planes needs to be stored. For that reason we will explain this algorithm in more detail.

The idea of the dual coordinate descent method by Fan et al. (2008) is to perform coordinate descent on a single dual variable α_i while maintaining the SVM normal vector $\boldsymbol{w}$. Note that SMO (cf. Sect. 30.5.2) updates two variables at the same time, while coordinate descent updates a single variable only. More formally, considering the dual objective function $D(\boldsymbol{\alpha}) = \frac{1}{2}\sum_{i,j=1}^{M} \alpha_i \alpha_j y_i y_j \, \mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x}_j) - \sum_{i=1}^{M} \alpha_i$, one solves the dual optimization problem for a single variable α_i via

$$\begin{aligned} \min_{d} \quad & D(\boldsymbol{\alpha} + d\, \mathbf{1_i}) \\ \text{subject to} \quad & 0 \leq \alpha_i + d \leq C \end{aligned}$$

where $\mathbf{1_i}$ is the vector whose i-th component is 1 while its other components are zero, i.e. $\mathbf{1_i} = (0, \ldots, 0, 1, 0, \ldots, 0)^T$. By inserting $(\boldsymbol{\alpha} + d\, \mathbf{1_i})$ into D and removing terms independent of d, we arrive at

$$\begin{aligned} \min_{d} \quad & \frac{d^2}{2} \mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x}_i) + d \left(\sum_{j=1}^{M} y_i y_j \, \mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x}_j) \alpha_j - 1 \right) \\ \text{subject to} \quad & 0 \leq \alpha_i + d \leq C. \end{aligned} \tag{30.60}$$

If α_i is optimal, that is $d = 0$, the projected gradient is 0 (cf. PG in Algorithm 13) and we leave this coordinate unchanged. Otherwise, asserting $\mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x}_i) > 0$ one computes the new value of α_i by determining the derivative of (30.60) with respect to d. Setting the derivative to zero while ensuring $0 \leq \alpha_i \leq C$ finally leads to the following update rule

$$\alpha_i \leftarrow \min \left(\max \left(\alpha_i - \frac{\sum_{j=1}^{M} y_i y_j \, \mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x}_j) \alpha_j - 1}{\mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x}_i)}, 0 \right), C \right).$$

It should be noted that for efficiency the $\mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x}_i)$ in the denominator can be precomputed. In addition, we know from (30.25) that $\boldsymbol{w} = \sum_{i=1}^{M} \alpha_i y_i \boldsymbol{x}_i$ and subsequently the numerator can be written as $y_i \boldsymbol{w}^T \boldsymbol{x}_i - 1$. We avoid to recompute $\boldsymbol{w}$ in each iteration by starting with $\boldsymbol{\alpha} = \mathbf{0}$ and $\boldsymbol{w} = \mathbf{0}$, and only compute updates on $\boldsymbol{w}$ via $\boldsymbol{w} \leftarrow \boldsymbol{w} + (\alpha_i - \bar{\alpha}_i) y_i \boldsymbol{x}_i$. As a result, dual coordinate descent closely resembles a perceptron-style algorithm with a step size obtained via the SVM dual.

The full algorithm is outlined in Algorithm 13. It can be shown that dual coordinate descent reaches a ε-precise solution $D(\alpha) \leq D(\alpha^*) + \varepsilon$ in $\mathcal{O}(\log(\frac{1}{\varepsilon}))$ iterations with an iteration cost of $\mathcal{O}(M \dim(\mathcal{X}))$ (Fan et al. 2008).

Algorithm 13 Dual Coordinate Descent (Fan et al. 2008)

$\boldsymbol{\alpha} = \mathbf{0}$ and $\boldsymbol{w} = \mathbf{0}$
repeat
  for $i = 1, \ldots, M$ **do**
    $G = y_i \boldsymbol{w}^T \boldsymbol{x}_i - 1$

$$PG = \begin{cases} \min(G, 0) & \text{if } \alpha_i = 0 \\ \max(G, 0) & \text{if } \alpha_i = C \\ G & \text{if } 0 < \alpha_i < C \end{cases}$$

    if $|PG| \neq 0$ **then**
      $\bar{\alpha}_i \leftarrow \alpha_i$
      $\alpha_i \leftarrow \min(\max(\alpha_i - G/\bar{Q}_{ii}, 0), C)$
      $\boldsymbol{w} \leftarrow \boldsymbol{w} + (\alpha_i - \bar{\alpha}_i) y_i \boldsymbol{x}_i$
    end if
  end for
until Optimality

Besides **dual coordinate descent**, several other approaches for efficiently training linear SVMs have been proposed, for example, based on stochastic learning concepts (e.g. Bordes et al. 2009; Bottou and Bousquet 2008; Shwartz et al. 2007; Yu et al. 2010). Finally, there have been a number of attempts to combine the advantages of fast linear SVM solvers with the non-linearity of kernels (Chang et al. 2010; Joachims and Yu 2009; Sonnenburg and Franc 2010)

30.6 Extensions of SVM

30.6.1 Regression

In this subsection we will give a short overview of the idea of **Support Vector Regression** (SVR). A regression problem is given whenever $\mathcal{Y} = \mathbb{R}$ for the training dataset $\mathcal{Z} = \{(\boldsymbol{x}_i, y_i) \in \mathcal{X} \times \mathcal{Y} | i = 1, \ldots, M\}$ (cf. Sect. 30.2.1) and our interest is to find a function of the form $f : \mathcal{X} \to \mathbb{R}$.

In our discussion of statistical learning in Sect. 30.2.1 we have not talked about loss functions except for saying that they should be non-negative functions of the form (30.1). The particular form of the loss function depends on the learning task. For the pattern recognition problem the 0/1-loss function was used (cf. Theorem 2). For the regression problem the following two loss functions are common: the simple squared loss

$$\ell(f(\boldsymbol{x}), y) = (f(\boldsymbol{x}) - y)^2, \tag{30.61}$$

and the ϵ-insensitive loss

$$\ell(f(\boldsymbol{x}), y) = \begin{cases} |f(\boldsymbol{x}) - y| - \epsilon, & \text{if } |f(\boldsymbol{x}) - y| > \epsilon, \\ 0, & \text{otherwise.} \end{cases} \tag{30.62}$$

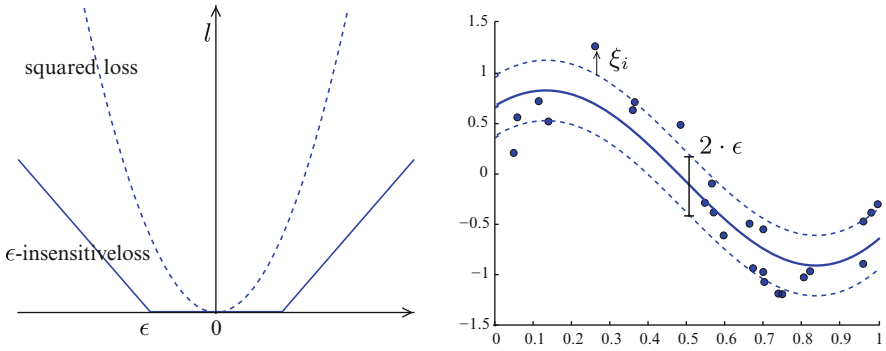


Fig. 30.11 The left subplot shows the two different loss functions for the regression problem. The right subplot gives a regression function derived with an ϵ-insensitive loss function. The solid line indicates the function, the dashed lines indicate the **ϵ-tube** around this function

For $\epsilon = 0$ the ϵ-insensitive loss equals the ℓ_1-norm, otherwise it linearly penalizes deviations from the correct predictions by more than ϵ.

In the left subplot of Fig. 30.11 the two error functions are shown. In the right subplot a regression function using the ϵ-insensitive loss function is shown for some artificial data. The dashed lines indicate the boundaries of the area where the loss is zero (the "tube"). Clearly most of the data is within the tube.

Similarly to the classification task, one is looking for the function that best describes the values y_i. In classification one is interested in the function that separates two classes; in contrast, in regression one looks for the function that contains the given dataset in its ϵ-tube. Some data points can be allowed to lie outside the ϵ-tube by introducing the slack-variables.

The primal formulation for the SVR is then given by:

$$\begin{aligned} \min_{\boldsymbol{w},b,\boldsymbol{\xi}^{(*)}} \quad & \frac{1}{2}\|\boldsymbol{w}\|^2 + C\sum_{i=1}^{M}(\xi_i + \xi_i^*), \\ \text{subject to} \quad & ((\boldsymbol{w}^\top \boldsymbol{x}_i) + b) - y_i \leq \epsilon + \xi_i, \\ & y_i - ((\boldsymbol{w}^\top \boldsymbol{x}_i) + b) \leq \epsilon + \xi_i^*, \\ & \xi_i^{(*)} \geq 0, \quad i = 1, \ldots, M. \end{aligned}$$

In contrast to the primal formulation for the classification task, we have to introduce two types of slack-variables ξ_i and ξ_i^*, one to control the error induced by observations that are larger than the upper bound of the ϵ-tube, and the other for the observations smaller than the lower bound. To enable the construction of a non-linear regression function, a dual formulation is obtained in the similar way to the classification SVM and the kernel trick is applied.

As a first application, SVR has been studied for analysis of time series by Müller et al. (1997). Further applications and an extensive description of SVR are provided by Vapnik (1998), Cristianini and Shawe-Taylor (2000), Schölkopf and Smola (2002).

30.6.2 One-Class Classification

Another common problem of statistical learning is one-class classification (novelty detection). Its fundamental difference from the standard classification problem is that the training data is not identically distributed to the test data. The dataset contains two classes: one of them, the target class, is well sampled, while the other class it absent or sampled very sparsely. Schölkopf et al. (2001) have proposed an approach in which the target class is separated from the origin by a hyperplane. Alternatively (Tax and Duin 2001), the target class can be modeled by fitting a **hypersphere** with minimal radius around it. We present this approach, illustrated in Fig. 30.12, in more detail below.

Mathematically the problem of fitting a hypersphere around the data is stated as:

$$\begin{aligned} \min_{R,\boldsymbol{a}} \quad & R^2 + C \sum_{i=0}^{M} \xi_i, \\ \text{subject to} \quad & ||\boldsymbol{x}_i - \boldsymbol{a}||^2 \leq R^2 + \xi_i, \; i = 1, \ldots, M, \\ & \xi_i \geq 0, \end{aligned} \tag{30.63}$$

where $\boldsymbol{a}$ is the center of the sphere, and R is the radius. Similarly to the SVM we make a "soft" fit by allowing non-negative slacks ξ_i. One can likewise apply the kernel trick by deriving the dual formulation of (30.63):

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & \sum_{i=1}^{M} \alpha_i \, \mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x}_i) - \frac{1}{2} \sum_{i,j=1}^{M} \alpha_i \alpha_j \, \mathrm{k}(\boldsymbol{x}_i, \boldsymbol{x}_j), \\ \text{subject to} \quad & 0 \leq \alpha_i \leq C, \; i = 1, \ldots, M, \end{aligned} \tag{30.64}$$

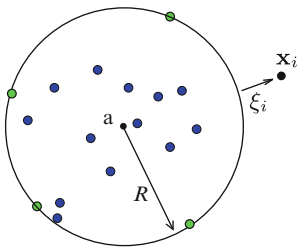


Fig. 30.12 Schematic illustration of the hypersphere model for describing a target class of data. The center $\boldsymbol{a}$ and the radius R should be optimized to minimize the volume of the captured space

$$\sum_{i=1}^{M} \alpha_i = 1.$$

The parameter C can be interpreted (Schölkopf et al. 2001) as the reciprocal of the quantity $\frac{1}{Mv}$, where v is an upper bound for the fraction of objects outside the boundary.

To decide whether a new object belongs to the target class one should determine its position with respect to the sphere using the formula

$$f(\boldsymbol{x}) = \text{sign}(R^2 - \text{k}(\boldsymbol{x}, \boldsymbol{x}) + 2 \sum_i \alpha_i \, \text{k}(\boldsymbol{x}, \boldsymbol{x}_i) - \sum_{i,j} \alpha_i \alpha_j \, \text{k}(\boldsymbol{x}_i, \boldsymbol{x}_j)). \quad (30.65)$$

An object with positive sign belongs to the target class and vice versa. An incremental version of the one-class classification SVM can be obtained using the approach of Sect. 30.5.3, with the parameters of the abstract formulation (30.56) defined as: $\boldsymbol{c} = \text{diag}(K)$, $\boldsymbol{a} = \boldsymbol{y}$ and $b = -1$ (Laskov et al. 2006).

30.7 Applications

30.7.1 Optical Character Recognition (OCR)

One of the first real-world experiments carried out with SVM was done at AT&T Bell Labs using optical character recognition (OCR) data (Cortes and Vapnik 1995; Schölkopf et al. 1995). These early experiments already showed the astonishingly high accuracies for SVMs which was on a par with convolutive multi-layer perceptrons. Below we list the classification performance of SVM, some variants not discussed in this chapter, and other classifiers on the USPS (US-Postal Service) benchmark (parts from Simard et al. 1998). The task is to classify handwritten digits into one of ten classes. Standard SVMs as presented here already exhibit a performance similar to other methods.

Linear PCA & Linear SVM (Schölkopf et al. 1998b)	8.7%
k-Nearest Neighbor	5.7%
LeNet1 (Bottou et al. 1994)	4.2%
Regularised RBF Networks (Rätsch 1998)	4.1%
Kernel-PCA & linear SVM (Schölkopf et al. 1998b)	4.0%
SVM (Schölkopf et al. 1995)	4.0%
Invariant SVM (Schölkopf et al. 1998a)	3.0%
Boosting (Drucker et al. 1993)	2.6%
Tangent Distance (Simard et al. 1998)	2.5%
Human error rate	2.5%

A benchmark problem larger than the USPS dataset (7,291 patterns) was collected by NIST and contains 120,000 handwritten digits. Invariant SVMs achieve an error rate of 0.7% (DeCoste and Schölkopf 2002) on this challenging and more realistic dataset, slightly better than the tangent distance method (1.1%) or single neural networks (LeNet 5: 0.9%). An ensemble of LeNet 4 networks that was trained on a vast number of artificially generated patterns (using invariance transformations) is on a par with the best SVM, and also achieved the accuracy of 0.7% (LeCun et al. 1995).

30.7.2 Text Categorization and Text Mining

The high dimensional problem of text categorization is an application for which SVMs have performed particularly well. A popular benchmark is the Reuters-22,173 text corpus containing 21,450 news stories, collected by Reuters in 1997, that are partitioned and indexed into 135 different categories. The features typically used to classify Reuters documents are 10,000-dimensional vectors containing word frequencies within a document. SVMs have achieved excellent results using such a coding (see e.g. Joachims 1997).

30.7.3 Network Intrusion Detection

The one-class formulation of the SVM presented in Sect. 30.6.2 provides another example for a successful application in computer security. In the last years, the amount and diversity of computer attacks has drastically increased, rendering regular defenses based on manually crafted detection rules almost ineffective. By contrast, the one-class SVM provides means for identifying unknown attacks automatically as novelties and thus has gained a strong focus in security research (e.g. Eskin et al. 2002; Nassar et al. 2008; Perdisci et al. 2009; Wahl et al. 2009).

As an example of this research, we present results from an application of the one-class SVM for detection of unknown attacks in network traffic of the FTP protocol (see Rieck 2009). Figure 30.13 shows the detection performance of the one-class SVM and the rule-based detection system "Snort". For analysing network data the one-class SVM is applied using a string kernel defined over n-grams as presented in Sect. 30.4.3. The one-class SVM significantly outperforms the regular detection system by identifying 80% of the unknown attacks with almost no false alarms, demonstrating the ability of SVMs to effectively generalize from data and surpass manually crafted rules and heuristics.

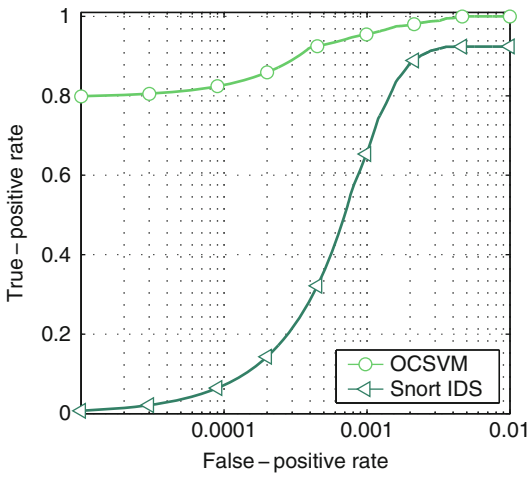


Fig. 30.13 One-class SVM vs. regular intrusion detection system (Rieck 2009)

30.7.4 Bioinformatics

SVMs are widely used in bioinformatics, defining the state-of-the-art in several applications, like splice site detection (Degroeve et al. 2005; Rätsch et al. 2007; Sonnenburg and Franc 2010; Sonnenburg et al. 2002; Sonnenburg et al. 2007b), detection of transcription starts (Sonnenburg et al. 2006), translation initiation site detection (Zien et al. 2000) and detection of several other genomic signals (Sonnenburg et al. 2008). Various further applications have been considered, like gene array expression monitoring (Brown et al. 2000), remote protein homology detection (Jaakkola et al. 2000; Leslie et al. 2002), protein sub-cellular localization Ong and Zien (2008), to detect protein protein interactions (Ben-Hur and Noble 2005) to analyze phylogenetic trees Vert (2002). For further information, the interested reader is referred to the tutorial on support vector machines and kernels (Ben-Hur et al. 2008).

30.7.5 Other Applications

There are numerous other applications to which SVM were successfully applied. Examples are object and face recognition tasks (Osuna et al. 1997b), inverse problems (Vapnik 1998; Weston et al. 1999), drug design in computational chemistry (Müller et al. 2005; Warmuth et al. 2003) and brain computer interfacing (Blankertz et al. 2007; Müller et al. 2008). A large collection of links to SVM applications is available at www.clopinet.com/isabelle/Projects/SVM/applist.html.

30.8 Summary and Outlook

We have shown in this chapter how the problem of learning from data can be cast formally into the problem of estimating functions from given observations. We have reviewed some basic notation and concepts from statistics and especially from statistical learning theory. The latter provides us with two extremely important insights: (1) what matters the most is not the dimensionality of the data but the complexity of the function class we choose our estimate from, (2) consistency plays an important role in successful learning. Closely related to these two questions is the issue of regularization. Regularization allows us to *control* the complexity of our learning machine and usually suffices to achieve consistency.

As an application of statistical learning theory we have presented the technique for constructing a maximum margin hyperplane. Whilst it is satisfactory to have a technique at hand that implements (at least partially) what the theory justifies, the algorithm is only capable of finding (linear) hyperplanes. To circumvent this restriction we introduced kernel functions yielding SVMs. Kernel functions allow us to reformulate many algorithms in some kernel feature space that is non-linearly related to the input space and yield powerful, non-linear techniques. Moreover, kernel functions enable us to apply learning techniques in structured domains, such as on string, trees and graphs. This abstraction using the kernel trick is possible whenever we are able to express an algorithm such that it only uses the data in the form of scalar products. However, since the algorithms are still linear in the feature space we can use the same theory and optimization strategies as before.

Kernel algorithms have seen an extensive development over the past years. Among many theoretical (Bartlett et al. 2002; Graepel et al. 2000; Williamson et al. 1998) and algorithmic (Joachims 1999; Platt 1999) results on SVM itself, new algorithms using the kernel trick have been proposed (e.g. Kernel PCA (Schölkopf et al. 1998b), Kernel ICA (Harmeling et al. 2003), temporal Kernel CCA (Bießmann et al. 2009) or Bayes–Point machines (Herbrich et al. 2001)). This development is still an ongoing and exciting field of study. A large collection of SVM implementations is contained in the shogun machine learning toolbox (Sonnenburg et al. 2010). Furthermore, various kernel-based learning methods are available from http://mloss.org/

References

Aizerman, M., Braverman, E., Rozonoer, L.: Theoretical foundations of the potential function method in pattern recognition learning. Autom. Remote Control **25**, 821–837 (1964)

Akaike, H.: A new look at the statistical model identification. IEEE Trans. Automat. Control **19**(6), 716–723 (1974)

Aronszajn, N.: Theory of reproducing kernels. Trans. Am. Math. Soc. **68**, 337–404 (1950)

Barron, A., Birgé, L., Massart, P.: Risk bounds for model selection via penalization. Probab. Theor. Relat. Fields **113**, 301–415 (1999)

Bartlett, P., Mendelson, S.: Rademacher and gaussian complexities: Risk bounds and structural results. J. Mach. Learn. Res. **3**, 463–482 (2002)

Bartlett, P., Long, P., Williamson, R.: Fat-shattering and the learnability of real-valued functions. J. Comput. Syst. Sci. **52**(3), 434–452 (1996)
Bartlett, P., Bousquet, O., Mendelson, S.: Localized rademacher complexities. In: Kivinen, J., Sloan, R. (eds.) Proceedings COLT, Lecture Notes in Computer Science, vol. 2375, pp. 44–58. Springer, Berlin (2002)
Ben-Hur, A., Noble, W.S.: Kernel methods for predicting protein-protein interactions. Bioinformatics, **21**(1), i38–i46 (2005)
Ben-Hur, A., Ong, C., Sonnenburg, S., Schölkopf, B., Rätsch, G.: Support vector machines and kernels for computational biology. PLoS Comput. Biol. **4**(10), e1000173 (2008)
Bennett, K., Mangasarian, O.: Robust linear programming discrimination of two linearly inseparable sets. Optim. Meth. Software **1**, 23–34 (1992)
Bertsekas, D.: Nonlinear Programming. Athena Scientific, Belmont, MA (1995)
Bießmann, F., Meinecke, F.C., Gretton, A., Rauch, A., Rainer, G., Logothetis, N., Müller, K.-R.: Temporal kernel canonical correlation analysis and its application in multimodal neuronal data analysis. Mach. Learn. **79**(1–2), 5—27 (2009); doi: 10.1007/s10994-009-5153-3. URL http://www.springerlink.com/content/e1425487365v2227.
Bishop, C.: Neural Networks for Pattern Recognition. Oxford University Press, Oxford (London/Melbourne) (1995)
Blankertz, B., Curio, G., Müller, K.-R.: Classifying single trial EEG: Towards brain computer interfacing. In: Diettrich, T.G., Becker, S., Ghahramani, Z. (eds.) Advances in Neural Inf. Proc. Systems (NIPS 01), vol. 14, pp. 157–164 (2002)
Blankertz, B., Dornhege, G., Krauledat, M., Müller, K.-R., Curio, G.: The non-invasive Berlin Brain-Computer Interface: Fast acquisition of effective performance in untrained subjects. NeuroImage **37**(2), 539–550 (2007); URL http://dx.doi.org/10.1016/j.neuroimage.2007.01.051.
Bordes, A., Bottou, L., Gallinari, P.: Sgd-qn: Careful quasi-newton stochastic gradient descent. JMLR **10** 1737–1754 (2009)
Boser, B., Guyon, I., Vapnik, V.: A training algorithm for optimal margin classifiers. In: Haussler, D. (eds.) Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory, pp. 144–152 (1992)
Bottou, L., Bousquet, O.: The tradeoffs of large scale learning. In NIPS 20. MIT Press, Cambridge, MA (2008)
Bottou, L., Cortes, C., Denker, J., Drucker, H., Guyon, I., Jackel, L., LeCun, Y., Müller, U., Säckinger, E., Simard, P., Vapnik, V.: Comparison of classifier methods: a case study in handwritten digit recognition. In Proceedings of the 12th International Conference on Pattern Recognition and Neural Networks, Jerusalem, pp. 77–87. IEEE Computer Society Press, Washington, DC, USA (1994)
Braun, M.L., Buhmann, J., Müller, K.-R.: On relevant dimensions in kernel feature spaces. J. Mach. Learn. Res. **9**, 1875–1908 (2008)
Breiman, L., Friedman, J., Olshen, J., Stone, C.: Classification and Regression Trees. Wadsworth, Belmont, CA (1984)
Brown, M., Grundy, W., Lin, D., Cristianini, N., Sugnet, C., Furey, T., Ares, M., Haussler, D.: Knowledge-based analysis of microarray gene expression data using support vector machines. Proc. Natl. Acad. Sci. **97**(1), 262–267 (2000)
Cancedda, N., Gaussier, E., Goutte, C., Renders, J.-M.: Word-sequence kernels. J. Mach. Learn. Res. **3**(Feb), 1059–1082 (2003)
Cauwenberghs, G., Poggio, T.: Incremental and decremental support vector machine learning. In: Leen, T., Diettrich, T., Tresp, V. (eds.) Advances in Neural Information Processing Systems 13, pp. 409–415 (2001)
Chang, Y.-W., Hsieh, C.-J., Chang, K.-W., Ringgaard, M., Lin, C.-J.: Training and testing low-degree polynomial data mappings via linear svm. JMLR **11**, 1471–1490 (2010)
Collins, M., Duffy, N.: Convolution kernel for natural language. In Advances in Neural Information Proccessing Systems (NIPS), vol. 16, pp. 625–632 (2002)
Cortes, C., Vapnik, V.: Support vector networks. Mach. Learn. **20**, 273–297 (1995)

Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines. Cambridge University Press, Cambridge, UK (2000)
Cuturi, M., Vert, J.-P., Matsui, T.: A kernel for time series based on global alignments. In Proceedings of the International Conferenc on Acoustics, Speech and Signal Processing (ICASSP), Honolulu, HI (2007)
Damashek, M.: Gauging similarity with n-grams: Language-independent categorization of text. Science **267**(5199), 843–848 (1995)
DeCoste, D., Schölkopf, B.: Training invariant support vector machines. Mach. Learn. **46**, 161–190 (2002)
Degroeve, S., Saeys, Y., Baets, B.D., Rouzé, P., de Peer, Y.V.: Splicemachine: predicting splice sites from high-dimensional local context representations. Bioinformatics **21**(8), 1332–1338 (2005)
Devroye, L., Györfi, L., Lugosi, G.: A Probabilistic Theory of Pattern Recognition. Number 31 in Applications of Mathematics. Springer, New York (1996)
Donoho, D., Johnstone, I., Kerkyacharian, G., Picard, D.: Density estimation by wavelet thresholding. Ann. Stat. **24**, 508–539 (1996)
Drucker, H., Schapire, R., Simard, P.: Boosting performance in neural networks. Intern. J. Pattern Recognit. Artif. Intell. **7**, 705–719 (1993)
Duda, R., Hart, P.E., Stork, D.G.: Pattern classification. (2nd edn.), Wiley, New York (2001)
Eskin, E., Arnold, A., Prerau, M., Portnoy, L., Stolfo, S.: Applications of Data Mining in Computer Security, chapter A geometric framework for unsupervised anomaly detection: detecting intrusions in unlabeled data. Kluwer, Dordecht (2002)
Fan, R., Chang, K., Hsieh, C., Wang, X., Lin, C.: LIBLINEAR: A library for large linear classification. JMLR **9**, 1871–1874 (2008)
Franc, V., Sonnenburg, S.: OCAS optimized cutting plane algorithm for support vector machines. In Proceedings of the 25nd International Machine Learning Conference. ACM Press, New York, NY, USA (2008); URL http://cmp.felk.cvut.cz/~xfrancv/ocas/html/index.html.
Franc, V., Sonnenburg, S.: Optimized cutting plane algorithm for large-scale risk minimization. J. Mach. Learn. Res. **10**(Oct), 2157–2192 (2009)
Freund, Y., Schapire, R.: A decision-theoretic generalization of on-line learning and an application to boosting. J. Comput. Syst. Sci. **55**(1), 119–139 (1997)
Gärtner, T., Lloyd, J., Flach, P.: Kernels and distances for structured data. Mach. Learn. **57**(3), 205–232 (2004)
Girosi, F.: An equivalence between sparse approximation and support vector machines. Neural Comput. **10**, 1455–1480 (1998)
Girosi, F., Jones, M., Poggio, T.: Priors, stabilizers and basis functions: From regularization to radial, tensor and additive splines. Technical Report A.I. Memo No. 1430, Massachusetts Institute of Technology (1993)
Graepel, T., Herbrich, R., Shawe-Taylor, J.: Generalization error bounds for sparse linear classifiers. In Proceedings of COLT, pp. 298–303, San Francisco, Morgan Kaufmann (2000)
Harmeling, S., Ziehe, A., Kawanabe, M., Müller, K.-R.: Kernel-based nonlinear blind source separation. Neural Comput. **15**, 1089–1124 (2003)
Haussler, D.: Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, UC Santa Cruz (1999)
Herbrich, R., Graepel, T., Campbell, C.: Bayes point machines. J. Mach. Learn. Res. **1**, 245–279 (2001)
Jaakkola, T., Diekhans, M., Haussler, D.: A discriminative framework for detecting remote protein homologies. J. Comp. Biol. **7**, 95–114 (2000)
Joachims, T.: Training linear SVMs in linear time. In International Conference on Knowledge Discovery and Data Mining (KDD), pp. 217–226 (2006)
Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. Technical Report 23, LS VIII, University of Dortmund (1997)
Joachims, T.: Making large–scale SVM learning practical. In: Schölkopf, B., Burges, C., Smola, A. (eds.) Advances in Kernel Methods – Support Vector Learning, pp. 169–184. MIT Press, Cambridge, MA (1999)

Joachims, T., Yu, C.-N.J.: Sparse kernel svms via cutting-plane training. Mach. Learn. **76**(2–3), 179–193 (2009)
Kashima, H., Koyanagi, T.: Kernels for semi-structured data. In International Conference on Machine Learning (ICML), pp. 291–298 (2002)
Kashima, H., Tsuda, K., Inokuchi, A.: Kernels for graphs. In Kernels and Bioinformatics, pp. 155–170. MIT press, Cambridge, MA (2004)
Kelly, J.: The cutting-plane method for solving convex programs. J. Soc. Ind. Appl. Math. **8**, 703–712 (1960)
Kivinen, J., Smola, A., Williamson, R.: Online learning with kernels. In: Diettrich, T.G., Becker, S., Ghahramani, Z. (eds.) Advances in Neural Inf. Proc. Systems (NIPS 01), pp. 785–792 (2001)
Kolmogorov, A.: Stationary sequences in hilbert spaces. Moscow Univ. Math. **2**, 1–40 (1941)
Laskov, P.: Feasible direction decomposition algorithms for training support vector machines. Mach. Learn. **46**, 315–349 (2002)
Laskov, P., Gehl, C., Krüger, S., Müller, K.R.: Incremental support vector learning: Analysis, implementation and applications. J. Mach. Learn. Res. **7**, 1909–1936 (2006)
LeCun, Y., Jackel, L., Bottou, L., Brunot, A., Cortes, C., Denker, J., Drucker, H., Guyon, I., Müller, U., Säckinger, E., Simard, P., Vapnik, V.: Comparison of learning algorithms for handwritten digit recognition. In: Fogelman-Soulié, F., Gallinari, P. (eds.) Proceedings ICANN'95 – International Conference on Artificial Neural Networks, vol. II, pp. 53–60. Nanterre, France (1995)
Leslie, C., Kuang, R.: Fast string kernels using inexact matching for protein sequences. J. Mach. Learn. Res. **5**, 1435–1455 (2004)
Leslie, C., Eskin, E., Noble, W.: The spectrum kernel: A string kernel for SVM protein classification. In Proceedings of Pacific Symposium on Biocomputing, pp. 564–575 (2002)
Leslie, C., Eskin, E., Cohen, A., Weston, J., Noble, W.: Mismatch string kernel for discriminative protein classification. Bioinformatics **1**(1), 1–10 (2003)
Lin, C.-J.: On the convergence of the decomposition method for support vector machines. IEEE Trans. Neural Networks **12**(6), 1288–1298 (2001)
Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C.: Text classification using string kernels. J. Mach. Learn. Res. **2**, 419–444 (2002)
Luenberger, D.: Introduction to Linear and Nonlinear Programming. Addison-Wesley, Reading, MA (1973)
Mallows, C.: Some comments on Cp. Technometrics **15**, 661–675 (1973)
Mercer, J.: Functions of positive and negative type and their connection with the theory of integral equations. Philos. Trans. Roy. Soc. London A **209**, 415–446 (1909)
Mika, S.: Kernel Fisher Discriminants. PhD thesis, Berlin Institute of Technology (2002)
Moody, J., Darken, C.: Fast learning in networks of locally-tuned processing units. Neural Comput. **1**(2), 281–294 (1989)
Morozov, V.: Methods for Solving Incorrectly Posed Problems. Springer, New York, NY (1984)
Moschitti, A.: Efficient convolution kernels for dependency and constituent syntactic trees. In European Conference on Machine Learning (ECML), pp. 318–329 (2006)
Müller, K.-R., Smola, A., Rätsch, G., Schölkopf, B., Kohlmorgen, J., Vapnik, V.: Predicting time series with support vector machines. In: Gerstner, W., Germond, A., Hasler, M., Nicoud, J.-D. (eds.) Artificial Neural Networks – ICANN '97, LNCS, vol. 1327, pp. 999–1004. Springer, Berlin (1997)
Müller, K.-R., Mika, S., Rätsch, G., Tsuda, K., Schölkopf, B.: An introduction to kernel-based learning algorithms. IEEE Neural Netw. **12**(2), 181–201 (2001)
Müller, K.-R., Rätsch, G., Sonnenburg, S., Mika, S., Grimm, M., Heinrich, N.: Classifying 'drug-likeness' with kernel-based learning methods. J. Chem. Inf. Model **45**, 249–253 (2005)
Müller, K.-R., Tangermann, M., Dornhege, G., Krauledat, M., Curio, G., Blankertz, B.: Machine learning for real-time single-trial EEG-analysis: From brain-computer interfacing to mental state monitoring. J. Neurosci. Meth. **167**(1), 82–90 (2008); URL http://dx.doi.org/10.1016/j.jneumeth.2007.09.022.

Nassar, M., State, R., Festor, O.: Monitoring SIP traffic using support vector machines. In Proceedings of Symposium on Recent Advances in Intrusion Detection, pp. 311–330 (2008)

Ong, C.S., Zien, A.: An automated combination of kernels for predicting protein subcellular localization. In Proceedings of the 8th Workshop on Algorithms in Bioinformatics (WABI), Lecture Notes in Bioinformatics, pp. 186–179. Springer, New York (2008)

Osuna, E., Freund, R., Girosi, F.: An improved training algorithm for support vector machines. In: Principe, J., Giles, L., Morgan, N., Wilson, E. (eds.) Neural Networks for Signal Processing VII – Proceedings of the 1997 IEEE Workshop, pp. 276–285. Springer, New York (1997a).

Osuna, E., Freund, R., Girosi, F.: Training support vector machines: An application to face detection. In Proceedings CVPR'97 (1997b)

Perdisci, R., Ariu, D., Fogla, P., Giacinto, G., Lee, W.: McPAD: A multiple classifier system for accurate payload-based anomaly detection. Computer Networks, pp. 864–881 (2009)

Platt, J.: Fast training of support vector machines using sequential minimal optimization. In: Schölkopf, B., Burges, C., Smola, A. (ed.): Advances in Kernel Methods – Support Vector Learning, pp. 185–208. MIT Press, Cambridge, MA (1999)

Ralaivola, L., d'Alché Buc, F.: Incremental support vector machine learning: A local approach. Lect. Notes Comput. Sci. **2130**, 322–329 (2001)

Rätsch, G.: Ensemble learning methods for classification. Master's thesis, Department of Computer Science, University of Potsdam, In German (1998)

Rätsch, G., Mika, S., Schölkopf, B., Müller, K.-R.: Constructing boosting algorithms from SVMs: an application to one-class classification. IEEE PAMI **24**(9), 1184–1199 (2002)

Rätsch, G., Sonnenburg, S., Schölkopf, B.: RASE: recognition of alternatively spliced exons in c. elegans. Bioinformatics **21**, i369–i377 (2005)

Rätsch, G., Sonnenburg, S., Srinivasan, J., Witte, H., Sommer, R., Müller, K.-R., Schölkopf, B.: Improving the c. elegans genome annotation using machine learning. PLoS Comput. Biol. **3**(2), e20 (2007)

Rieck, K.: Machine Learning for Application-Layer Intrusion Detection. PhD thesis, Berlin Institute of Technology, Berlin (2009)

Rieck, K., Krueger, T., Brefeld, U., Müller, K.-R.: Approximate tree kernels. J. Mach. Learn. Res. **11**(Feb), 555–580 (2010)

Rüping, S.: Incremental learning with support vector machines. Technical Report TR-18, Universität Dortmund, SFB475 (2002)

Schölkopf, B., Burges, C., Vapnik, V.: Extracting support data for a given task. In: Fayyad, U., Uthurusamy, R. (eds.) Proceedings, First International Conference on Knowledge Discovery & Data Mining. AAAI Press, Menlo Park, CA (1995)

Schölkopf, B., Smola, A.: Learning with Kernels. MIT Press, Cambridge, MA (2002)

Schölkopf, B., Simard, P., Smola, A., Vapnik, V.: Prior knowledge in support vector kernels. In: Jordan, M., Kearns, M., Solla, S. (eds) Advances in Neural Information Processing Systems, vol. 10, pp. 640–646. MIT Press, Cambridge, MA (1998a)

Schölkopf, B., Smola, A., Müller, K.-R.: Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput. **10**, 1299–1319 (1998b)

Schölkopf, B., Mika, S., Burges, C., Knirsch, P., Müller, K.-R., Rätsch, G., Smola, A.: Input space vs. feature space in kernel-based methods. IEEE Trans. Neural Netw. / A Publication of the IEEE Neural Netw. Council **10**(5), 1000–1017 (1999)

Schölkopf, B., Smola, A., Williamson, R., Bartlett, P.: New support vector algorithms. Neural Comput. **12**, 1207–1245 (2000)

Schölkopf, B., Platt, J., Shawe-Taylor, J., Smola, A., Williamson, R.: Estimating the support of a high-dimensional distribution. Neural Comput. **13**(7), 1443–1471 (2001)

Shawe-Taylor, J., Cristianini, N.: Kernel methods for pattern analysis. Cambridge University Press, Cambridge (London/New York) (2004)

Shawe-Taylor, J., Bartlett, P., Williamson, R.: Structural risk minimization over data-dependent hierachies. IEEE Trans. Inform. Theor. **44**(5), 1926–1940 (1998)

Shwartz, S.-S., Singer, Y., Srebro, N.: Pegasos: Primal estimated sub-gradient solver for svm. In ICML, pp. 807–814. ACM Press, New York (2007)

Simard, P., LeCun, Y., Denker, J., Victorri, B.: Transformation invariance in pattern recognition – tangent distance and tangent propagation. In: Orr, G., Müller, K.-R. (eds.) Neural Networks: Tricks of the Trade, vol. 1524, pp. 239–274. Springer LNCS (1998)

Smola, A., Schölkopf, B., Müller, K.-R.: The connection between regularization operators and support vector kernels. Neural Netw. **11**, 637–649 (1998)

Sonnenburg, S., Franc, V.: COFFIN: a computational framework for linear SVMs. In Proceedings of the 27th International Machine Learning Conference, Haifa (2010); (accepted).

Sonnenburg, S., Rätsch, G., Jagota, A., Müller, K.-R.: New methods for splice-site recognition. In: Dorronsoro, J. (eds.) Proceedings of International conference on artificial Neural Networks – ICANN'02, pp. 329–336. LNCS 2415, Springer, Berlin (2002)

Sonnenburg, S., Zien, A., Rätsch, G.: ARTS: Accurate Recognition of Transcription Starts in Human. Bioinformatics **22**(14), e472–480 (2006)

Sonnenburg, S., Rätsch, G., Rieck, K.: Large scale learning with string kernels. In: Bottou, L., Chapelle, O., DeCoste, D., Weston, J. (eds.) Large Scale Kernel Machines, pp. 73–103. MIT Press, Cambridge, MA (2007a).

Sonnenburg, S., Schweikert, G., Philips, P., Behr, J., Rätsch, G.: Accurate Splice Site Prediction. BMC Bioinformatics, Special Issue from NIPS workshop on New Problems and Methods in Computational Biology Whistler, Canada, 18 December 2006, **8**(Suppl. 10):S7 (2007b)

Sonnenburg, S., Zien, A., Philips, P., Rätsch, G.: POIMs: positional oligomer importance matrices – understanding support vector machine based signal detectors. Bioinformatics **24**(13), i6–i14 (2008)

Sonnenburg, S., Rätsch, G., Henschel, S., Widmer, C., Behr, J., Zien, A., de Bona, F., Binder, A., Gehl, C., Franc, V.: The SHOGUN machine learning toolbox. J. Mach. Learn. Res. **11**, 1799–1802 (2010); URL http://www.shogun-toolbox.org.

Tax, D., Duin, R.: Uniform object generation for optimizing one-class classifiers. J. Mach. Learn. Res. pp. 155–173 (2001)

Tax, D., Laskov, P.: Online SVM learning: from classification to data description and back. In: Molina, C. (eds.) Proc. NNSP, pp. 499–508 (2003)

Teo, C.H., Le, Q., Smola, A., Vishwanathan, S.: A scalable modular convex solver for regularized risk minimization. In KDD'07 (2007)

Teo, C.H., Vishwanthan, S., Smola, A. J., Le, Q.V.: Bundle methods for regularized risk minimization. J. Mach. Learn. Res. **11**(Jan), 311–365 (2010)

Tikhonov, A., Arsenin, V.: Solutions of Ill-posed Problems. In: Winston, W.H., Washington, DC (1977)

Tsuda, K., Kawanabe, M., Rätsch, G., Sonnenburg, S., Müller, K.-R.: A new discriminative kernel from probabilistic models. Neural Comput. **14**, 2397–2414 (2002)

Vapnik, V.: Estimation of Dependences Based on Empirical Data. Springer, Berlin (1982)

Vapnik, V.: Statistical Learning Theory. Wiley, New York (1998)

Vapnik, V., Chervonenkis, A.: The necessary and sufficient conditions for consistency in the empirical risk minimization method. Pattern Recogn. Image Anal. **1**(3), 283–305 (1991)

Vert, J.-P.: A tree kernel to analyze phylogenetic profiles. Bioinformatics **18**, S276–S284 (2002)

Vert, J.-P., Saigo, H., Akutsu, T.: Kernel Methods in Computational Biology, chapter Local alignment kernels for biological sequences, pp. 131–154. MIT Press, Cambridge, MA (2004)

Vishwanathan, S., Smola, A.: Fast kernels for string and tree matching. In: Tsuda, K., Schölkopf, B., Vert, J. (eds.) Kernels and Bioinformatics, pp. 113–130. MIT Press, Cambridge, MA (2004)

Vishwanathan, S.V.N., Schraudolph, N.N., Kondor, R., Borgwardt, K.M.: Graph kernels. J. Mach. Learn. Res. **11**(Apr) (2010)

Wahba, G.: Spline bases, regularization, and generalized cross-validation for solving approximation problems with large quantities of noisy data. In Proceedings of the International Conference on Approximation theory. Academic Press, Austin, Texas (1980)

Wahl, S., Rieck, K., Laskov, P., Domschitz, P., Müller, K.-R.: Securing IMS against novel threats. Bell Labs Technical J. **14**(1), 243–257 (2009)

Warmuth, M.K., Liao, J., Rätsch, G.M.M., Putta, S., Lemmem, C.: Support Vector Machines for active learning in the drug discovery process. J. Chem. Inform. Sci. **43**(2), 667–673 (2003)

Watkins, C.: Dynamic alignment kernels. In: Smola, A., Bartlett, P., Schölkopf, B., Schuurmans, D. (ed.): Advances in Large Margin Classifiers, pp. 39–50. MIT Press, Cambridge, MA (2000)

Weston, J., Gammerman, A., Stitson, M., Vapnik, V., Vovk, V., Watkins, C.: Support vector density estimation. In: Schölkopf, B., Burges, C., Smola, A. (eds.) Advances in Kernel Methods – Support Vector Learning, pp. 293–305. MIT Press, Cambridge, MA (1999)

Williamson, R., Smola, A., Schölkopf, B.: Generalization performance of regularization networks and support vector machines via entropy numbers of compact operators. NeuroCOLT Technical Report NC-TR-98-019, Royal Holloway College, University of London, UK (1998)

Yu, J., Vishwanathan, S., Gunter, S., Schraudolph, N.N.: A quasi-newton approach to nonsmooth convex optimization problems in machine learning. JMLR **11**, 1145–1200 (2010)

Zien, A., Rätsch, G., Mika, S., Schölkopf, B., Lengauer, T., Müller, K.-R.: Engineering support vector machine kernels that recognize translation initiation sites in DNA. BioInformatics **16**(9), 799–807 (2000)

Zoutendijk, G.: Methods of feasible directions. Elsevier, Amsterdam (1960)

Chapter 31
Learning Under Non-stationarity: Covariate Shift Adaptation by Importance Weighting

Masashi Sugiyama

31.1 Introduction

The goal of supervised learning is to infer an unknown input-output dependency from training samples, by which output values for unseen test input points can be predicted. When developing a method of supervised learning, it is commonly assumed that the input points in the training set and the input points used for testing follow the *same* probability distribution (Bishop 1995; Duda et al. 2001) Hastie et al. (2001); Schölkopf and Smola (2002); Vapnik (1998); Wahba (1990). However, this common assumption is not fulfilled, for example, when outside of the training region is extrapolated or when training input points are designed by an active learning (a.k.a. experimental design) algorithm Kanamori (2007); Kanamori and Shimodaira (2003); Sugiyama (2006); Sugiyama and Nakajima (2009); Wiens (2000). Situations where training and test input points follow different probability distributions but the conditional distributions of output values given input points are unchanged are called *covariate shift* Shimodaira (2000). In this chapter, we review recently proposed techniques for alleviating for the influence of covariate shift.

Under covariate shift, standard learning techniques such as maximum likelihood estimation are biased. It was shown that the bias caused by covariate shift can be asymptotically canceled by weighting the loss function according to the *importance* – the ratio of test and training input densities Quiñonero-Candela et al. (2009); Shimodaira (2000); Sugiyama et al. (2007); Sugiyama and Müller (2005); Sugiyama and Kawanabe (2011, to appear); Zadrozny (2004). Similarly, standard model selection criteria such as cross-validation Stone (1974); Wahba (1990) or Akaike's information criterion Akaike (1974) lose their unbiasedness under covariate shift. It was shown that proper unbiasedness can also be recovered by modifying the

M. Sugiyama (✉)
Tokyo Institute of Technology, Meguro-ku, Tokyo, Japan
e-mail: sugi@cs.titech.ac.jp

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_31,

methods based on importance weighting Shimodaira (2000); Sugiyama et al. (2007); Sugiyama and Müller (2005); Zadrozny (2004).

As explained above, the importance weight plays a central role in covariate shift adaptation. However, since the importance weight is unknown in practice, it should be estimated from data. A naive approach to this task is to first use kernel density estimation Härdle et al. (2004) for obtaining estimators of the training and test input densities, and then taking the ratio of the estimated densities. However, division by estimated quantities can magnify the estimation error, so directly estimating the importance weight in a single-shot process would be more preferable. Following this idea, various methods for directly estimating the importance have been explored Bickel et al. (2007); Cheng and Chu (2004); Ćwik and Mielniczuk (1989); Huang et al. (2007); Kanamori et al. (2009a); Qin (1998); Silverman (1978); Sugiyama et al. (2008). These direct estimation approaches have been demonstrated to be more accurate than the two-step density estimation approach.

Examples of successful real-world applications of covariate shift adaptation include brain-computer interface Sugiyama et al. (2007), robot control Akiyama et al. (2010); Hachiya et al. (2009, 2011), speaker identification Yamada et al. (2010a), age prediction from face images Ueki et al. (2011), wafer alignment in semiconductor exposure apparatus Sugiyama and Nakajima (2009), and natural language processing Tsuboi et al. (2009).

The rest of this chapter is organized as follows. In Sect. 31.2, the problem of supervised learning under covariate shift is mathematically formulated. In Sect. 31.3, various learning methods under covariate shift are introduced. In Sect. 31.4, the issue of model selection under covariate shift is addressed. In Sect. 31.5, methods of importance estimation are reviewed. Finally, we conclude in Sect. 31.6.

A more extensive description of covariate shift adaptation techniques is available in Sugiyama and Kawanabe (2011, to appear).

31.2 Formulation of Supervised Learning Under Covariate Shift

In this section, we formulate the supervised learning problem under covariate shift.

Let us consider the supervised learning problem of estimating an unknown input-output dependency from training samples. Let

$$\{(\boldsymbol{x}_i^{\mathrm{tr}}, y_i^{\mathrm{tr}}) \mid \boldsymbol{x}_i^{\mathrm{tr}} \in \mathcal{X} \subset \mathbb{R}^d, y_i^{\mathrm{tr}} \in \mathcal{Y} \subset \mathbb{R}\}_{i=1}^{n_{\mathrm{tr}}},$$

be the training samples. $\boldsymbol{x}_i^{\mathrm{tr}}$ is a training input point drawn from a probability distribution with density $p_{\mathrm{tr}}(\boldsymbol{x})$. y_i^{tr} is a training output value following a conditional probability distribution with conditional density $p(y|\boldsymbol{x} = \boldsymbol{x}_i^{\mathrm{tr}})$. $p(y|\boldsymbol{x})$ may be regarded as the sum of the true output $f(\boldsymbol{x})$ and noise ϵ:

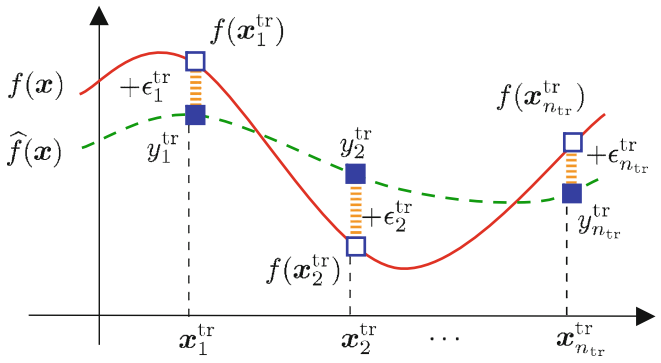


Fig. 31.1 Framework of supervised learning

$$y = f(\boldsymbol{x}) + \epsilon.$$

We assume that the noise ϵ has mean 0 and variance σ^2. This formulation is summarized in Fig. 31.1.

Let $(\boldsymbol{x}^{\mathrm{te}}, y^{\mathrm{te}})$ be a test sample, which is not given to the user in the training phase, but will be provided in the test phase in the future. $\boldsymbol{x}^{\mathrm{te}} \in \mathcal{X}$ is a test input point following a probability distribution with density $p_{\mathrm{te}}(\boldsymbol{x})$, which is different from $p_{\mathrm{tr}}(\boldsymbol{x})$. $y^{\mathrm{te}} \in \mathcal{Y}$ is a test output value following $p(y|\boldsymbol{x} = \boldsymbol{x}^{\mathrm{te}})$, which is the same conditional density as the training phase. The goal of supervised learning is to obtain an approximation $\hat{f}(\boldsymbol{x})$ to the true function $f(\boldsymbol{x})$ for predicting the test output value y^{te}. More formally, we would like to obtain the approximation $\hat{f}(\boldsymbol{x})$ that minimizes the test error expected over all test samples (which is called the *generalization error*):

$$G := \mathop{\mathbb{E}}_{\boldsymbol{x}^{\mathrm{te}}} \mathop{\mathbb{E}}_{y^{\mathrm{te}}} \left[\mathrm{loss}(\hat{f}(\boldsymbol{x}^{\mathrm{te}}), y^{\mathrm{te}}) \right],$$

where $\mathbb{E}_{\boldsymbol{x}^{\mathrm{te}}}$ denotes the expectation over $\boldsymbol{x}^{\mathrm{te}}$ drawn from $p_{\mathrm{te}}(\boldsymbol{x})$ and $\mathbb{E}_{y^{\mathrm{te}}}$ denotes the expectation over y^{te} drawn from $p(y|\boldsymbol{x} = \boldsymbol{x}^{\mathrm{te}})$. $\mathrm{loss}(\hat{y}, y)$ is the loss function which measures the discrepancy between the true output value y and its estimate $\hat{y}$. When the output domain $\mathcal{Y}$ is continuous, the problem is called *regression* and the *squared-loss* is often used.

$$\mathrm{loss}(\hat{y}, y) = (\hat{y} - y)^2.$$

On the other hand, when $\mathcal{Y} = \{+1, -1\}$, the problem is called (binary) *classification* and the *0/1-loss* is a typical choice.

$$\mathrm{loss}(\hat{y}, y) = \begin{cases} 0 \text{ if } \mathrm{sgn}(\hat{y}) = y, \\ 1 \text{ otherwise,} \end{cases}$$

where $\mathrm{sgn}(y) = +1$ if $y \geq 0$ and $\mathrm{sgn}(y) = -1$ if $y < 0$. Note that the 0/1-loss corresponds to the misclassification rate.

We use a parametric function $\hat{f}(\boldsymbol{x};\boldsymbol{\theta})$ for learning, where $\boldsymbol{\theta}$ is a parameter. A model $\hat{f}(\boldsymbol{x};\boldsymbol{\theta})$ is said to be *correctly specified* if there exists a parameter $\boldsymbol{\theta}^*$ such that $\hat{f}(\boldsymbol{x};\boldsymbol{\theta}^*) = f(\boldsymbol{x})$; otherwise the model is said to be *misspecified*. In practice, the model used for learning would be misspecified to a greater or less extent since we do not generally have enough prior knowledge for correctly specifying the model. Thus it is important to consider misspecified models when developing machine learning algorithms.

In standard supervised learning theories Bishop (1995); Duda et al. (2001); Hastie et al. (2001); Schölkopf and Smola (2002); Vapnik (1998); Wahba (1990), the test input point $\boldsymbol{x}^{\mathrm{te}}$ is assumed to follow the same distribution as the training input point $\boldsymbol{x}^{\mathrm{tr}}$. On the other hand, in this chapter, we consider the situation called *covariate shift* Shimodaira (2000), i.e., the training input point $\boldsymbol{x}^{\mathrm{tr}}$ and the test input point $\boldsymbol{x}^{\mathrm{te}}$ have different distributions. Under covariate shift, most of the standard learning techniques do not work well due to the differing distributions. Below, we review recently developed techniques for mitigating the influence of covariate shift.

31.3 Function Learning Under Covariate Shift

A standard method to learn the parameter $\boldsymbol{\theta}$ in the model $\hat{f}(\boldsymbol{x};\boldsymbol{\theta})$ would be *empirical risk minimization* (ERM) Schölkopf and Smola (2002); Vapnik (1998):

$$\hat{\boldsymbol{\theta}}_{\mathrm{ERM}} := \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \mathrm{loss}(\hat{f}(\boldsymbol{x}_i^{\mathrm{tr}};\boldsymbol{\theta}), y_i^{\mathrm{tr}}) \right].$$

If $p_{\mathrm{tr}}(\boldsymbol{x}) = p_{\mathrm{te}}(\boldsymbol{x})$, $\hat{\boldsymbol{\theta}}_{\mathrm{ERM}}$ converges to the optimal parameter $\boldsymbol{\theta}^*$ Shimodaira (2000):

$$\boldsymbol{\theta}^* := \operatorname*{argmin}_{\boldsymbol{\theta}}[G].$$

However, under covariate shift where $p_{\mathrm{tr}}(\boldsymbol{x}) \neq p_{\mathrm{te}}(\boldsymbol{x})$, $\hat{\boldsymbol{\theta}}_{\mathrm{ERM}}$ does not converge to $\boldsymbol{\theta}^*$ if the model is misspecified.[1]

In this section, we review various learning methods for covariate shift adaptation and show their numerical examples.

[1] $\hat{\boldsymbol{\theta}}_{\mathrm{ERM}}$ still converges to $\boldsymbol{\theta}^*$ under covariate shift if the model is correctly specified.

31.3.1 Importance Weighting Techniques for Covariate Shift Adaptation

Here, we introduce various regression and classification techniques for covariate shift adaptation.

Importance Weighted ERM

The inconsistency of ERM is due to the difference between training and test input distributions. *Importance sampling* Fishman (1996) is a standard technique to compensate for the difference of distributions. The following identity shows the essence of importance sampling:

$$\mathop{\mathbb{E}}_{\boldsymbol{x}^{\mathrm{te}}}[g(\boldsymbol{x}^{\mathrm{te}})] = \int g(\boldsymbol{x}) p_{\mathrm{te}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \int g(\boldsymbol{x}) \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})} p_{\mathrm{tr}}(\boldsymbol{x}) \mathrm{d}\boldsymbol{x} = \mathop{\mathbb{E}}_{\boldsymbol{x}^{\mathrm{tr}}}\left[g(\boldsymbol{x}^{\mathrm{tr}}) \frac{p_{\mathrm{te}}(\boldsymbol{x}^{\mathrm{tr}})}{p_{\mathrm{tr}}(\boldsymbol{x}^{\mathrm{tr}})} \right],$$

where $\mathbb{E}_{\boldsymbol{x}^{\mathrm{tr}}}$ and $\mathbb{E}_{\boldsymbol{x}^{\mathrm{te}}}$ denote the expectations over $\boldsymbol{x}^{\mathrm{tr}}$ and $\boldsymbol{x}^{\mathrm{te}}$ drawn from $p_{\mathrm{tr}}(\boldsymbol{x})$ and $p_{\mathrm{te}}(\boldsymbol{x})$, respectively. The quantity

$$\frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})}$$

is called the *importance*. The above identity shows that the expectation of a function $g(\boldsymbol{x})$ over $p_{\mathrm{te}}(\boldsymbol{x})$ can be computed by the importance-weighted expectation of $g(\boldsymbol{x})$ over $p_{\mathrm{tr}}(\boldsymbol{x})$. Thus the difference of distributions can be systematically adjusted by importance weighting.

Applying the above importance weighting technique to ERM, we obtain *importance-weighted ERM* (IWERM):

$$\hat{\boldsymbol{\theta}}_{\mathrm{IWERM}} := \mathop{\mathrm{argmin}}_{\boldsymbol{\theta}} \left[\frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \frac{p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}})}{p_{\mathrm{tr}}(\boldsymbol{x}_i^{\mathrm{tr}})} \mathrm{loss}(\hat{f}(\boldsymbol{x}_i^{\mathrm{tr}}; \boldsymbol{\theta}), y_i^{\mathrm{tr}}) \right].$$

$\hat{\boldsymbol{\theta}}_{\mathrm{IWERM}}$ converges to $\boldsymbol{\theta}^*$ under covariate shift, even if the model is misspecified Shimodaira (2000). In practice, IWERM may be *regularized*, e.g., by slightly flattening the importance weight and/or adding a penalty term as

$$\mathop{\mathrm{argmin}}_{\boldsymbol{\theta}} \left[\frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \left(\frac{p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}})}{p_{\mathrm{tr}}(\boldsymbol{x}_i^{\mathrm{tr}})} \right)^{\gamma} \mathrm{loss}(\hat{f}(\boldsymbol{x}_i^{\mathrm{tr}}; \boldsymbol{\theta}), y_i^{\mathrm{tr}}) + \lambda \boldsymbol{\theta}^{\top} \boldsymbol{\theta} \right],$$

where $0 \leq \gamma \leq 1$ is the flattening parameter, $\lambda \geq 0$ is the regularization parameter, and $^{\top}$ denotes the transpose of a matrix or a vector.

Importance-Weighted Regression Methods

Least-squares (LS) would be one of the most fundamental regression techniques. The importance-weighted regression method for the squared-loss (see Fig. 31.2), called *importance-weighted LS* (IWLS), is given as follows:

$$\hat{\boldsymbol{\theta}}_{\mathrm{IWLS}} := \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \left(\frac{p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}})}{p_{\mathrm{tr}}(\boldsymbol{x}_i^{\mathrm{tr}})} \right)^{\gamma} \left(\hat{f}(\boldsymbol{x}_i^{\mathrm{tr}}; \boldsymbol{\theta}) - y_i^{\mathrm{tr}} \right)^2 + \lambda \boldsymbol{\theta}^\top \boldsymbol{\theta} \right]. \quad (31.1)$$

Let us employ the following linear model:

$$\hat{f}(\boldsymbol{x}; \boldsymbol{\theta}) = \sum_{\ell=1}^{b} \boldsymbol{\theta}_\ell \phi_\ell(\boldsymbol{x}), \quad (31.2)$$

where $\{\phi_\ell(\boldsymbol{x})\}_{\ell=1}^b$ are fixed linearly-independent basis functions. Then the solution $\hat{\boldsymbol{\theta}}_{\mathrm{IWLS}}$ is given *analytically* as

$$\hat{\boldsymbol{\theta}}_{\mathrm{IWLS}} = (\boldsymbol{X}^{\mathrm{tr}\top} \boldsymbol{W}^{\gamma} \boldsymbol{X}^{\mathrm{tr}} + n_{\mathrm{tr}} \lambda \boldsymbol{I}_b)^{-1} \boldsymbol{X}^{\mathrm{tr}\top} \boldsymbol{W}^{\gamma} \boldsymbol{y}^{\mathrm{tr}}, \quad (31.3)$$

where $\boldsymbol{X}^{\mathrm{tr}}$ is the *design matrix*, i.e., $\boldsymbol{X}^{\mathrm{tr}}$ is the $n_{\mathrm{tr}} \times b$ matrix with the (i, ℓ)-th element $X_{i,\ell}^{\mathrm{tr}} = \phi_\ell(\boldsymbol{x}_i^{\mathrm{tr}})$, $\boldsymbol{W}$ is the diagonal matrix with the i-th diagonal element $\frac{p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}})}{p_{\mathrm{tr}}(\boldsymbol{x}_i^{\mathrm{tr}})}$, $\boldsymbol{I}_b$ is the b-dimensional identity matrix, and $\boldsymbol{y}^{\mathrm{tr}}$ is the n_{tr}-dimensional vector with the i-th element y_i^{tr}.

The LS method often suffers from excessive sensitivity to *outliers* (i.e., irregular values) and less reliability. A popular alternative is *importance-weighted least absolute* (IWLA) regression – instead of the squared loss, the absolute loss is used (see Fig. 31.2):

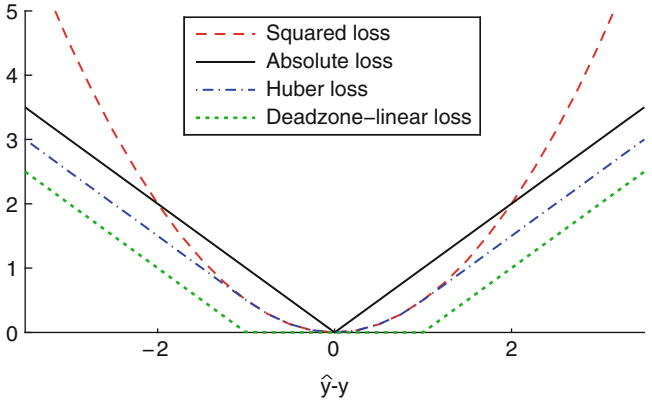


Fig. 31.2 Loss functions for regression. y is the true output value at an input point and $\hat{y}$ is its estimate

$$\hat{\boldsymbol{\theta}}_{\mathrm{IWLA}} = \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \left(\frac{p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}})}{p_{\mathrm{tr}}(\boldsymbol{x}_i^{\mathrm{tr}})} \right)^{\gamma} \left| \hat{f}(\boldsymbol{x}_i^{\mathrm{tr}}; \boldsymbol{\theta}) - y_i^{\mathrm{tr}} \right| + \lambda \boldsymbol{\theta}^{\top} \boldsymbol{\theta} \right].$$

For the linear model (31.2), the above optimization problem is reduced to a quadratic program, which can be solved by a standard optimization software. If the regularization term $\boldsymbol{\theta}^{\top}\boldsymbol{\theta}$ is replaced by the ℓ_1-regularizer $\sum_{\ell=1}^{b} |\boldsymbol{\theta}_\ell|$, the optimization problem is reduced to a linear program, which may be solved more efficiently. Furthermore, the ℓ_1-regularizer is known to induce a *sparse* solution Chen et al. (1998); Tibshirani (1996); Williams (1995).

Although the LA method is robust against outliers, it tends to have a large variance when the noise is Gaussian. The use of the *Huber loss* can mitigate this problem:

$$\hat{\boldsymbol{\theta}}_{\mathrm{Huber}} = \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \left(\frac{p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}})}{p_{\mathrm{tr}}(\boldsymbol{x}_i^{\mathrm{tr}})} \right)^{\gamma} \rho_{\tau} \left(\hat{f}(\boldsymbol{x}_i^{\mathrm{tr}}; \boldsymbol{\theta}) - y_i^{\mathrm{tr}} \right) + \lambda \boldsymbol{\theta}^{\top} \boldsymbol{\theta} \right],$$

where τ (≥ 0) is the robustness parameter and ρ_τ is the Huber loss defined as follows (see Fig. 31.2):

$$\rho_{\tau}(y) := \begin{cases} \frac{1}{2} y^2 & \text{if } |y| \leq \tau, \\ \tau |y| - \frac{1}{2}\tau^2 & \text{if } |y| > \tau. \end{cases}$$

Thus, the squared loss is applied to "good" samples with small fitting error, and the absolute loss is applied to "bad" samples with large fitting error. The above optimization problem can be reduced to a quadratic program Mangasarian and Musicant (2000), which can be solved by a standard optimization software.

Another variant of the IWLA is *importance-weighted support vector regression* (IWSVR):

$$\hat{\boldsymbol{\theta}}_{\mathrm{SVR}} = \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \left(\frac{p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}})}{p_{\mathrm{tr}}(\boldsymbol{x}_i^{\mathrm{tr}})} \right)^{\gamma} \left| \hat{f}(\boldsymbol{x}_i^{\mathrm{tr}}; \boldsymbol{\theta}) - y_i^{\mathrm{tr}} \right|_{\epsilon} + \lambda \boldsymbol{\theta}^{\top} \boldsymbol{\theta} \right],$$

where $|\cdot|_\epsilon$ is the *deadzone-linear loss* (or *Vapnik's ϵ-insensitive loss*) defined as follows (see Fig. 31.2):

$$|x|_{\epsilon} := \begin{cases} 0 & \text{if } |x| \leq \epsilon, \\ |x| - \epsilon & \text{if } |x| > \epsilon. \end{cases}$$

For the linear model (31.2), the above optimization problem is reduced to a quadratic program Vapnik (1998), which can be solved by a standard optimization software. If the regularization term $\boldsymbol{\theta}^{\top}\boldsymbol{\theta}$ is replaced by the ℓ_1-regularizer $\sum_{\ell=1}^{b} |\boldsymbol{\theta}_\ell|$, the optimization problem is reduced to a linear program.

Importance-Weighted Classification Methods

In the binary classification scenario where $\mathcal{Y} = \{+1, -1\}$, *Fisher discriminant analysis* (FDA) Fisher (1936), *logistic regression* (LR) Hastie et al. (2001), *support vector machine* (SVM) Schölkopf and Smola (2002); Vapnik (1998), and *boosting* Breiman (1998); Freund and Schapire (1996); Friedman et al. (2000) would be popular learning algorithms. They can be regarded as ERM methods with different loss functions (see Fig. 31.3).

An importance-weighted version of FDA, IWFDA, is given by

$$\hat{\boldsymbol{\theta}}_{\mathrm{IWFDA}} := \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \left(\frac{p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}})}{p_{\mathrm{tr}}(\boldsymbol{x}_i^{\mathrm{tr}})} \right)^{\gamma} \left(1 - y_i^{\mathrm{tr}} \hat{f}(\boldsymbol{x}_i^{\mathrm{tr}}; \boldsymbol{\theta}) \right)^2 + \lambda \boldsymbol{\theta}^\top \boldsymbol{\theta} \right],$$

which is essentially equivalent to (31.1) since $(y_i^{\mathrm{tr}})^2 = 1$.

An importance-weighted version of LR, IWLR, is given by

$$\hat{\boldsymbol{\theta}}_{\mathrm{IWLR}} := \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\sum_{i=1}^{n_{\mathrm{tr}}} \left(\frac{p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}})}{p_{\mathrm{tr}}(\boldsymbol{x}_i^{\mathrm{tr}})} \right)^{\gamma} \log \left(1 + \exp \left(-y_i^{\mathrm{tr}} \hat{f}(\boldsymbol{x}_i^{\mathrm{tr}}; \boldsymbol{\theta}) \right) \right) + \lambda \boldsymbol{\theta}^\top \boldsymbol{\theta} \right],$$

which is usually solved by (quasi-)Newton methods.

An importance-weighted version of SVM, IWSVM, is given by

$$\hat{\boldsymbol{\theta}}_{\mathrm{IWSVM}} := \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\sum_{i=1}^{n_{\mathrm{tr}}} \left(\frac{p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}})}{p_{\mathrm{tr}}(\boldsymbol{x}_i^{\mathrm{tr}})} \right)^{\gamma} \max \left(0, 1 - y_i^{\mathrm{tr}} \hat{f}(\boldsymbol{x}_i^{\mathrm{tr}}; \boldsymbol{\theta}) \right) + \lambda \boldsymbol{\theta}^\top \boldsymbol{\theta} \right],$$

whose solution can be obtained by a standard quadratic programming solver.

An importance-weighted version of Boosting, IWB, is given by

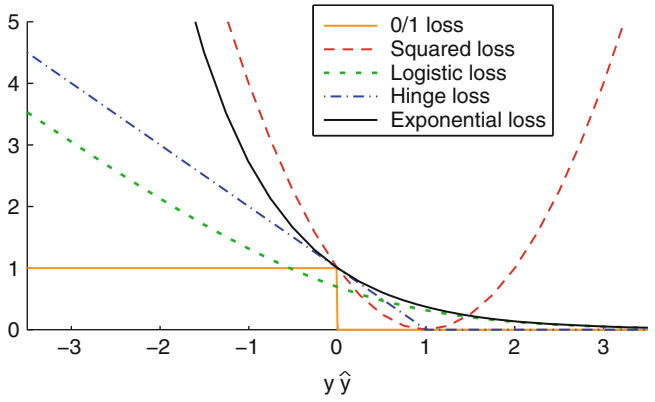


Fig. 31.3 Loss functions for classification. y is the true output value at an input point and $\hat{y}$ is its estimate

$$\hat{\boldsymbol{\theta}}_{\mathrm{IWB}} := \operatorname*{argmin}_{\boldsymbol{\theta}} \left[\sum_{i=1}^{n_{\mathrm{tr}}} \left(\frac{p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}})}{p_{\mathrm{tr}}(\boldsymbol{x}_i^{\mathrm{tr}})} \right)^{\gamma} \exp\left(-y_i^{\mathrm{tr}} \hat{f}(\boldsymbol{x}_i^{\mathrm{tr}}; \boldsymbol{\theta}) \right) + \lambda \boldsymbol{\theta}^{\top} \boldsymbol{\theta} \right],$$

which is usually solved by stage-wise optimization.

31.3.2 Numerical Examples

Here we illustrate the behavior of IWERM using toy regression and classification data sets.

Regression

Let us consider one-dimensional regression problem. Let the learning target function be $f(x) = \mathrm{sinc}(x)$, and let the training and test input densities be

$$p_{\mathrm{tr}}(x) = N(x; 1, (1/2)^2) \quad \text{and} \quad p_{\mathrm{te}}(x) = N(x; 2, (1/4)^2),$$

where $N(x; \mu, \sigma^2)$ denotes the Gaussian density with mean μ and variance σ^2. As illustrated in Fig. 31.5, we are considering a (weak) extrapolation problem since the training input points are distributed in the left-hand side of the input domain and the test input points are distributed in the right-hand side.

We create the training output value $\{y_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ as $y_i^{\mathrm{tr}} = f(x_i^{\mathrm{tr}}) + \epsilon_i^{\mathrm{tr}}$, where $\{\epsilon_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ are i.i.d. noise drawn from $N(\epsilon; 0, (1/4)^2)$. Let the number of training samples be $n_{\mathrm{tr}} = 150$, and we use the following linear model:

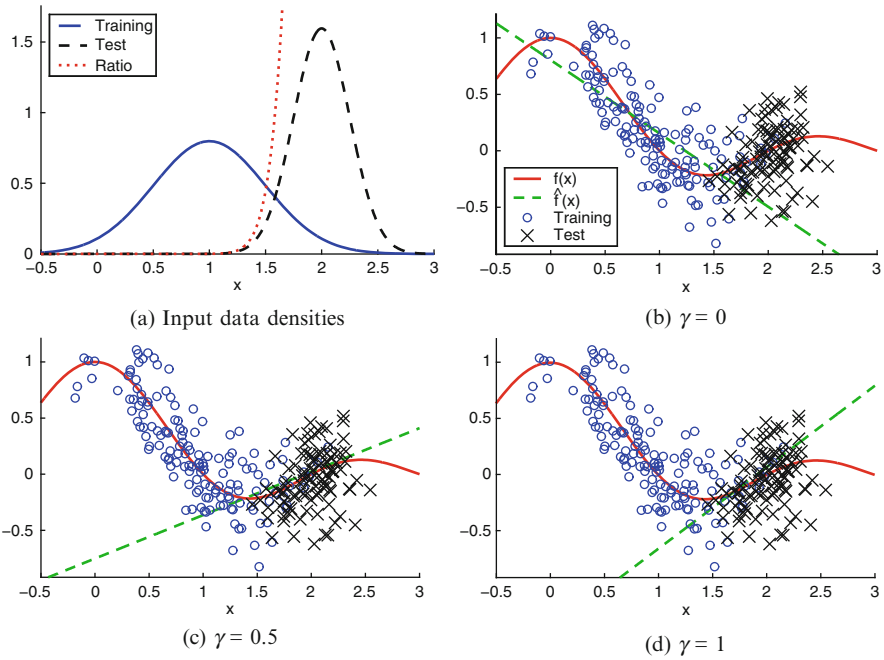
$$\hat{f}(x; \boldsymbol{\theta}) = \boldsymbol{\theta}_1 x + \boldsymbol{\theta}_2.$$

The parameter $\boldsymbol{\theta}$ is learned by IWLS.

Here we fix the regularization parameter to $\lambda = 0$, and compare the performance of IWLS for $\gamma = 0, 0.5, 1$. When $\gamma = 0$, a good approximation of the left-hand side of the sinc function can be obtained (see Fig. 31.4). However, this is not appropriate for estimating the test output values ("×" in the figure). Thus, IWLS with $\gamma = 0$ (i.e., ordinary LS) results in a large test error. Figure 31.4 depicts the learned function when $\gamma = 1$, which tends to approximate the test output values well, but having a large variance. Figure 31.4 depicts a learned function when $\gamma = 0.5$, which yields even better estimation of the test output values for this particular data realization.

Classification

Let us consider a binary classification problem on the two-dimensional input space. Let the class posterior probabilities given input $\boldsymbol{x}$ be



(a) Input data densities (b) $\gamma = 0$

(c) $\gamma = 0.5$ (d) $\gamma = 1$

Fig. 31.4 An illustrative regression example with covariate shift. (**a**) The probability density functions of the training and test input points and their ratio (i.e., the importance). (**b**–**d**) The learning target function $f(x)$ (the solid line), training samples ("∘"), a learned function $\hat{f}(x)$ (the dashed line), and test samples ("×")

$$p(y = +1 \,|\, \boldsymbol{x}) = \frac{1}{2}\left(1 + \tanh\left(x^{(1)} + \min(0, x^{(2)})\right)\right), \tag{31.4}$$

where $\boldsymbol{x} = (x^{(1)}, x^{(2)})^\top$ and $p(y = -1 \,|\, \boldsymbol{x}) = 1 - p(y = +1 \,|\, \boldsymbol{x})$. The optimal decision boundary, i.e., a set of all $\boldsymbol{x}$ such that $p(y = +1 \,|\, \boldsymbol{x}) = p(y = -1 \,|\, \boldsymbol{x}) = 1/2$ is illustrated in Fig. 31.5.

Let the training and test input densities be

$$p_{\mathrm{tr}}(\boldsymbol{x}) = \frac{1}{2} N\left(\boldsymbol{x}; \begin{bmatrix} -2 \\ 3 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}\right) + \frac{1}{2} N\left(\boldsymbol{x}; \begin{bmatrix} 2 \\ 3 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 4 \end{bmatrix}\right),$$

$$p_{\mathrm{te}}(\boldsymbol{x}) = \frac{1}{2} N\left(\boldsymbol{x}; \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right) + \frac{1}{2} N\left(\boldsymbol{x}; \begin{bmatrix} 4 \\ -1 \end{bmatrix}, \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}\right),$$

where $N(\boldsymbol{x}; \boldsymbol{\mu}, \boldsymbol{\Sigma})$ is the multivariate Gaussian density with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$. This setup implies that we are considering a (weak) extrapolation problem. Contours of the training and test input densities are illustrated in Fig. 31.5.

Let the number of training samples be $n_{\mathrm{tr}} = 500$, and we create training input points $\{\boldsymbol{x}_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ following $p_{\mathrm{tr}}(\boldsymbol{x})$ and training output labels $\{y_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ following $p(y|\boldsymbol{x} = \boldsymbol{x}_i^{\mathrm{tr}})$. Similarly, let the number of test samples be $n_{\mathrm{te}} = 500$, and we create n_{te} test input points $\{\boldsymbol{x}_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$ following $p_{\mathrm{te}}(\boldsymbol{x})$ and test output labels $\{y_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$ following $p(y|\boldsymbol{x} = \boldsymbol{x}_j^{\mathrm{te}})$. We use the following linear model:

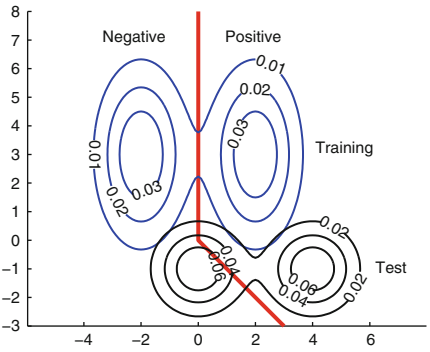
$$\hat{f}(\boldsymbol{x};\boldsymbol{\theta}) = \boldsymbol{\theta}_1 x^{(1)} + \boldsymbol{\theta}_2 x^{(2)} + \boldsymbol{\theta}_3.$$

The parameter $\boldsymbol{\theta}$ is learned by IWFDA.

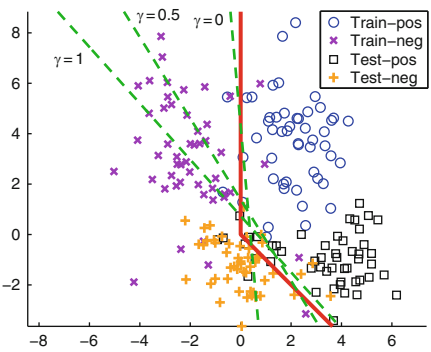
Here we fix the regularization parameter to $\lambda = 0$, and compare the performance of IWFDA for $\gamma = 0, 0.5, 1$. Figure 31.5 depicts an example of realizations of training and test samples, and decision boundaries obtained by IWFDA. For this particular realization of data samples, $\gamma = 0.5$ or 1 works better than $\gamma = 0$.

31.4 Model Selection Under Covariate Shift

As illustrated in the previous section, importance-weighting is a promising approach to covariate shift adaptation, given that the flattening parameter γ is chosen appropriately. Although $\gamma = 0.5$ worked well both for the toy regression and classification experiments in the previous section, $\gamma = 0.5$ may not always be the best choice. Indeed, an appropriate value of γ depends on the learning target function, models, the noise level in the training samples, etc. Therefore, *model selection* needs to be appropriately carried out for enhancing the generalization capability under covariate shift.



(a) Optimal decision boundary (the thick solid line) and contours of training and test input densities (thin solid lines).



(b) Optimal decision boundary (solid line) and learned boundaries (dashed lines). '○' and '×' denote the positive and negative training samples, while '□' and '+' denote the positive and negative test samples.

Fig. 31.5 An illustrative classification example with covariate shift

The goal of model selection is to determine the model (e.g, basis functions, the flattening parameter γ, and the regularization parameter λ) so that the generalization error is minimized Akaike (1970, 1974, 1980); Craven and Wahba (1979); Efron and Tibshirani (1993); Ishiguro et al. (1997); Konishi and Kitagawa (1996); Mallows (1973); Murata et al. (1994); Rissanen (1978, 1987); Schwarz (1978); Shibata (1989); Sugiyama et al. (2004); Sugiyama and Müller (2002); Sugiyama and Ogawa (2001); Takeuchi (1976); Vapnik (1998); Wahba (1990). The true generalization error is not accessible since it contains the unknown learning target function. Thus, some generalization error estimator needs to be used instead. However, standard generalization error estimators such as *cross-validation* (CV) are heavily biased under covariate shift, and therefore they are no longer reliable. In this section, we review a modified CV method that possesses proper unbiasedness even under covariate shift.

31.4.1 Importance-Weighted Cross-Validation

One of the popular techniques for estimating the generalization error is CV Stone (1974); Wahba (1990). CV has been shown to give an *almost* unbiased estimate of the generalization error with finite samples Luntz and Brailovsky (1969); Schölkopf and Smola (2002). However, such almost unbiasedness is no longer fulfilled under covariate shift.

To cope with this problem, a variant of CV called *importance-weighted CV* (IWCV) has been proposed Sugiyama et al. (2007). Let us randomly divide the training set $\mathcal{Z} = \{(\boldsymbol{x}_i^{\mathrm{tr}}, y_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}}$ into k disjoint non-empty subsets $\{\mathcal{Z}_i\}_{i=1}^{k}$ of (approximately) the same size. Let $\hat{f}_{\mathcal{Z}_i}(\boldsymbol{x})$ be a function learned from $\{\mathcal{Z}_{i'}\}_{i' \neq i}$ (i.e., without $\mathcal{Z}_i$). Then the *k-fold IWCV* (kIWCV) estimate of the generalization error G is given by

$$\hat{G}_{k\mathrm{IWCV}} = \frac{1}{k} \sum_{i=1}^{k} \frac{1}{|\mathcal{Z}_i|} \sum_{(\boldsymbol{x},y) \in \mathcal{Z}_i} \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})} \mathrm{loss}(\hat{f}_{\mathcal{Z}_i}(\boldsymbol{x}), y),$$

where $|\mathcal{Z}_i|$ is the number of samples in the subset $\mathcal{Z}_i$.

When $k = n_{\mathrm{tr}}$, kIWCV is particularly called *IW leave-one-out CV* (IWLOOCV):

$$\hat{G}_{\mathrm{IWLOOCV}} = \frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} \frac{p_{\mathrm{te}}(\boldsymbol{x}_i^{\mathrm{tr}})}{p_{\mathrm{tr}}(\boldsymbol{x}_i^{\mathrm{tr}})} \mathrm{loss}(\hat{f}_i(\boldsymbol{x}_i^{\mathrm{tr}}), y_i^{\mathrm{tr}}),$$

where $\hat{f}_i(\boldsymbol{x})$ is a function learned from $\{(\boldsymbol{x}_{i'}^{\mathrm{tr}}, y_{i'}^{\mathrm{tr}})\}_{i' \neq i}$ (i.e., without $(\boldsymbol{x}_i^{\mathrm{tr}}, y_i^{\mathrm{tr}})$). It was proved that IWLOOCV gives an *almost* unbiased estimate of the generalization error even under covariate shift Sugiyama et al. (2007). More precisely, IWLOOCV for n_{tr} training samples gives an unbiased estimate of the generalization error for $n_{\mathrm{tr}} - 1$ training samples:

$$\mathop{\mathbb{E}}_{\{\boldsymbol{x}_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}} \mathop{\mathbb{E}}_{\{y_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}} \left[\hat{G}_{\mathrm{IWLOOCV}}\right] = \mathop{\mathbb{E}}_{\{\boldsymbol{x}_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}} \mathop{\mathbb{E}}_{\{y_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}} [G'] \approx \mathop{\mathbb{E}}_{\{\boldsymbol{x}_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}} \mathop{\mathbb{E}}_{\{y_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}} [G],$$

where $\mathbb{E}_{\{\boldsymbol{x}_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}}$ denotes the expectation over $\{\boldsymbol{x}_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ drawn i.i.d. from $p_{\mathrm{tr}}(\boldsymbol{x})$, $\mathbb{E}_{\{y_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}}$ denotes the expectation over $\{y_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ each drawn from $p(y|\boldsymbol{x} = \boldsymbol{x}_i^{\mathrm{tr}})$, and G' denotes the generalization error for $n_{\mathrm{tr}} - 1$ training samples. A similar proof is also possible for kIWCV, but the bias is slightly larger Hastie et al. (2001).

The almost unbiasedness of IWCV holds for any loss function, any model, and any parameter learning method; even non-identifiable models Watanabe (2009) or non-parametric learning methods Schölkopf and Smola (2002) are allowed. Thus IWCV is a highly flexible model selection technique under covariate shift. For other model selection criteria under covariate shift, see Shimodaira (2000) for regular models with smooth losses and Sugiyama and Müller (2005) for linear models with the squared loss.

31.4.2 Numerical Examples

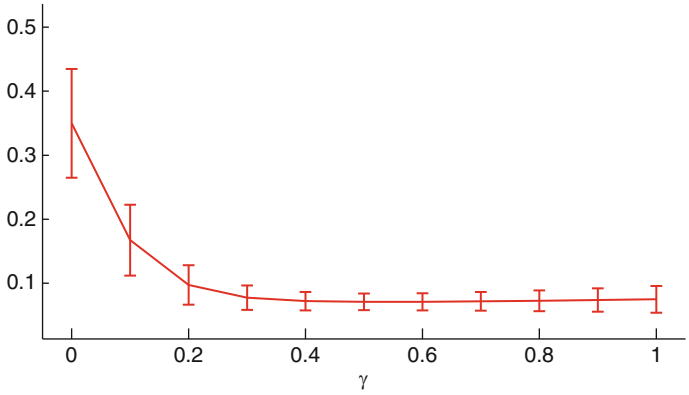
Here we illustrate the behavior of IWCV using the same toy data sets as Sect. 31.3.2.

Regression

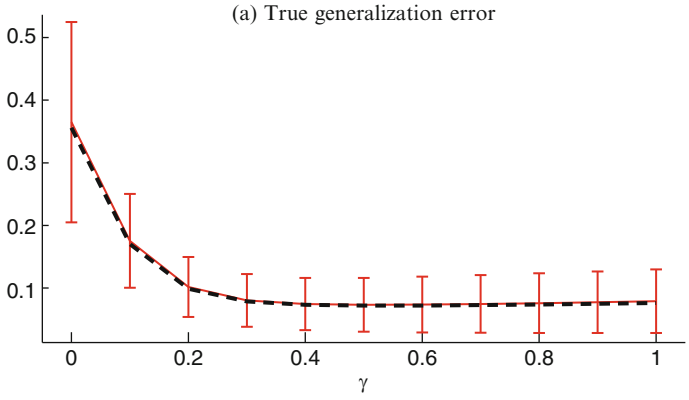
Let us continue the one-dimensional regression simulation in Sect. 31.3.2.

As illustrated in Fig. 31.4 in Sect. 31.3.2, IWLS with flattening parameter $\gamma = 0.5$ appears to work well for that particular realization of data samples. However, the best value of γ would depend on the realization of samples. In order to investigate this systematically, let us repeat the simulation 1,000 times with different random seeds, i.e., in each run $\{(x_i^{\mathrm{tr}}, \epsilon_i^{\mathrm{tr}})\}_{i=1}^{n_{\mathrm{tr}}}$ are randomly drawn and the scores of tenfold IWCV and tenfold ordinary CV are calculated for $\gamma = 0, 0.1, 0.2, \ldots, 1$. The means and standard deviations of the generalization error G and its estimate by each method are depicted as functions of γ in Fig. 31.6. The graphs show that IWCV gives very accurate unbiased estimates of the generalization error, while ordinary CV is heavily biased.

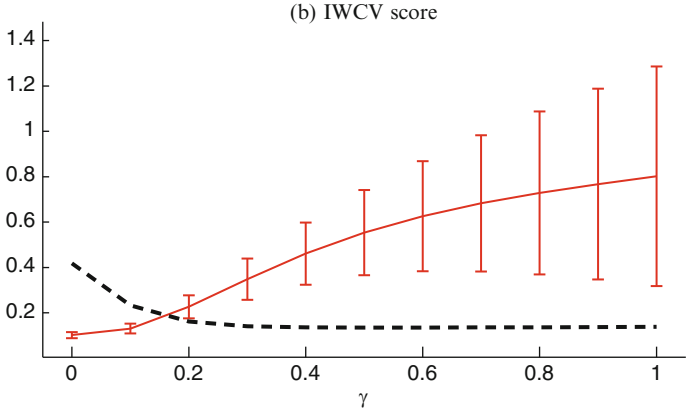
Next we investigate the model selection performance. The flattening parameter γ is chosen from $\{0, 0.1, 0.2, \ldots, 1\}$ so that the score of each model selection criterion is minimized. The mean and standard deviation of the generalization error G of the learned function obtained by each method over 1,000 runs are described in Table 31.1. This shows that IWCV gives significantly smaller generalization errors than ordinary CV, under the *t-test* Henkel (1976) at the significance level 5%. For reference, the generalization error when the flattening parameter γ is chosen optimally (i.e., in each trial, γ is chosen so that the true generalization error is minimized) is described as "Optimal" in the table. The result



(a) True generalization error



(b) IWCV score



(c) Ordinary CV score

Fig. 31.6 Generalization error and its estimates obtained by IWCV and ordinary CV as functions of the flattening parameter γ in IWLS for the regression examples in Fig. 31.4. Thick dashed curves in the bottom graphs depict the true generalization error for clear comparison

Table 31.1 The mean and standard deviation of the generalization error G obtained by each method for the toy regression data set. The best method and comparable ones by the t-test at the significance level 5% are indicated by "○". For reference, the generalization error obtained with the optimal γ (i.e., the minimum generalization error) is described as "Optimal"

IWCV	Ordinary CV	Optimal
$^{\circ}0.077 \pm 0.020$	0.356 ± 0.086	0.069 ± 0.011

Table 31.2 The mean and standard deviation of the generalization error G (i.e., the misclassification rate) obtained by each method for the toy classification data set. The best method and comparable ones by the t-test at the significance level 5% are indicated by "○". For reference, the generalization error obtained with the optimal γ (i.e., the minimum generalization error) is described as "Optimal"

IWCV	Ordinary CV	Optimal
$^{\circ}0.108 \pm 0.027$	0.131 ± 0.029	0.091 ± 0.009

shows that the performance of IWCV is rather close to that of the optimal choice.

Classification

Let us continue the toy classification simulation in Sect. 31.3.2.

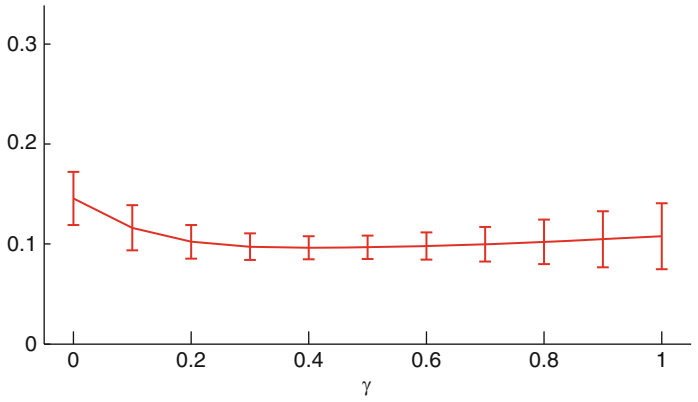
In Fig. 31.5 in Sect. 31.3.2, IWFDA with a middle/large flattening parameter γ appears to work well for that particular realization of samples. Here, we investigate the choice of the flattening parameter value by IWCV and ordinary CV. Figure 31.7 depicts the means and standard deviations of the generalization error G (i.e., the misclassification rate) and its estimate by each method over 1,000 runs, as functions of the flattening parameter γ in IWFDA. The graphs clearly show that IWCV gives much better estimates of the generalization error than ordinary CV.

Next we investigate the model selection performance. The flattening parameter γ is chosen from $\{0, 0.1, 0.2, \ldots, 1\}$ so that the score of each model selection criterion is minimized. The mean and standard deviation of the generalization error G of the learned function obtained by each method over 1,000 runs are described in Table 31.2. The table shows that IWCV gives significantly smaller test errors than ordinary CV, and the performance of IWCV is rather close to that of the optimal choice.

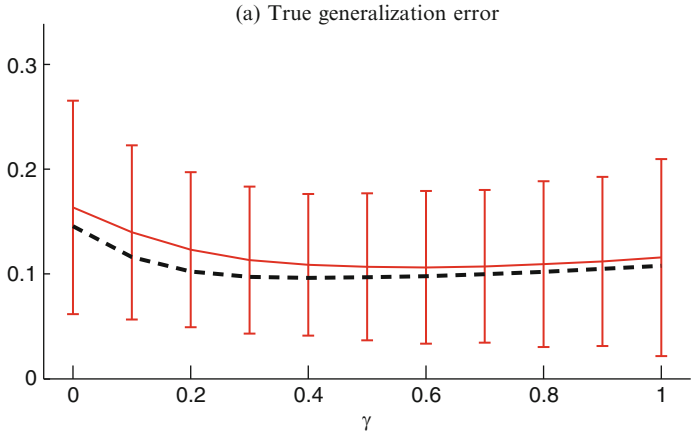
31.5 Importance Estimation

In the previous sections, we have seen that the importance weight

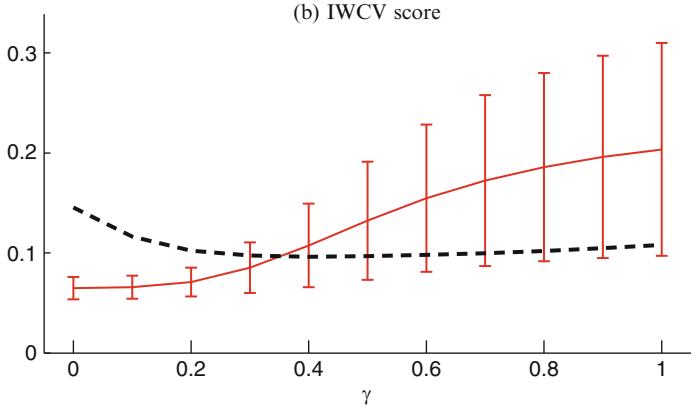
$$w(\boldsymbol{x}) = \frac{p_{\mathrm{te}}(\boldsymbol{x})}{p_{\mathrm{tr}}(\boldsymbol{x})}$$



(a) True generalization error



(b) IWCV score



(c) Ordinary CV score

Fig. 31.7 The generalization error G (i.e., the misclassification rate) and its estimates obtained by IWCV and ordinary CV as functions of the flattening parameter γ in IWFDA for the toy classification examples in Fig. 31.5. Thick dashed curves in the bottom graphs depict the true generalization error for clear comparison

plays a central role in covariate shift adaptation. However, the importance weight is unknown in practice and needs to be estimated from data. In this section, we review importance estimation methods.

Here we assume that in addition to the training input samples $\{\boldsymbol{x}_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ drawn independently from $p_{\mathrm{tr}}(\boldsymbol{x})$, we are given test input samples $\{\boldsymbol{x}_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$ drawn independently from $p_{\mathrm{te}}(\boldsymbol{x})$. Thus the goal of the importance estimation problem addressed here is to estimate the importance function $w(\boldsymbol{x})$ from $\{\boldsymbol{x}_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ and $\{\boldsymbol{x}_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$.

31.5.1 Kernel Density Estimation

Kernel density estimation (KDE) is a non-parametric technique to estimate a probability density function $p(\boldsymbol{x})$ from its i.i.d. samples $\{\boldsymbol{x}_i\}_{i=1}^{n}$. For the Gaussian kernel

$$K_\sigma(\boldsymbol{x}, \boldsymbol{x}') = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{x}'\|^2}{2\sigma^2}\right), \tag{31.5}$$

KDE is expressed as

$$\hat{p}(\boldsymbol{x}) = \frac{1}{n_{\mathrm{tr}}(2\pi\sigma^2)^{d/2}} \sum_{\ell=1}^{n} K_\sigma(\boldsymbol{x}, \boldsymbol{x}_\ell).$$

The performance of KDE depends on the choice of the kernel width σ. It can be optimized by cross-validation (CV) as follows Härdle et al. (2004): First, divide the samples $\{\boldsymbol{x}_i\}_{i=1}^{n}$ into k disjoint non-empty subsets $\{\mathcal{X}_r\}_{r=1}^{k}$ of (approximately) the same size. Then obtain a density estimator $\hat{p}_{\mathcal{X}_r}(\boldsymbol{x})$ from $\{\mathcal{X}_i\}_{i\neq r}$ (i.e., without $\mathcal{X}_r$), and compute its log-likelihood for the hold-out subset $\mathcal{X}_r$:

$$\frac{1}{|\mathcal{X}_r|} \sum_{\boldsymbol{x}\in\mathcal{X}_r} \log \hat{p}_{\mathcal{X}_r}(\boldsymbol{x}),$$

where $|\mathcal{X}|$ denotes the number of elements in the set $\mathcal{X}$. Repeat this procedure for $r = 1, 2, \ldots, k$ and choose the value of σ such that the average of the above hold-out log-likelihood over all r is maximized. Note that the average hold-out log-likelihood is an almost unbiased estimate of the Kullback-Leibler divergence from $p(\boldsymbol{x})$ to $\hat{p}(\boldsymbol{x})$, up to an irrelevant constant.

KDE can be used for importance estimation by first obtaining density estimators $\hat{p}_{\mathrm{tr}}(\boldsymbol{x})$ and $\hat{p}_{\mathrm{te}}(\boldsymbol{x})$ separately from $\{\boldsymbol{x}_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$ and $\{\boldsymbol{x}_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$, and then estimating the importance by $\hat{w}(\boldsymbol{x}) = \hat{p}_{\mathrm{te}}(\boldsymbol{x})/\hat{p}_{\mathrm{tr}}(\boldsymbol{x})$. However, division by an estimated density can magnify the estimation error, so directly estimating the importance weight in a single-shot process would be more preferable.

31.5.2 Kullback-Leibler Importance Estimation Procedure

The *Kullback-Leibler importance estimation procedure* (KLIEP) Sugiyama et al. (2008) directly gives an estimate of the importance function without going through density estimation by matching the two densities $p_{\mathrm{tr}}(\boldsymbol{x})$ and $p_{\mathrm{te}}(\boldsymbol{x})$ in terms of the *Kullback-Leibler divergence* Kullback and Leibler (1951).

Let us model the importance weight $w(\boldsymbol{x})$ by the following kernel model:

$$\hat{w}(\boldsymbol{x}) = \sum_{\ell=1}^{n_{\mathrm{te}}} \alpha_\ell K_\sigma(\boldsymbol{x}, \boldsymbol{x}_\ell^{\mathrm{te}}),$$

where $\boldsymbol{\alpha} = (\alpha_1, \alpha_2, \ldots, \alpha_{n_{\mathrm{te}}})^\top$ are parameters to be learned from data samples and $K_\sigma(\boldsymbol{x}, \boldsymbol{x}')$ is the Gaussian kernel (see (31.5)). An estimate of the density $p_{\mathrm{te}}(\boldsymbol{x})$ is given by using the model $\hat{w}(\boldsymbol{x})$ as $\hat{p}_{\mathrm{te}}(\boldsymbol{x}) = \hat{w}(\boldsymbol{x}) p_{\mathrm{tr}}(\boldsymbol{x})$. In KLIEP, the parameters $\boldsymbol{\alpha}$ are determined so that the Kullback-Leibler divergence from $p_{\mathrm{te}}(\boldsymbol{x})$ to $\hat{p}_{\mathrm{te}}(\boldsymbol{x})$ is minimized:

$$\mathrm{KL}(\boldsymbol{\alpha}) := \mathop{\mathbb{E}}_{\boldsymbol{x}^{\mathrm{te}}} \left[\log \frac{p_{\mathrm{te}}(\boldsymbol{x}^{\mathrm{te}})}{\hat{w}(\boldsymbol{x}^{\mathrm{te}}) p_{\mathrm{tr}}(\boldsymbol{x}^{\mathrm{te}})} \right] = \mathop{\mathbb{E}}_{\boldsymbol{x}^{\mathrm{te}}} \left[\log \frac{p_{\mathrm{te}}(\boldsymbol{x}^{\mathrm{te}})}{p_{\mathrm{tr}}(\boldsymbol{x}^{\mathrm{te}})} \right] - \mathop{\mathbb{E}}_{\boldsymbol{x}^{\mathrm{te}}} \left[\log \hat{w}(\boldsymbol{x}^{\mathrm{te}}) \right],$$

where $\mathbb{E}_{\boldsymbol{x}^{\mathrm{te}}}$ denotes the expectation over $\boldsymbol{x}^{\mathrm{te}}$ drawn from $p_{\mathrm{te}}(\boldsymbol{x})$. The first term is a constant, so it can be safely ignored. We define the negative of the second term by KL':

$$\mathrm{KL}'(\boldsymbol{\alpha}) := \mathop{\mathbb{E}}_{\boldsymbol{x}^{\mathrm{te}}} \left[\log \hat{w}(\boldsymbol{x}^{\mathrm{te}}) \right]. \tag{31.6}$$

Since $\hat{p}_{\mathrm{te}}(\boldsymbol{x})$ $(= \hat{w}(\boldsymbol{x}) p_{\mathrm{tr}}(\boldsymbol{x}))$ is a probability density function, it should satisfy

$$1 = \int_{\mathcal{D}} \hat{p}_{\mathrm{te}}(\boldsymbol{x}) d\boldsymbol{x} = \int_{\mathcal{D}} \hat{w}(\boldsymbol{x}) p_{\mathrm{tr}}(\boldsymbol{x}) d\boldsymbol{x} = \mathop{\mathbb{E}}_{\boldsymbol{x}^{\mathrm{tr}}} \left[\hat{w}(\boldsymbol{x}^{\mathrm{tr}}) \right]. \tag{31.7}$$

The KLIEP optimization problem is given by replacing the expectations in Eqs.(31.6) and (31.7) with empirical averages:

$$\max_{\{\alpha_\ell\}_{\ell=1}^{n_{\mathrm{te}}}} \left[\sum_{j=1}^{n_{\mathrm{te}}} \log \left(\sum_{\ell=1}^{n_{\mathrm{te}}} \alpha_\ell K(\boldsymbol{x}_j^{\mathrm{te}}, \boldsymbol{x}_\ell^{\mathrm{te}}) \right) \right]$$

$$\text{subject to } \frac{1}{n_{\mathrm{tr}}} \sum_{\ell=1}^{n_{\mathrm{te}}} \alpha_\ell \left(\sum_{i=1}^{n_{\mathrm{tr}}} K(\boldsymbol{x}_i^{\mathrm{tr}}, \boldsymbol{x}_\ell^{\mathrm{te}}) \right) = 1 \text{ and } \alpha_1, \alpha_2, \ldots, \alpha_{n_{\mathrm{te}}} \geq 0.$$

This is a *convex* optimization problem and the global solution – which tends to be *sparse* Boyd and Vandenberghe (2004) – can be obtained, e.g., by simply

Input: $\{\boldsymbol{x}_i^{\mathrm{tr}}\}_{i=1}^{n_{\mathrm{tr}}}$, $\{\boldsymbol{x}_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$, and σ
Output: $\hat{w}(\boldsymbol{x})$

$A_{j,\ell} \longleftarrow K_\sigma(\boldsymbol{x}_j^{\mathrm{te}}, \boldsymbol{x}_\ell^{\mathrm{te}})$ for $j, \ell = 1, 2, \ldots, n_{\mathrm{te}}$;
$b_\ell \longleftarrow \frac{1}{n_{\mathrm{tr}}} \sum_{i=1}^{n_{\mathrm{tr}}} K_\sigma(\boldsymbol{x}_i^{\mathrm{tr}}, \boldsymbol{x}_\ell^{\mathrm{te}})$ for $\ell = 1, 2, \ldots, n_{\mathrm{te}}$;
Initialize $\boldsymbol{\alpha}$ ($> \mathbf{0}_{n_{\mathrm{te}}}$) and ε ($0 < \varepsilon \ll 1$);
Repeat until convergence
$\boldsymbol{\alpha} \longleftarrow \boldsymbol{\alpha} + \varepsilon \boldsymbol{A}^\top (\mathbf{1}_{n_{\mathrm{te}}} ./ \boldsymbol{A}\boldsymbol{\alpha})$; % Gradient ascent
$\boldsymbol{\alpha} \longleftarrow \boldsymbol{\alpha} + (1 - \boldsymbol{b}^\top \boldsymbol{\alpha}) \boldsymbol{b} / (\boldsymbol{b}^\top \boldsymbol{b})$; % Constraint satisfaction
$\boldsymbol{\alpha} \longleftarrow \max(\mathbf{0}_{n_{\mathrm{te}}}, \boldsymbol{\alpha})$; % Constraint satisfaction
$\boldsymbol{\alpha} \longleftarrow \boldsymbol{\alpha} / (\boldsymbol{b}^\top \boldsymbol{\alpha})$; % Constraint satisfaction
end
$\hat{w}(\boldsymbol{x}) \longleftarrow \sum_{\ell=1}^{n_{\mathrm{te}}} \alpha_\ell K_\sigma(\boldsymbol{x}, \boldsymbol{x}_\ell^{\mathrm{te}})$;

Fig. 31.8 Pseudo code of KLIEP. $\mathbf{0}_{n_{\mathrm{te}}}$ denotes the n_{te}-dimensional vector with all zeros, and $\mathbf{1}_{n_{\mathrm{te}}}$ denotes the n_{te}-dimensional vector with all ones. "./" indicates the element-wise division, and inequalities and the "max" operation for vectors are applied in the element-wise manner

performing gradient ascent and feasibility satisfaction iteratively. A pseudo code is summarized in Fig. 31.8. The Gaussian width σ can be optimized by CV over KL′, where only the test samples $\{\boldsymbol{x}_j^{\mathrm{te}}\}_{j=1}^{n_{\mathrm{te}}}$ are divided into k disjoint subsets Sugiyama et al. (2008).

A MATLAB® implementation of the entire KLIEP algorithm is available from the following web page. http://sugiyama-www.cs.titech.ac.jp/~sugi/software/KLIEP/

31.5.3 Numerical Examples

Here, we illustrate the behavior of the KLIEP method.

Let us consider the following one-dimensional importance estimation problem:

$$p_{\mathrm{tr}}(x) = N(x; 1, (1/2)^2) \quad \text{and} \quad p_{\mathrm{te}}(x) = N(x; 2, (1/4)^2).$$

Let the number of training samples be $n_{\mathrm{tr}} = 200$ and the number of test samples be $n_{\mathrm{te}} = 1{,}000$.

Figure 31.9 depicts the true importance and its estimates by KLIEP, where three different Gaussian widths $\sigma = 0.02, 0.2, 0.8$ are tested. The graphs show that the performance of KLIEP is highly dependent on the Gaussian width. More specifically, the estimated importance function $\hat{w}(x)$ is highly fluctuated when σ is small, while it is overly smoothed when σ is large. When σ is chosen appropriately, KLIEP seems to work reasonably well for this example.

Figure 31.10 depicts the values of the true J (see (31.6)) and its estimate by fivefold CV; the means, the 25 percentiles, and the 75 percentiles over 100 trials are plotted as functions of the Gaussian width σ. This shows that CV gives a very good estimate of J, which results in an appropriate choice of σ.

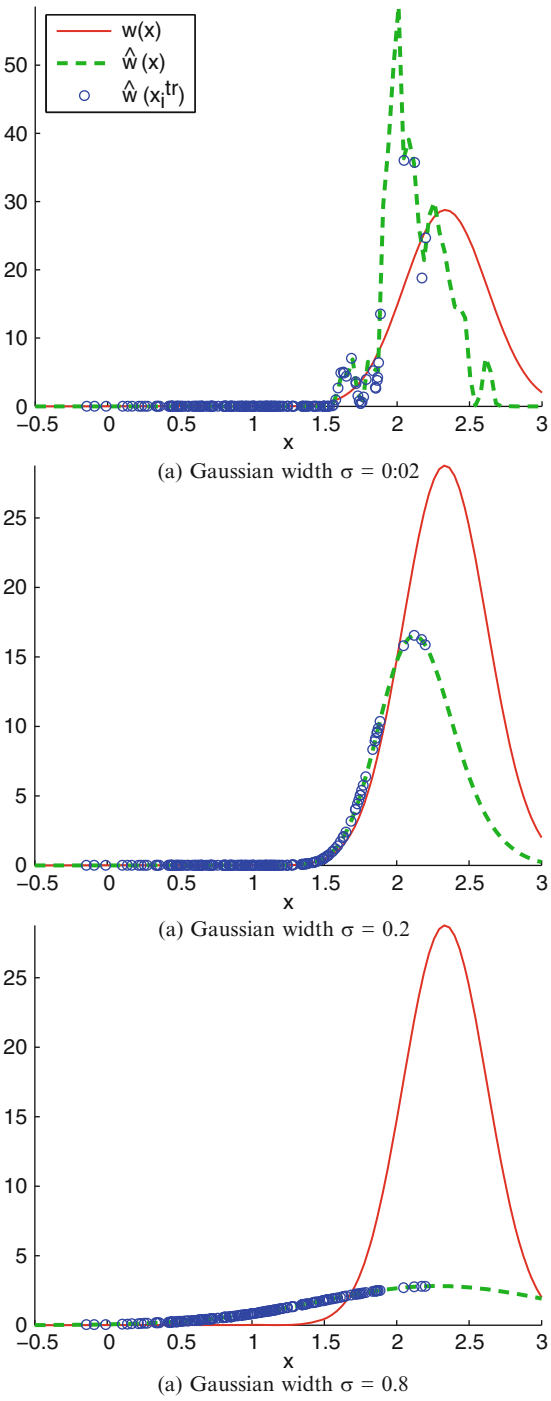


Fig. 31.9 Results of importance estimation by KLIEP. $w(x)$ is the true importance function and $\hat{w}(x)$ is its estimation obtained by KLIEP

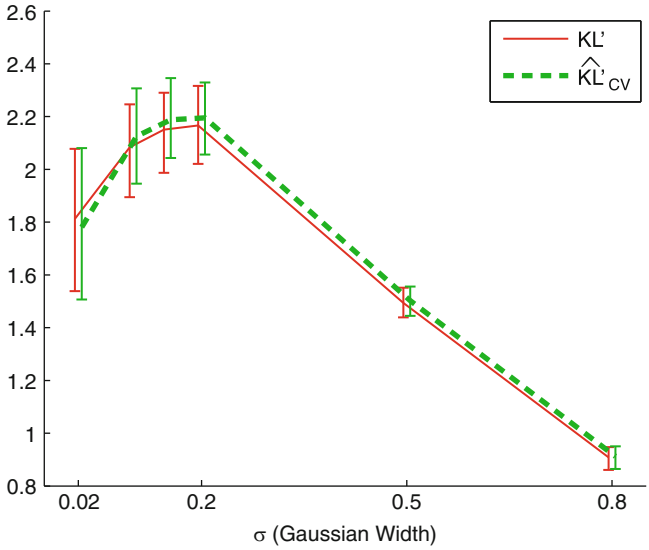


Fig. 31.10 Model selection curve for KLIEP. KL' is the true score of an estimated importance (see (31.6)) and $\widehat{\mathrm{KL}}'_{\mathrm{CV}}$ is its estimate by fivefold CV

31.6 Conclusions and Outlook

In standard supervised learning theories, test input points are assumed to follow the same probability distribution as training input points. However, this assumption is often violated in real-world learning problems. In this chapter, we reviewed recently proposed techniques for covariate shift adaptation, including importance-weighted empirical risk minimization, importance-weighted cross-validation, and direct importance estimation.

In Sect. 31.5, we introduced the KLIEP algorithm for importance estimation, where linearly-parameterized models were used. It was shown that the KLIEP idea can also be naturally applied to log-linear models Tsuboi et al. (2009), Gaussian mixture models Yamada and Sugiyama (2009), and probabilistic principal component analysis mixture models Yamada et al. (2010b). Other than KLIEP, various methods of direct importance estimation have also been proposed Bickel et al. (2007); Cheng and Chu (2004); Ćwik and Mielniczuk (1989); Huang et al. (2007); Kanamori et al. (2009a); Qin (1998); Silverman (1978). Among them, the method proposed in Kanamori et al. (2009a) called *unconstrained least-squares importance fitting* (uLSIF) gives an analytic-form solution and the solution can be computed very efficiently in a stable manner. Thus it can be applied to large-scale data sets.

Recently, importance estimation methods which incorporate dimensionality reduction have been developed. A method proposed by Sugiyama et al. (2010a) uses

a supervised dimensionality reduction technique called *local Fisher discriminant analysis* Sugiyama (2007) for identifying a subspace in which two densities are significantly different (which is called the *hetero-distributional subspace*). Another method proposed by Sugiyama et al. (2011) tries to find the hetero-distributional subspace by directly minimizing the discrepancy between the two distributions. Theoretical analysis of importance estimation has also been conducted thoroughly Bensaid and Fabre (2007); Chen et al. (2009); Cheng and Chu (2004); Ćwik and Mielniczuk (1989); Gijbels and Mielniczuk (1995); Jacob and Oliveira (1997); Kanamori et al. (2009b, 2010); Nguyen et al. (2010); Qin (1998); Silverman (1978); Sugiyama et al. (2008).

It has been shown that various statistical data processing tasks can be solved through importance estimation Sugiyama et al. (2009, 2012 to appear), including multi-task learning Bickel et al. (2007), inlier-based outlier detection Hido et al. (2008, 2011); Silverman (1978); Smola et al. (2009), change detection in time series Kawahara and Sugiyama (2011, to appear), mutual information estimation Suzuki et al. (2008, 2009b), independent component analysis Suzuki and Sugiyama (2011), feature selection Suzuki et al. (2009a), sufficient dimension reduction Suzuki and Sugiyama (2010), causal inference Yamada and Sugiyama (2010), conditional density estimation Sugiyama et al. (2010b), and probabilistic classification Sugiyama (2010). Thus, following this line of research, further improving the accuracy and computational efficiency of importance estimation as well as further exploring possible application of importance estimation would be a promising direction to be pursued.

Acknowledgements The author was supported by MEXT Grant-in-Aid for Young Scientists (A) 20680007, SCAT, and AOARD.

References

Akaike, H.: Statistical predictor identification. Ann. Inst. Stat. Math. **22**, 203–217 (1970)
Akaike, H.: A new look at the statistical model identification. IEEE Trans. Automat. Contr. **AC-19**, 716–723 (1974)
Akaike, H.: Likelihood and the Bayes procedure. Bayesian Statistics, pp. 141–166. Valencia University Press, Spain (1980)
Akiyama, T., Hachiya, H., Sugiyama, M.: Efficient exploration through active learning for value function approximation in reinforcement learning. Neural Netw. **23**(5), 639–648 (2010)
Bensaid, N., Fabre, J.P.: Optimal asymptotic quadratic error of kernel estimators of Radon-Nikodym derivatives for strong mixing data. J. Nonparametr. Stat. **19**, 77–088 (2007)
Bickel, S., Brückner, M., Scheffer, T.: Discriminative learning for differing training and test distributions. Proceedings of the 24th International Conference on Machine Learning pp. 81–88 (2007)
Bishop, C.M.: Neural Networks for Pattern Recognition. Clarendon Press, Oxford (1995)
Boyd, S., Vandenberghe, L.: Convex Optimization. Cambridge University Press, Cambridge, UK (2004)
Breiman, L.: Arcing classifiers. Ann. Stat. **26**, 801–849 (1998)

Chen, S.-M., Hsu, Y.-S., Liaw, J.-T.: On kernel estimators of density ratio. Statistics **43**, 463–479 (2009)
Chen, S.S., Donoho, D.L., Saunders, M.A.: Atomic decomposition by basis pursuit. SIAM J. Sci. Comput. **20**, 33–61 (1998)
Cheng, K.F., Chu, C.K.: Semiparametric density estimation under a two-sample density ratio model. Bernoulli **10**, 583–604 (2004)
Craven, P., Wahba, G.: Smoothing noisy data with spline functions: Estimating the correct degree of smoothing by the method of generalized cross-validation. Numer. Math. **31**, 377–403 (1979)
Ćwik, J., Mielniczuk, J.: Estimating density ratio with application to discriminant analysis. Comm. Stat. Theor. Meth. **18**, 3057–3069 (1989)
Duda, R.O., Hart, P.E., Stor, D.G.: Pattern Classification. Wiley, New York (2001)
Efron, B., Tibshirani, R.J.: An Introduction to the Bootstrap. Chapman & Hall, New York (1993)
Fisher, R.A.: The use of multiple measurements in taxonomic problems. Ann. Eugen. **7**, 179–188 (1936)
Fishman, G.S.: Monte Carlo: Concepts, Algorithms, and Applications. Springer, Berlin, Germany (1996)
Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. Proceedings of 13th International Conference on Machine Learning pp. 148–146. Morgan Kaufmann (1996)
Friedman, J., Hastie, T., Tibshirani, R.: Additive logistic regression: A statistical view of boosting. Ann. Stat. **28**, 337–407 (2000)
Gijbels, I., Mielniczuk, J.: Asymptotic properties of kernel estimators of the Radon-Nikodym derivative with applications to discriminant analysis. Stat. Sin. **5**, 261–278 (1995)
Hachiya, H., Akiyama, T., Sugiyama, M., Peters, J.: Adaptive importance sampling for value function approximation in off-policy reinforcement learning. Neural Netw. **22**, 1399–1410 (2009)
Hachiya, H., Peters, J., Sugiyama, M.: Reward weighted regression with sample reuse for direct policy search in reinforcement learning. Neural Computation, **23**, 2798–2832 (2011)
Härdle, W., Müller, M., Sperlich, S., Werwatz, A.: *Nonparametric and semiparametric models*. Springer, Berlin, Germany (2004)
Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York (2001)
Henkel, R.E.: Tests of Significance. SAGE Publication, Beverly Hills (1976)
Hido, S., Tsuboi, Y., Kashima, H., Sugiyama, M., Kanamori, T.: Inlier-based outlier detection via direct density ratio estimation. Proceedings of IEEE International Conference on Data Mining (ICDM2008) pp. 223–232. Pisa, Italy (2008)
Hido, S., Tsuboi, Y., Kashima, H., Sugiyama, M., Kanamori, T.: Statistical outlier detection using direct density ratio estimation. Knowledge and Information Systems, **26**, 309–336 (2011)
Huang, J., Smola, A., Gretton, A., Borgwardt, K.M., Schölkopf, B.: Correcting sample selection bias by unlabeled data. In: Schölkopf, B., Platt, J., Hoffman, T. (eds.) Advances in neural information processing systems 19, 601–608. MIT Press, Cambridge, MA (2007)
Ishiguro, M., Sakamoto, Y., Kitagawa, G.: Bootstrapping log likelihood and EIC, an extension of AIC. Ann. Inst. Stat. Math. **49**, 411–434 (1997)
Jacob, P., Oliveira, P.E.: Kernel estimators of general Radon-Nikodym derivatives. Statistics **30**, 25–46 (1997)
Kanamori, T.: Pool-based active learning with optimal sampling distribution and its information geometrical interpretation. Neurocomputing **71**, 353–362 (2007)
Kanamori, T., Hido, S., Sugiyama, M.: A least-squares approach to direct importance estimation. J. Mach. Learn. Res. **10**, 1391–1445 (2009a)
Kanamori, T., Shimodaira, H.: Active learning algorithm using the maximum weighted log-likelihood estimator. J. Stat. Plann. Infer. **116**, 149–162 (2003)
Kanamori, T., Suzuki, T., Sugiyama, M.: Condition number analysis of kernel-based density ratio estimation (Technical Report). arXiv (2009b)
Kanamori, T., Suzuki, T., Sugiyama, M.: Theoretical analysis of density ratio estimation. IEICE Trans. Fund. Electron. Comm. Comput. Sci. **E93-A**, 787–798 (2010)

Kawahara, Y., Sugiyama, M.: Sequential change-point detection based on direct density-ratio estimation. Statistical Analysis and Data Mining (2011); to appear
Konishi, S., Kitagawa, G.: Generalized information criteria in model selection. Biometrika **83**, 875–890 (1996)
Kullback, S., Leibler, R.A.: On information and sufficiency. Ann. Math. Stat. **22**, 79–86 (1951)
Luntz, A., Brailovsky, V.: On estimation of characters obtained in statistical procedure of recognition. Technicheskaya Kibernetica **3**, (1969)
Mallows, C.L.: Some comments on C_P. Technometrics **15**, 661–675 (1973)
Mangasarian, O.L., Musicant, D.R.: Robust linear and support vector regression. IEEE Trans. Pattern Anal. Mach. Intell. **22**, 950–955 (2000)
Murata, N., Yoshizawa, S., Amari, S.: Network information criterion – Determining the number of hidden units for an artificial neural network model. IEEE Trans. Neural Netw. **5**, 865–872 (1994)
Nguyen, X., Wainwright, M.J., Jordan, M.I.: Estimating divergence functionals and the likelihood ratio by convex risk minimization. IEEE Trans. Information Theory, **56**, 5847–5861 (2010)
Qin, J.: Inferences for case-control and semiparametric two-sample density ratio models. Biometrika **85**, 619–639 (1998)
Quiñonero-Candela, J., Sugiyama, M., Schwaighofer, A., Lawrence, N. (ed.): Dataset Shift in Machine Learning. MIT Press, Cambridge, MA (2009)
Rissanen, J.: Modeling by shortest data description. Automatica **14**, 465–471 (1978)
Rissanen, J.: Stochastic complexity. J. Roy. Stat. Soc. B **49**, 223–239 (1987)
Schölkopf, B., Smola, A.J.: Learning with Kernels. MIT Press, Cambridge, MA (2002)
Schwarz, G.: Estimating the dimension of a model. Ann. Stat. **6**, 461–464 (1978)
Shibata, R.: Statistical aspects of model selection. In: Willems, J.C. (eds.) From Data to Model, pp. 215–240. Springer, New York (1989)
Shimodaira, H.: Improving predictive inference under covariate shift by weighting the log-likelihood function. J. Stat. Plan. Infer. **90**, 227–244 (2000)
Silverman, B.W.: Density ratios, empirical likelihood and cot death. J. Roy. Stat. Soc. C **27**, 26–33 (1978)
Smola, A., Song, L., Teo, C.H.: Relative novelty detection. Twelfth International Conference on Artificial Intelligence and Statistics pp. 536–543. (2009)
Stone, M.: Cross-validatory choice and assessment of statistical predictions. J. Roy. Stat. Soc. B **36**, 111–147 (1974)
Sugiyama, M.: Active learning in approximately linear regression based on conditional expectation of generalization error. J. Mach. Learn. Res. **7**, 141–166 (2006)
Sugiyama, M.: Dimensionality reduction of multimodal labeled data by local Fisher discriminant analysis. J. Mach. Learn. Res. **8**, 1027–1061 (2007)
Sugiyama, M.: Superfast-trainable multi-class probabilistic classifier by least-squares posterior fitting. IEICE Trans. Infor. Syst. E93-D, **10**, 2690–2701 (2010)
Sugiyama, M., Yamada, M., von Bünau, P., Suzuki, T., Kanamori, T., Kawanabe, M.: Direct density-ratio estimation with dimensionality reduction via least-squares hetero-distributional subspace search. Neural Networks, **24**, 183–198 (2011).
Sugiyama, M., Kanamori, T., Suzuki, T., Hido, S., Sese, J., Takeuchi, I., Wang, L.: A density-ratio framework for statistical data processing. IPSJ Trans. Comput. Vision Appl. **1**, 183–208 (2009)
Sugiyama, M., Kawanabe, M., Chui, P.L.: Dimensionality reduction for density ratio estimation in high-dimensional spaces. Neural Netw. **23**, 44–59 (2010a)
Sugiyama, M., Kawanabe, M., Müller, K.-R.: Trading variance reduction with unbiasedness: The regularized subspace information criterion for robust model selection in kernel regression. Neural Comput. **16**, 1077–1104 (2004)
Sugiyama, M., Krauledat, M., Müller, K.-R.: Covariate shift adaptation by importance weighted cross validation. J. Mach. Learn. Res. **8**, 985–1005 (2007)
Sugiyama, M., Müller, K.-R.: The subspace information criterion for infinite dimensional hypothesis spaces. J. Mach. Learn. Res. **3**, 323–359 (2002)

Sugiyama, M., Müller, K.-R.: Input-dependent estimation of generalization error under covariate shift. Stat. Decis. **23**, 249–279 (2005)
Sugiyama, M., Nakajima, S.: Pool-based active learning in approximate linear regression. Mach. Learn. **75**, 249–274 (2009)
Sugiyama, M., Ogawa, H.: Subspace information criterion for model selection. Neural Comput. **13**, 1863–1889 (2001)
Sugiyama, M., Suzuki, T., Kanamori, T.: Density ratio estimation in machine learning: A versatile tool for statistical data processing. Cambridge University Press, Cambridge, UK (2012, to appear)
Sugiyama, M., Suzuki, T., Nakajima, S., Kashima, H., von Bünau, P., Kawanabe, M.: Direct importance estimation for covariate shift adaptation. Ann. Inst. Stat. Math. **60**, 699–746 (2008)
Sugiyama, M., Takeuchi, I., Suzuki, T., Kanamori, T., Hachiya, H., Okanohara, D.: Least-squares conditional density estimation. IEICE Trans. Inform. Syst. **E93-D**, 583–594 (2010b)
Sugiyama, M., Kawanabe, M.: Machine learning in non-stationary environments: introduction to covariate shift adaptation, MIT Press, Cambridge, MA (2011), to appear
Suzuki, T., Sugiyama, M.: Least-squares independent component analysis. Neural Computation, **23**, 284–301 (2011)
Suzuki, T., Sugiyama, M.: Sufficient dimension reduction via squared-loss mutual information estimation. Proceedings of The Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS2010) pp. 804–811. Sardinia, Italy (2010)
Suzuki, T., Sugiyama, M., Kanamori, T., Sese, J.: Mutual information estimation reveals global associations between stimuli and biological processes. BMC Bioinformatics **10**, S52 (2009a)
Suzuki, T., Sugiyama, M., Sese, J., Kanamori, T.: Approximating mutual information by maximum likelihood density ratio estimation. JMLR Workshop and Conference Proceedings, pp. 5–20 (2008)
Suzuki, T., Sugiyama, M., Tanaka, T.: Mutual information approximation via maximum likelihood estimation of density ratio. Proceedings of 2009 IEEE International Symposium on Information Theory (ISIT2009), pp. 463–467. Seoul, Korea (2009b)
Takeuchi, K.: Distribution of information statistics and validity criteria of models. Math. Sci. **153**, 12–18 (1976); in Japanese
Tibshirani, R.: Regression shrinkage and selection via the lasso. J. Roy. Stat. Soc. B, **58**, 267–288 (1996)
Tsuboi, Y., Kashima, H., Hido, S., Bickel, S., Sugiyama, M.: Direct density ratio estimation for large-scale covariate shift adaptation. J. Inform. Process. **17**, 138–155 (2009)
Ueki, K., Sugiyama, M., Ihara, Y.: Lighting condition adaptation for perceived age estimation. IEICE Trans. Information and Systems, **E94-D**, 392–395 (2011)
Vapnik, V.N.: Statistical Learning Theory. New York, Wiley (1998)
Wahba, G.: Spline models for observational data. Philadelphia and Pennsylvania: Society for Industrial and Applied Mathematics (1990)
Watanabe, S.: Algebraic Geometry and Statistical Learning. Cambridge University Press, Cambridge, UK (2009)
Wiens, D.P.: Robust weights and designs for biased regression models: Least squares and generalized M-estimation. J. Stat. Plann. Infer. **83**, 395–412 (2000)
Williams, P.M.: Bayesian regularization and pruning using a Laplace prior. Neural Comput. **7**, 117–143 (1995)
Yamada, M., Sugiyama, M.: Direct importance estimation with Gaussian mixture models. IEICE Trans. Inform. Syst. **E92-D**, 2159–2162 (2009)
Yamada, M., Sugiyama, M.: Dependence minimizing regression with model selection for non-linear causal inference under non-Gaussian noise. Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence (AAAI2010) pp. 643-648, Atlanta, GA (2010)
Yamada, M., Sugiyama, M., Matsui, T.: Semi-supervised speaker identification under covariate shift. Signal Processing, **90**, 2353–2361 (2010a)

Yamada, M., Sugiyama, M., Wichern, G., Simm, J.: Direct importance estimation with a mixture of probabilistic principal component analyzers. IEICE Trans. Information and Systems, **E93-D**, 2846–2849 (2010b)

Zadrozny, B.: Learning and evaluating classifiers under sample selection bias. Proceedings of the Twenty-First International Conference on Machine Learning, pp. 903–910. ACM Press, New York, NY (2004)

Chapter 32
Saddlepoint Approximations: A Review and Some New Applications

Simon A. Broda and Marc S. Paolella

32.1 Introduction

The saddlepoint method of approximation is attributed to Daniels (1954), and can be described in basic terms as yielding an accurate and usually fast and very numerically reliable approximation to the mass or density function (hereafter pdf), and the cumulative distribution function (cdf), of a random variable, say X, based on knowledge of its moment generating function (mgf). Denote the latter by $\mathbb{M}_X(s)$, where s is the real argument of the function, such that s is contained in the convergence strip of $\mathbb{M}_X(s)$, to be defined below. Several surveys and monographs are available; the best starting point is the currently definitive exposition in Butler (2007), along with the first textbook dedicated to the subject, Jensen (1995). Our goal is to outline the basics of the methodology in the easiest way possible, and then to illustrate a small subset of its many applications.

The outline of this article is as follows. Section 32.2 provides basic derivations of the fundamental approximations, and an illustration of the mechanics via a nontrivial example. Section 32.3 contains less well-known material on the use of the saddlepoint approximation for partial expectations, which are of utmost value in financial applications. Section 32.4 outlines the major results in the multivariate case and provides an example of application to the noncentral t distribution, the exact calculation of which, particularly in the doubly noncentral case, is foreboding. Section 32.5 details the mechanics of the saddlepoint approximation

S.A. Broda (✉)
Department of Quantitative Economics, University of Amsterdam, Amsterdam, The Netherlands
e-mail: s.a.broda@uva.nl

M.S. Paolella
Swiss Banking Institute, University of Zurich, Zurich, Switzerland

Swiss Finance Institute, Zurich, Switzerland
e-mail: paolella@isb.uzh.ch

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_32,

for quadratic forms, and ratios of quadratic forms, which are ubiquitous in linear regression models and time series analysis. Lastly, Sect. 32.6 provides results for a highly relevant application in financial risk management. An appendix details the derivation of a technical result.

32.2 Basic Derivation of the Univariate PDF and CDF Approximations

We first briefly review some definitions and results. The mgf of random variable X is the function $\mathbb{M}_X : \mathbb{R} \mapsto \mathbb{X}_{\geq 0}$ (where $\mathbb{X}$ is the extended real line) given by $s \mapsto \mathbb{E}\left[e^{sX}\right]$. The mgf $\mathbb{M}_X$ is said to exist if it is finite on a neighborhood of zero, i.e., if there is an $h > 0$ such that, $\forall s \in (-h, h)$, $\mathbb{M}_X(s) < \infty$. If $\mathbb{M}_X$ exists, then the largest (open) interval U around zero such that $\mathbb{M}_X(s) < \infty$ for $s \in U$ is referred to as the convergence strip (of the mgf) of X. If $\mathbb{M}_X$ exists, then all positive moments of X exist, i.e., $\forall r \in \mathbb{R}_{>0}$, $\mathbb{E}\left[|X|^r\right] < \infty$, and also $\mathbb{M}_X^{(j)}(s)\Big|_{s=0} = \mathbb{E}\left[X^j\right]$, where $\mathbb{M}_X^{(j)}$ is the jth derivative.

The cumulant generating function, or cgf, is defined as $\mathbb{K}_X(s) = \log \mathbb{M}_X(s)$, with the terms κ_i in the series expansion $\mathbb{K}_X(s) = \sum_{r=0}^{\infty} \kappa_r s^r / r!$ referred to as the cumulants of X. The jth derivative of $\mathbb{K}_X(s)$ evaluated at $s = 0$ is thus κ_j. It is straightforward to show that

$$\kappa_1 = \mu, \qquad \kappa_2 = \mu_2, \qquad \kappa_3 = \mu_3 \qquad \text{and} \qquad \kappa_4 = \mu_4 - 3\mu_2^2, \tag{32.1}$$

where $\mu_j = \mathbb{E}[(X - \mu)^j]$, and $\mu = \mathbb{E}[X]$.

We can now illustrate the simplest derivation of the saddlepoint approximation (hereafter SPA) to the pdf. The SPA is usually derived as an approximation to the pdf of the mean of n iid random variables. In such a setting, it can be shown that the accuracy of the approximation increases as n grows. This is because the SPA is, in fact, the first term in an asymptotic expansion. In this paper however, we will set $n = 1$ and treat the SPA simply as an approximation. Butler (2007) presents more sophisticated derivations, which add further insight into the nature of the approximation and lend themselves to analysis of its properties and extension to higher order terms.

Let X be a random variable, with continuous pdf f_X or discrete mass function f_X, and mgf $\mathbb{M}_X$ existing in an open neighborhood U around zero. From (32.1), the mean and variance of X can be expressed as $\mathbb{K}'_X(0)$ and $\mathbb{K}''_X(0)$, respectively. If T_s is a random variable having density

$$f_{T_s}(x; s) = \frac{e^{xs} f_X(x)}{\mathbb{M}_X(s)}, \tag{32.2}$$

for some $s \in U$, then T_s is referred to as an exponentially tilted random variable. Notice that its density integrates to one. Its mgf and cgf are easily seen to be

$$\mathbb{M}_{T_s}(t) = \frac{\mathbb{M}_X(t+s)}{\mathbb{M}_X(s)}, \quad \mathbb{K}_{T_s}(t) = \mathbb{K}_X(t+s) - \mathbb{K}_X(s),$$

so that $\mathbb{E}[T_s] = \mathbb{K}'_T(0) = \mathbb{K}'_X(s)$ and $\mathbb{V}(T_s) = \mathbb{K}''_X(s)$. Let $s_0 \in U$. Now consider using the normal distribution to approximate the true distribution of T_{s_0}; it must have mean $x_0 := \mathbb{K}'_X(s_0)$ and variance $v_0 := \mathbb{K}''_X(s_0)$, and is thus given by $x \mapsto \phi(x; x_0, v_0)$, where ϕ is the normal pdf. Use of (32.2) then yields an approximation for f_X as

$$x \mapsto \phi(x; x_0, v_0)\, \mathbb{M}_X(s_0)\, \mathrm{e}^{-s_0 x} = \frac{1}{\sqrt{2\pi\, v_0}} \exp\left\{-\frac{1}{2v_0}(x - x_0)^2\right\} \mathbb{M}_X(s_0)\, \mathrm{e}^{-s_0 x}.$$

The accuracy of this approximation to f_X, for a fixed x, depends crucially on the choice of s_0. We know that, in general, the normal approximation to the distribution of a random variable X is accurate near the mean of X, but degrades in the tails. As such, we are motivated to choose an s_0 such that x is close to the mean of the tilted distribution. In particular, we would like to find a value $\hat{s}$ such that

$$\mathbb{K}'_X(\hat{s}) = x, \tag{32.3}$$

for which it can be shown that a unique solution exists when $\hat{s}$ is restricted to U, the convergence strip of the mgf of X. The normal density approximation to the tilted random variable with mean x at the point x is then $\phi\left(x; \mathbb{K}'_X(\hat{s}), \mathbb{K}''_X(\hat{s})\right)$, and the approximation for f_X simplifies to

$$\hat{f}_X(x) = \frac{1}{\sqrt{2\pi\, \mathbb{K}''_X(\hat{s})}} \exp\{\mathbb{K}_X(\hat{s}) - x\hat{s}\}, \quad x = \mathbb{K}'_X(\hat{s}). \tag{32.4}$$

Approximation $\hat{f}_X$ is referred to as the (first order) saddlepoint approximation to f_X, where $\hat{s} = \hat{s}(x)$ is the solution to the saddlepoint equation, and is referred to as the saddlepoint at x. In many applications of interest, this has to be determined numerically. The approximation is valid for all values of x in the interior of the support of X; for example, if X follows a gamma distribution, then the SPA is valid only for $x > 0$, and if $X \sim \mathrm{Bin}(n, p)$, then the SPA is valid for $x = 1, 2, \ldots, n-1$.

The SPA (32.4) will not, in general, integrate to one, although it will usually not be far off. It will often be possible to re-normalize it, i.e.,

$$\bar{\hat{f}}_X(x) = \frac{\hat{f}_X(x)}{\int \hat{f}_X(x)\, \mathrm{d}x}, \tag{32.5}$$

which is a proper density. In doing so, it is advantageous to change the integration variable from x to $\hat{s}$ in order to avoid having to solve the saddlepoint equation for each evaluation of the integrand.

Before continuing, it is worth illustrating how the exact pdf could be recovered from the mgf. More generally in fact, the characteristic function, or cf, is used. It is defined as $\mathbb{E}\left[\mathrm{e}^{isX}\right]$, where $i^2 = -1$, and is usually denoted as $\varphi_X(s)$. Let X be a random variable whose mgf exists. Comparing the definitions, it appears that $\varphi_X(s) = \mathbb{M}_X(is)$, although formally, this does not make sense because we are plugging a complex variable into a function that only admits real arguments. Fortunately, in the vast majority of real applications, the mgf will have a functional form which allows for complex arguments in an obvious manner. See Paolella (2007, Sect. 1.2.4) and the references therein for further details. If X is a continuous random variable with pdf f_X and $\int_{-\infty}^{\infty} |\varphi_X(s)|\,\mathrm{d}s < \infty$, then

$$f_X(x) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \mathrm{e}^{-isx}\varphi_X(s)\,\mathrm{d}s, \tag{32.6}$$

which is referred to as the inversion formula (for the pdf). If the cgf of X exists, then (32.6) can be expressed as

$$f_X(x) = \frac{1}{2\pi i}\int_{-i\infty}^{i\infty} \exp\{\mathbb{K}_X(s) - sx\}\,\mathrm{d}s. \tag{32.7}$$

Gil–Pelaez (1951) derived the inversion formula for the cdf as

$$F_X(x) = \frac{1}{2} + \frac{1}{2\pi}\int_0^{\infty} \frac{\mathrm{e}^{isx}\varphi_X(-s) - \mathrm{e}^{-isx}\varphi_X(s)}{is}\,\mathrm{d}s. \tag{32.8}$$

Now consider the saddlepoint method. An approximation to the cdf of X could be obtained by numerically integrating the density SPA $\hat{f}$ or $\tilde{\hat{f}}$. However, in a celebrated paper, Lugannani and Rice (1980) derived a simple expression for it, given by

$$\hat{F}_X(x^-) = \Phi(\hat{w}) + \phi(\hat{w})\left\{\frac{1}{\hat{w}} - \frac{1}{\hat{u}}\right\}, \qquad x \neq \mathbb{E}[X], \tag{32.9}$$

where $F_X(x^-) = \Pr(X < x)$ (strict inequality), Φ and ϕ are the cdf and pdf of the standard normal distribution, respectively,

$$\hat{w} = \operatorname{sgn}(\hat{s})\sqrt{2\hat{s}x - 2\mathbb{K}_X(\hat{s})}, \tag{32.10a}$$

and

$$\hat{u} = \begin{cases} \hat{s}\sqrt{\mathbb{K}''_X(\hat{s})}, & \text{if } x \text{ is continuous,} \\ \left(1 - \mathrm{e}^{-\hat{s}}\right)\sqrt{\mathbb{K}''_X(\hat{s})}, & \text{if } x \text{ is discrete.} \end{cases} \tag{32.10b}$$

It is important to keep in mind that (32.9) is an approximation to $\Pr(X < x)$ and not $F_X(x) = \Pr(X \leq x)$, this being only relevant when X is discrete. There are other expressions for the cdf approximation in the discrete case; see Butler (2007). If $x = \mathbb{E}[X]$, then $\mathbb{K}'_X(0) = \mathbb{E}[X]$ and $\hat{s} = 0$ is the saddlepoint for $\mathbb{E}[X]$. Thus, at the mean, $\mathbb{K}_X(0) = 0$, so that $\hat{w} = 0$, rendering $\hat{F}$ in (32.9) useless. This singularity is removable however, and the corresponding value of the cdf approximation can be shown to be

$$\hat{F}_X(\mathbb{E}[X]) = \frac{1}{2} + \frac{\mathbb{K}'''_X(0)}{6\sqrt{2\pi}\mathbb{K}''_X(0)^{3/2}}. \tag{32.11}$$

For practical use however, it is numerically wiser to use linear interpolation based on the SPA to $\mathbb{E}[X] \pm \epsilon$, where ϵ is chosen small enough to ensure high accuracy, but large enough to ensure numerical stability of (32.9) and (32.10).

The easiest way to derive (32.9) is by an application of a result due to Temme (1982), who shows that

$$\int_{-\infty}^{y} g(x)\phi(x)\mathrm{d}x \approx g(0)\Phi(y) + \phi(y)\left\{\frac{g(0) - g(y)}{y}\right\}. \tag{32.12}$$

In order to apply this, observe that an approximation to the cdf is obtained by integrating the approximate pdf, via

$$\begin{aligned} F_X(y) \approx \int_{-\infty}^{y} \hat{f}_X(x)\,\mathrm{d}x &= \int_{-\infty}^{y} \frac{1}{\sqrt{2\pi\,\mathbb{K}''_X(\hat{s})}} \exp\{\mathbb{K}_X(\hat{s}) - x\hat{s}\}\,\mathrm{d}x \\ &= \int_{-\infty}^{y} \frac{1}{\sqrt{\mathbb{K}''_X(\hat{s})}}\phi(\hat{w})\,\mathrm{d}x, \end{aligned}$$

where $\hat{w}$ is as in (32.10a). Note that both $\hat{s}$ and $\hat{w}$ depend on x through (32.3). In order to put this in the form of (32.12), we need to change variables from x to $\hat{w}$, for which we require an expression for $\mathrm{d}\hat{w}/\mathrm{d}x$. Differentiating both sides of (32.10a) with respect to x yields

$$\frac{\mathrm{d}\hat{w}}{\mathrm{d}x} = \frac{\hat{s} + (x - \mathbb{K}'_X(\hat{s}))\mathrm{d}\hat{s}/\mathrm{d}x}{\hat{w}} = \frac{\hat{s}}{\hat{w}},$$

where the last equality follows from (32.3). Thus, changing variables from x to $\hat{w}$,

$$F_X(y) \approx \int_{-\infty}^{\hat{w}(y)} \frac{\hat{w}}{\hat{s}\sqrt{\mathbb{K}''_X(\hat{s})}}\phi(\hat{w})\,\mathrm{d}\hat{w} =: \int_{-\infty}^{\hat{w}(y)} g_1(\hat{w})\phi(\hat{w})\,\mathrm{d}\hat{w},$$

where $\hat{w}(y)$ corresponds to $\hat{w}$ evaluated at the point $x = y$. This is precisely in the form of (32.12) and will yield the desired result if we can show that

$$\lim_{\hat{w}\to 0} g_1(\hat{w}) = \lim_{\hat{w}\to 0} \frac{\hat{w}}{\hat{s}\sqrt{\mathbb{K}''_X(\hat{s})}} = 1.$$

As discussed above, $\hat{w} = 0$ occurs at the point $x = \mathbb{E}[X]$, where also $\hat{s} = 0$. Assuming that the limit is finite, we can apply l'Hôpital's rule, which yields

$$\frac{1}{\sqrt{\mathbb{K}''_X(0)}} \lim_{\hat{w}\to 0} \frac{\hat{w}}{\hat{s}} = \frac{1}{\sqrt{\mathbb{K}''_X(0)}} \frac{\lim_{\hat{w}\to 0} \mathrm{d}\hat{w}/\mathrm{d}x}{\lim_{\hat{w}\to 0} \mathrm{d}\hat{s}/\mathrm{d}x} = \frac{1}{\sqrt{\mathbb{K}''_X(0)}} \frac{\lim_{\hat{w}\to 0} \hat{s}/\hat{w}}{\left(\mathbb{K}''_X(0)\right)^{-1}}, \quad (32.13)$$

where

$$\lim_{\hat{w}\to 0} \mathrm{d}\hat{s}/\mathrm{d}x = \left(\mathbb{K}''_X(0)\right)^{-1}$$

is obtained by differentiating (32.3). Now the left hand side of (32.13) is the reciprocal of the right hand side, so that the limit must be 1.

Expression (32.4) is the leading term in an asymptotic expansion; the second order approximation is given by (see Daniels 1987)

$$\tilde{f}(x) = \hat{f}(x)\left(1 + \frac{\hat{\kappa}_4}{8} - \frac{5}{24}\hat{\kappa}_3^2\right), \quad (32.14)$$

where $\hat{\kappa}_i = \mathbb{K}_X^{(i)}(\hat{s}) / \left[\mathbb{K}''_X(\hat{s})\right]^{i/2}$. Similarly, generalizing (32.9),

$$\tilde{F}(x) = \hat{F}(x) - \phi(\hat{w})\left\{\hat{u}^{-1}\left(\frac{\hat{\kappa}_4}{8} - \frac{5}{24}\hat{\kappa}_3^2\right) - \hat{u}^{-3} - \frac{\hat{\kappa}_3}{2\hat{u}^2} + \hat{w}^{-3}\right\} \quad (32.15)$$

for $x \neq \mathbb{E}[X]$. As with the first order approximation (32.9), linear interpolation around $x = \mathbb{E}[X]$ is most effective for obtaining the limit of the approximation at the mean of X.

Example 1. (Differences of iid gamma random variables.) Let $Z = X_1 - X_2$ for $X_i \overset{\text{iid}}{\sim} \mathrm{Gam}(a)$. It is straightforward to show that, for $X \sim \mathrm{Gam}(a)$, $\mathbb{M}_X(s) = (1-s)^{-a}$, $s < 1$. Then

$$\mathbb{M}_{(-X)}(s) = \mathbb{E}\left[\mathrm{e}^{s(-X)}\right] = \mathbb{M}_X(-s),$$

and it follows that the mgf of Z is given by

$$\begin{aligned}\mathbb{M}_Z(s) &= \mathbb{E}\left[\mathrm{e}^{sZ}\right] = \mathbb{E}\left[\mathrm{e}^{s(X_1-X_2)}\right] = \mathbb{M}_X(s)\,\mathbb{M}_X(-s) \\ &= (1-s)^{-a}(1+s)^{-a} = \left(1-s^2\right)^{-a}, \quad |s| < 1.\end{aligned}$$

Thus, $\mathbb{K}_Z(s) = -a\log\left(1-s^2\right)$ for $|s| < 1$,

$$\mathbb{K}'_Z(s) = \frac{2as}{1-s^2}, \quad \mathbb{K}''_Z(s) = 2a\left(1-s^2\right)^{-1} + 4as^2\left(1-s^2\right)^{-2},$$

and the saddlepoint $\hat{s}$ is given by the solution of the quadratic $xs^2 + 2as - x = 0$, i.e., $\hat{s} = \left(-a \pm \sqrt{a^2+x^2}\right)/x$ such that $|\hat{s}| < 1$. To see which of the two roots is correct, use the fact that $a > 0$ and the requirement that $-1 < s < 1$. To confirm that the root with the $+$ is the correct one, we need to show

$$-1 < \frac{-a + \sqrt{a^2+x^2}}{x} < 1.$$

For $x > 0$, simple manipulations lead to

$$a^2 - 2ax + x^2 = (a-x)^2 < a^2 + x^2 < (x+a)^2 = 2ax + a^2 + x^2$$

or $-2ax + x^2 < x^2 < 2ax + x^2$, which is true. Similarly, for $x < 0$, we get $-2ax + x^2 > x^2 > 2ax + x^2$, which is true. To normalize the saddlepoint density of Z, use the fact that $x = \mathbb{K}'_Z(\hat{s})$ to give $\mathrm{d}x = \mathbb{K}''_Z(\hat{s})\,\mathrm{d}\hat{s}$. Then, using the above expressions for $\mathbb{K}_Z(\hat{s})$, $\mathbb{K}'_Z(\hat{s})$ and $\mathbb{K}''_Z(\hat{s})$, we easily get

$$\begin{aligned} S(a) &= \int_{-\infty}^{\infty} \hat{f}(x)\,\mathrm{d}x \\ &= \frac{1}{\sqrt{2\pi}} \int_{-1}^{1} \sqrt{\mathbb{K}''_Z(\hat{s})} \exp\left\{\mathbb{K}_Z(\hat{s}) - \hat{s}\mathbb{K}'_Z(\hat{s})\right\} \mathrm{d}\hat{s} \qquad (32.16) \\ &= \frac{1}{\sqrt{2\pi}} \int_{-1}^{1} \sqrt{2a\left(1-t^2\right)^{-1-2a} + 4at^2\left(1-t^2\right)^{-2(1+a)}} \exp\left\{\frac{-2at^2}{1-t^2}\right\} \mathrm{d}t, \end{aligned}$$

which can be numerically approximated for each value $a > 0$. Figure 32.1 shows $S(a)$ for $a = 1, 2, \ldots, 50$, and shows that, as $a \to \infty$, $S(a) \uparrow 1$, but rather slowly.

Instead of integrating each time, one could attempt to fit a low order polynomial or some suitable function in a to the points in the figure. Some trial and error leads to the specification

$$\begin{aligned} S(a) \approx\ & 0.57801 + 0.0036746a - 0.18594a^{1/2} + 0.39464a^{1/3} \\ & + 0.092409\left(1 - e^{-a}\right), \end{aligned}$$

which exhibits a maximal absolute error of about 0.00045.

It is noteworthy that the SPA has no numeric trouble at all when a is close to one or for density values in the tails, as does the inversion of the cf (32.6). ■

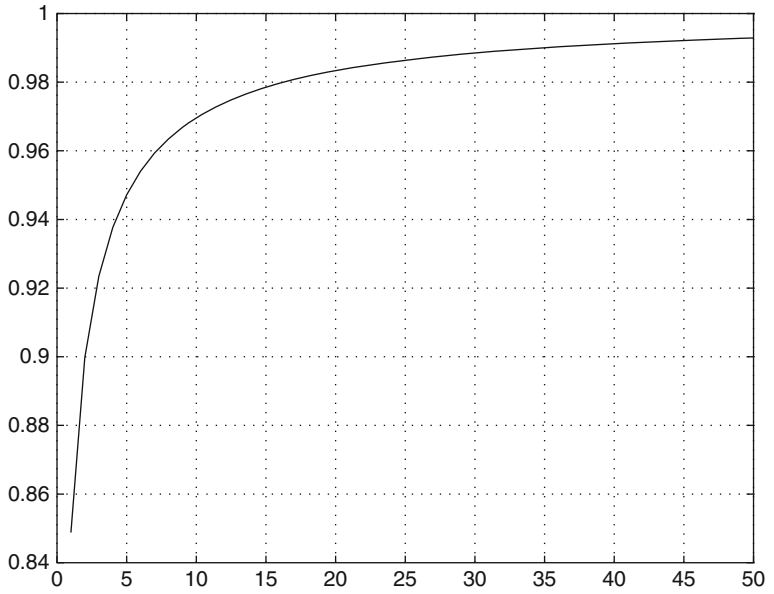


Fig. 32.1 Plot of $S(a)$ in (32.16) versus a

32.3 Partial Expectations

Suppose that random variable X has a density $f_X(x)$. The *partial expectation*

$$G_X(x) = \int_{-\infty}^{x} y f_X(y) \mathrm{d}y \tag{32.17}$$

plays a vital role in certain applications in finance, for example in option pricing. Another important application is the computation of a risk measure known as the *expected shortfall*. This will be considered in more detail in Sect. 32.6 below.

As before for the pdf and cdf, we first illustrate how $G_X(x)$ can be recovered exactly from the moment generating function: in Broda and Paolella (2009a), it is shown that

$$G_X(x) = \frac{\mathbb{M}'_X(0)}{2} - \frac{1}{\pi} \int_0^{\infty} \mathrm{Im}\left[\mathbb{M}'_X(is)\mathrm{e}^{-isx}\right] \frac{\mathrm{d}s}{s}. \tag{32.18}$$

As regards the saddlepoint approximation, Martin (2006) shows that an SPA to $G_X(x)$ is given by

$$\hat{G}_X(x) = \Phi(\hat{w})\,\mu + \phi(\hat{w})\left(\frac{\mu}{\hat{w}} - \frac{x}{\hat{u}}\right), \tag{32.19}$$

where $\mu = \mathbb{K}'_X(0)$, and $\hat{w}$ and $\hat{u}$ are as in (32.10).

As for the SPA to the cdf, the result is most conveniently derived by an application of Temme's approximation: the partial mean can be approximated as

$$G_X(y) \approx \int_{-\infty}^{y} x\hat{f}_X(x)\,\mathrm{d}x = \int_{-\infty}^{y} \frac{x}{\sqrt{2\pi\,\mathbb{K}_X''(\hat{s})}} \exp\{\mathbb{K}_X(\hat{s}) - x\hat{s}\}\,\mathrm{d}x$$
$$= \int_{-\infty}^{y} \frac{x}{\sqrt{\mathbb{K}_X''(\hat{s})}} \phi(\hat{w})\,\mathrm{d}x.$$

Changing variables from x to $\hat{w}$ as before yields

$$G_X(y) \approx \int_{-\infty}^{\hat{w}(y)} \frac{x\hat{w}}{\hat{s}\sqrt{\mathbb{K}_X''(\hat{s})}} \phi(\hat{w})\,\mathrm{d}\hat{w} =: \int_{-\infty}^{\hat{w}(y)} g_2(\hat{w})\phi(\hat{w})\,\mathrm{d}\hat{w},$$

to which the approximation can be applied. The result follows by noting that

$$\lim_{\hat{w}\to 0} g_2(\hat{w}) = \lim_{\hat{w}\to 0} x g_1(\hat{w}) = \lim_{\hat{w}\to 0} x = \mu,$$

as $\hat{w} = 0$ occurs at the point $x = \mu$.

Again, (32.19) is the leading term in an asymptotic expansion; the second order approximation is derived in Broda and Paolella (2009c) as

$$\tilde{G}_X(x) = \hat{G}_X(x) - \phi(\hat{w})\left(\frac{x}{\hat{u}}\left\{\frac{1}{8}\hat{\kappa}_4 - \frac{5}{24}\hat{\kappa}_3^2\right\} - \frac{x}{\hat{u}^3} - \frac{x\hat{\kappa}_3}{2\hat{u}^2} + \frac{\mu}{\hat{w}^3} + \frac{1}{\hat{s}\hat{u}}\right).$$

Butler and Wood (2004) derive the more convenient looking

$$\tilde{G}_X(x) = \hat{G}_X(x) - \phi(\hat{w})\left(\frac{\mu}{\hat{w}^3} - \frac{x}{\hat{w}^3} + \frac{1}{\hat{s}\hat{u}}\right). \tag{32.20}$$

While technically only a first-order approximation, its accuracy in applications can be quite remarkable, as illustrated in Sect. 32.6.

32.4 Vectors of Random Variables

The pmf or pdf saddlepoint approximation (32.4) generalizes naturally to the multivariate case. For a d-dimensional random vector $\mathbf{X}$ having joint cgf $\mathbb{K}$ with gradient $\mathbb{K}'$ and Hessian $\mathbb{K}''$, the approximation is given by

$$\hat{f}_{\mathbf{X}}(\mathbf{x}) = (2\pi)^{-d/2}|\mathbb{K}''(\hat{\mathbf{s}})|^{-1/2} \exp\left(\mathbb{K}(\hat{\mathbf{s}}) - \hat{\mathbf{s}}'\mathbf{x}\right), \tag{32.21}$$

where the multivariate saddlepoint satisfies $\mathbb{K}'(\hat{\mathbf{s}}) = \mathbf{x}$ for $\hat{\mathbf{s}}$ in the convergence region of the mgf of $\mathbf{X}$. Saddlepoint cdf approximations for the general multivariate setting are less straightforward, and we restrict ourselves to the bivariate case. Section 32.4.1 illustrates the SPA for the conditional distribution $X \mid Y$, Sect. 32.4.2 details the cdf approximation in the continuous bivariate case, and Sect. 32.4.3 shows a method for approximating the marginal distribution.

32.4.1 Conditional Distributions

Let $\mathbb{K}(s,t)$ denote the joint cgf for continuous random variables X and Y, assumed convergent over $\mathcal{S}$, an open neighborhood of $(0,0)$. The gradient of $\mathbb{K}$ is $\mathbb{K}'(s,t) = (\mathbb{K}'_s(s,t), \mathbb{K}'_t(s,t))'$, where

$$\mathbb{K}'_s(s,t) := \frac{\partial}{\partial s}\mathbb{K}(s,t), \quad \mathbb{K}'_t(s,t) := \frac{\partial}{\partial t}\mathbb{K}(s,t),$$

and

$$\mathbb{K}''(s,t) := \begin{bmatrix} \mathbb{K}''_{ss}(s,t) & \mathbb{K}''_{st}(s,t) \\ \mathbb{K}''_{ts}(s,t) & \mathbb{K}''_{tt}(s,t) \end{bmatrix}, \quad \mathbb{K}''_{ss}(s,t) = \frac{\partial^2}{\partial s^2}\mathbb{K}(s,t), \quad \text{etc.}, \tag{32.22}$$

is the Hessian.

Let $\mathcal{X}$ be the interior of the convex hull of the joint support of (X,Y). Skovgaard (1987) derived a double-saddlepoint approximation for the conditional cdf of X at x given $Y = y$, for $(x,y) \in \mathcal{X}$. In this case, the gradient is a one-to-one mapping from the convergence strip $\mathcal{S}$ onto $\mathcal{X}$. As the name implies, there are two saddlepoints to compute when using this approximation. The first is the unique preimage of (x,y) in $\mathcal{S}$, denoted $(\tilde{s},\tilde{t})$, computed as the solutions to

$$\mathbb{K}'_s(\tilde{s},\tilde{t}) = x, \qquad \mathbb{K}'_t(\tilde{s},\tilde{t}) = y. \tag{32.23}$$

This is the numerator saddlepoint in the approximation. The second saddlepoint is found by fixing $s = 0$ and solving $\mathbb{K}'_t(0,\tilde{t}_0) = y$ for the unique value of $\tilde{t}_0$ in $\{t : (0,t) \in \mathcal{S}\}$. This is the denominator saddlepoint. The cdf approximation is then given by

$$\Pr(X \le x \mid Y = y) \approx \Phi(\tilde{w}) + \phi(\tilde{w})\left\{\tilde{w}^{-1} - \tilde{u}^{-1}\right\}, \qquad \tilde{s} \neq 0, \tag{32.24}$$

where

$$\tilde{w} = \operatorname{sgn}(\tilde{s})\sqrt{2}\sqrt{\tilde{s}x + \tilde{t}y - \mathbb{K}(\tilde{s},\tilde{t}) - \tilde{t}_0 y + \mathbb{K}(0,\tilde{t}_0)}, \tag{32.25}$$

$$\tilde{u} = \tilde{s}\sqrt{\left|\mathbb{K}''(\tilde{s},\tilde{t})\right| \,/\, \mathbb{K}''_{tt}(0,\tilde{t}_0)}. \tag{32.26}$$

To see that $\tilde{w}$ makes sense, note that

$$\sup_{(s,t)\in\mathcal{S}} [sx + ty - \mathbb{K}(s,t)] \tag{32.27}$$

occurs at $(\tilde{s}, \tilde{t})$, as (32.23) is its critical value. Also, the supremum (32.27) must be greater than the constrained supremum over $\{(0,t) \in \mathcal{S}\}$, which occurs at $\tilde{t}_0$. Thus, the term inside the square root is positive. The term $\tilde{u}$ is also well-defined because $\mathbb{K}''(s,t)$ is positive definite for all $(s,t) \in \mathcal{S}$.

32.4.2 Bivariate CDF Approximation

Wang (1990) shows that a saddlepoint approximation to the cdf of $\mathbf{X} = (X, Y)$ at $\mathbf{x} = (x, y)$ is given by

$$\hat{F}_{\mathbf{X}}(\mathbf{x}) = \Phi_2(\tilde{x}_1, \tilde{y}_1, \tilde{\rho}) + \Phi(\tilde{w}_0)\tilde{n} + \Phi(\tilde{w})\tilde{n}_0 + \tilde{n}\tilde{n}_0, \tag{32.28}$$

where

$$\tilde{x}_1 = \operatorname{sgn}(\tilde{t}_0)\sqrt{2\big(\tilde{t}_0 y - \mathbb{K}(0,\tilde{t}_0)\big)}, \quad \tilde{w}_0 = \operatorname{sgn}(\tilde{t})\sqrt{2\big(\mathbb{K}(\tilde{s},0) - \mathbb{K}(\tilde{s},\tilde{t}) + \tilde{t}y\big)},$$

$$\tilde{y}_1 = \frac{\tilde{w} - b\tilde{x}_1}{\sqrt{1+b^2}}, \quad \tilde{\rho} = -\frac{b}{\sqrt{1+b^2}}, \quad b = \frac{\tilde{w}_0 - \tilde{x}_1}{\tilde{w}}, \quad \tilde{n} = \phi(\tilde{w})\left[\frac{1}{\tilde{w}} - \frac{1}{\tilde{u}}\right],$$

$$\tilde{n}_0 = \phi(\tilde{x}_1)\left[\frac{1}{w_0} - \frac{1}{\tilde{u}_0}\right], \quad \tilde{u} = \tilde{s}\sqrt{\frac{|\mathbb{K}''(\tilde{s},\tilde{t})|}{\mathbb{K}''_{tt}(\tilde{s},\tilde{t})}} \quad \tilde{u}_0 = \tilde{t}\sqrt{\mathbb{K}''_{tt}(\tilde{s},\tilde{t})},$$

$\tilde{w}$ and the saddlepoints $(\tilde{s}, \tilde{t})$ and $\tilde{t}_0$ are as in the Skovgaard approximation from Sect. 32.4.1, and $\Phi_2(x, y, \rho)$ denotes the bivariate standard normal c.d.f. with correlation ρ. See Paolella (2007, Sect. 5.2.2) for further details and examples of this case. Kolassa (2003) gives an approximation valid for $d > 2$.

32.4.3 Marginal Distributions

Let $\mathbf{X} = (X_1, X_2)$ be a continuous bivariate random variable with joint cumulant generating function $\mathbb{K}(\mathbf{t}) \equiv \mathbb{K}(s,t)$. Consider a bijection

$$\mathbf{Y} = (Y_1, Y_2) = g^{-1}(\mathbf{X}) = (g_1^{-1}(\mathbf{X}), g_2^{-1}(\mathbf{X}))',$$

so that $\mathbf{X} = g(\mathbf{Y}) = (g_1(\mathbf{Y}), g_2(\mathbf{Y}))'$, and denote by

$$\nabla_{y_i} g(\mathbf{Y}) = \left(\frac{\partial g_1}{\partial Y_i}, \frac{\partial g_2}{\partial Y_i} \right)', \quad \nabla^2_{y_i} g(\mathbf{Y}) = \left(\frac{\partial^2 g_1}{\partial Y_i^2}, \frac{\partial^2 g_2}{\partial Y_i^2} \right)' \quad i \in \{1, 2\},$$

the vectors of first and second derivatives of g with respect to Y_i. Interest centers on the marginal distribution of Y_1. Daniels and Young (1991) show that saddlepoint approximations to the marginal pdf and cdf of y_1 are given by

$$\tilde{f}_{Y_1}(y_1) = \phi(\tilde{w})/\tilde{u} \tag{32.29}$$

and

$$\tilde{F}_{Y_1}(y_1) = \Phi(\tilde{w}) + \phi(\tilde{w}) \left(\frac{1}{\tilde{w}} - \frac{\tilde{d}}{\tilde{u}} \right), \tag{32.30}$$

respectively, where

$$\tilde{w} = \sqrt{2\big(\tilde{\mathbf{t}}' g(\tilde{\mathbf{y}}) - \mathbb{K}(\tilde{\mathbf{t}})\big)}\, \mathrm{sgn}(y_1 - \alpha), \quad \tilde{\mathbf{y}} = (y_1, \tilde{y}_2),$$

$$\alpha = g_1^{-1}(\mathbb{K}'(\mathbf{0})), \quad \tilde{d} = \big(\tilde{\mathbf{t}}' \nabla_{y_1} g(\tilde{\mathbf{y}})\big)^{-1},$$

$$\tilde{u} = \frac{\sqrt{\det\big(\mathbb{K}''(\tilde{\mathbf{t}})\big) \left[\nabla_{y_2} g(\tilde{\mathbf{y}})' \big(\mathbb{K}''(\tilde{\mathbf{t}})\big)^{-1} \nabla_{y_2} g(\tilde{\mathbf{y}}) + \tilde{\mathbf{t}}' \nabla^2_{y_2} g(\tilde{\mathbf{y}}) \right]}}{\det\big(\partial g / \partial \mathbf{y}(\tilde{\mathbf{y}})\big)},$$

and, for each value of y_1, $\tilde{\mathbf{t}}$ and $\tilde{y}_2$ solve the system

$$\mathbb{K}'(\tilde{\mathbf{t}}) = g(\tilde{\mathbf{y}})$$
$$\tilde{\mathbf{t}}' \nabla_{y_2} g(\tilde{\mathbf{y}}) = 0.$$

As usual in saddlepoint applications, if additional accuracy is desired, the pdf approximation can be renormalized by numerically integrating it over its support.

Example 2. (Student's t) Let $X_1 \sim \mathrm{N}(0, 1)$, independent of $X_2 \sim \chi^2_n$, and let $\mathbf{X} = g(\mathbf{Y}) = (Y_1 Y_2, n Y_2^2)$, so that $(Y_1, Y_2) = g^{-1}(X_1, X_2) = \big(X_1 / \sqrt{X_2/n}, \sqrt{X_2/n}\big)'$, and Y_1 has a Student's t distribution with n degrees of freedom. It might come as a surprise that an SPA is available for this distribution, as its mgf obviously does not exist. The trick is to use the definition of the t random variable as a ratio of random variables, the components of which do in fact have mgfs. The joint cgf of (X_1, X_2) is, from independence,

$$\mathbb{K}(\mathbf{t}) = \mathbb{K}_{X_1}(s) + \mathbb{K}_{X_2}(t) = \frac{1}{2} s^2 - \frac{n}{2} \log(1 - 2t),$$

where $\mathbb{K}_{X_1}$ and $\mathbb{K}_{X_2}$ are the cgfs of X_1 and X_2, respectively. The saddlepoint $(\tilde{\mathbf{t}}, \tilde{y}_2) = (\tilde{s}, \tilde{t}, \tilde{y}_2)$, $\tilde{t} < \frac{1}{2}$, $\tilde{y}_2 > 0$, solves the system of equations

$$s = y_1 y_2, \quad \frac{1}{(1-2t)} = y_2^2, \quad s y_1 + 2nt y_2 = 0,$$

which can be solved to give

$$\tilde{y}_2 = \sqrt{n/(y_1^2+n)}, \quad \tilde{s} = y_1 \tilde{y}_2, \quad \tilde{t} = -\frac{y_1 \tilde{s}}{2n\tilde{y}_2}.$$

The other required quantities are given by

$$\tilde{d} = (y_1 \tilde{y}_2^2)^{-1}, \quad \tilde{u} = \tilde{y}_2^{-1}, \quad w = \sqrt{-2n \log(\tilde{y}_2)}\ \mathrm{sgn}(y_1),$$

and, plugging in, the pdf approximation becomes

$$\hat{f}_t(y_1; n) = \frac{1}{\sqrt{2\pi}} \left(\frac{n}{y_1^2+n} \right)^{\frac{1}{2}(n+1)}, \tag{32.31}$$

which is exact after re-normalization. The fact that the re-normalized SPA is exact for the usual Student's t distribution is remarkable, and also serves to suggest that it will be highly accurate in the noncentral case, considered next. ■

Example 3. (Noncentral Student's t). The singly noncentral t is fundamental in statistics, as it is the distribution associated with the power of the classic t test. It has also gained recognition as a useful ad hoc distribution for modeling asset returns, given its fat tails and asymmetry. In particular, it was first advocated by Harvey and Siddique (1999) and used subsequently by others for modeling financial asset returns, e.g., Tsionas (2002) and Broda and Paolella (2007), and for even less common applications, such as modeling real-estate values (Coleman and Mansour 2005). Use of the MLE for point estimation will be numerically costly with the noncentral t distribution (see the equations below for the pdf), a drawback explicitly mentioned in Premaratne and Bera (2000, p. 6), who address it by using the Pearson type IV distribution as an approximation to it. However, the SPA is applicable, and even results in a closed form solution to the saddlepoint equation (in both the singly and doubly noncentral cases), so that computation of the MLE is nearly as fast as using the usual Student's t distribution.

Let $X \sim \mathrm{N}(\mu, 1)$ independent of $Y \sim \chi^2(k, \boldsymbol{\theta})$. Random variable $T = X/\sqrt{Y/k}$ is said to follow a doubly noncentral t distribution with k degrees of freedom, numerator noncentrality parameter μ and denominator noncentrality parameter $\boldsymbol{\theta}$. We write $T \sim t''(k, \mu, \boldsymbol{\theta})$. If $\boldsymbol{\theta} = 0$, then T is singly noncentral t with noncentrality parameter μ, and we write $T \sim t'(k, \mu)$. The pdf and cdf of the singly noncentral t can be expressed as indefinite integrals or infinite sums. In particular,

$$F_T(t; k, \mu) = \frac{2^{-k/2+1} k^{k/2}}{\Gamma(k/2)} \int_0^\infty \Phi(tz; \mu)\, z^{k-1} \exp\left\{ -\frac{1}{2} k z^2 \right\} \mathrm{d}z, \tag{32.32}$$

where $\Phi(z;\mu)$ is the cdf of the normal distribution with mean μ and variance one, at z. Differentiating (32.32) using Leibniz' rule yields an expression for the pdf as

$$f_T(t;k,\mu) = K\int_0^\infty z^k \exp\left\{-\frac{1}{2}\left[(tz-\mu)^2 + kz^2\right]\right\} \mathrm{d}z, \quad K = \frac{2^{-k/2+1}k^{k/2}}{\Gamma(k/2)\sqrt{2\pi}}. \tag{32.33}$$

Also, for the pdf,

$$\begin{aligned} f_T(t;k,\mu) = {} & \mathrm{e}^{-\mu^2/2}\frac{\Gamma((k+1)/2)\,k^{k/2}}{\sqrt{\pi}\,\Gamma(k/2)}\left(\frac{1}{k+t^2}\right)^{\frac{k+1}{2}} \\ & \times\left(\sum_{i=0}^{\infty}\frac{(t\mu)^i}{i!}\left(\frac{2}{t^2+k}\right)^{i/2}\frac{\Gamma((k+i+1)/2)}{\Gamma((k+1)/2)}\right), \end{aligned} \tag{32.34}$$

while for the cdf, we calculate $\Pr(T \le t) = \Pr(T \le 0) + \Pr(0 \le T \le t)$ for $t > 0$, with $\Pr(T \le 0) = \Phi(-\mu;0)$ and

$$\Pr(0 \le T \le t) = \frac{1}{2}\mathrm{e}^{-\mu^2/2}\sum_{i=0}^{\infty}\mu^i\frac{(1/2)^{i/2}}{\Gamma(i/2+1)}\bar{B}_{m(t)}\left(\frac{1+i}{2},\frac{k}{2}\right), \quad t > 0, \tag{32.35}$$

where $\bar{B}$ is the incomplete beta ratio and $m(t) = t^2/\left(k+t^2\right)$. The derivation of all these expressions, and equations for the cdf with $t < 0$, are given in Paolella (2007, Sect. 10.4).

Now let $x_1 \sim \mathrm{N}(\mu,1)$, independent of $x_2 \sim \chi^2(k,\boldsymbol{\theta})$, and let $\mathbf{x} = g(\mathbf{y}) = (y_1y_2, y_2^2k)$, so that $(y_1,y_2) = g^{-1}(x_1,x_2) = \left(x_1/\sqrt{x_2/k}, \sqrt{x_2/k}\right)'$ and $y_1 \sim t''(k,\mu,\boldsymbol{\theta})$. The joint cumulant generating function of (x_1,x_2) is, from independence,

$$K(\mathbf{t}) = K_{x_1}(t_1) + K_{x_2}(t_2) = t_1\mu + \frac{1}{2}t_1^2 - \frac{k}{2}\log(1-2t_2) + \frac{t_2\boldsymbol{\theta}}{1-2t_2},$$

where K_{x_1} and K_{x_2} are the cumulant generating functions of x_1 and x_2, respectively. The saddlepoint $(\hat{\mathbf{t}},\hat{y}_2) = (\hat{t}_1,\hat{t}_2,\hat{y}_2)$, $\hat{t}_2 < \frac{1}{2}$, $\hat{y}_2 > 0$, solves the system of equations

$$\mu + t_1 = y_1y_2, \quad \frac{k}{(1-2t_2)} + \frac{\boldsymbol{\theta}}{(1-2t_2)^2} = ny_2^2, \quad t_1y_1 + 2nt_2y_2 = 0.$$

Straightforward calculation reveals that

$$\hat{t}_1 = -\mu + y_1\hat{y}_2, \quad \hat{t}_2 = -\frac{y_1\hat{t}_1}{2k\hat{y}_2}, \tag{32.36}$$

and $\hat{y}_2$ solves the cubic $s(y_2) := a_3y_2^3 + a_2y_2^2 + a_1y_2 + a_0 = 0$, where

$$a_3 = y_1^4 + 2ny_1^2 + k^2, \quad a_2 = -2y_1^3\mu - 2y_1 k\mu, \quad a_1 = y_1^2\mu^2 - ny_1^2 - k^2 - \boldsymbol{\theta} k,$$

and $a_0 = y_1 k\mu$. Upon defining

$$c_2 = \frac{a_2}{a_3}, \quad c_1 = \frac{a_1}{a_3}, \quad c_0 = \frac{a_0}{a_3}, \quad q = \frac{1}{3}c_1 - \frac{1}{9}c_2^2,$$

$$r = \frac{1}{6}(c_1 c_2 - 3c_0) - \frac{1}{27}c_2^3, \quad m = q^3 + r^2, \quad \text{and} \quad s_{1,2} = (r \pm \sqrt{m})^{1/3},$$

the roots of the cubic are given by

$$z_1 = (s_1 + s_2) - \frac{c_2}{3} \qquad \text{and} \qquad z_{2,3} = -\frac{1}{2}(s_1 + s_2) - \frac{c_2}{3} \pm \frac{i\sqrt{3}}{2}(s_1 - s_2).$$

The saddlepoint solution is always z_1, as proved in Broda and Paolella (2007). It can also be expressed as

$$\hat{y}_2 = \sqrt{-4q}\cos\left(\cos^{-1}\left(r/\sqrt{-q^3}\right)/3\right) - \frac{c_2}{3}, \tag{32.37}$$

thus avoiding complex arithmetic.

With

$$\nabla_{y_1} g(\mathbf{y}) = (y_2, 0)', \quad \nabla_{y_2} g(\mathbf{y}) = (y_1, 2ny_2)',$$
$$\nabla_{y_2}^2 g(\mathbf{y}) = (0, 2k)', \quad K''(\mathbf{t}) = \operatorname{diag}\left(1, 2k(1-2t_2)^{-2} + 4\boldsymbol{\theta}(1-2t_2)^{-3}\right),$$
$$\det\left[\partial g/\partial \mathbf{y}\right] = \det\left[\nabla_{y_1} g(\mathbf{y}),\ \nabla_{y_2} g(\mathbf{y})\right] = 2ny_2^2$$

and after some simplification, the quantities entering approximations (32.29) and (32.30) take the simple form

$$d = (\hat{t}_1 \hat{y}_2)^{-1}, \qquad u = \sqrt{(y_1^2 + 2k\hat{t}_2)(2k\nu^2 + 4\boldsymbol{\theta}\nu^3) + 4k^2\hat{y}_2^2} \Big/ \left(2k\hat{y}_2^2\right),$$

$$w = \sqrt{-\mu\hat{t}_1 - k\log\nu - 2\boldsymbol{\theta}\nu\hat{t}_2}\,\operatorname{sgn}\left(y_1 - \alpha\right), \qquad \alpha = \mu/\sqrt{1 + \boldsymbol{\theta}/k},$$

where $\nu = (1 - 2\hat{t}_2)^{-1}$, and $\hat{\mathbf{t}} = (\hat{t}_1, \hat{t}_2)'$ and $\hat{y}_2$ are given by (32.36) and (32.37), respectively. In the singly noncentral case with $\boldsymbol{\theta} = 0$, these reduce to

$$u = \sqrt{(\mu y_1 \hat{y}_2 + 2k)/(2k)}/\hat{y}_2, \qquad \text{and} \qquad w = \sqrt{-\mu\hat{t}_1 - 2k\log(\hat{y}_2)}\,\operatorname{sgn}(y_1 - \mu),$$

where d, $\hat{t}_1$ and $\hat{t}_2$ are as before, and

$$\hat{y}_2 = \frac{\mu y_1 + \sqrt{4k(y_1^2 + k) + \mu^2 y_1^2}}{2(y_1^2 + k)}.$$

This is the approximation given in DiCiccio and Martin (1991, p. 897, Eq. (18)). In the central case with $\mu = 0$, we get (32.31). ■

32.5 Quadratic Forms in Gaussian Vectors

The map $\mathbb{R}^n \to \mathbb{R}$ is a *quadratic form* if it can be expressed as $\mathbf{x} \mapsto \mathbf{x}'\mathbf{A}\mathbf{x}$, where $\mathbf{A}$ is an $n \times n$ real symmetric matrix. Let $\mathbf{A}$ be such a matrix, and $\mathbf{X} \sim \mathrm{N}_n(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with $\boldsymbol{\Sigma} > 0$. The scalar random variable $Y = \mathbf{X}'\mathbf{A}\mathbf{X}$ is referred to as a *quadratic form (in normal variables)*. Quadratic forms arise ubiquitously when working with regression and time-series models. If $Y = \mathbf{X}'\mathbf{B}\mathbf{X}$ with $\mathbf{B}$ not symmetric, observe that $Y' = Y$, i.e., $(\mathbf{X}'\mathbf{B}\mathbf{X})' = \mathbf{X}'\mathbf{B}'\mathbf{X}$ so that $Y = \mathbf{X}'\left(\mathbf{B} + \mathbf{B}'\right)\mathbf{X}/2 = \mathbf{X}'\mathbf{A}\mathbf{X}$ with $\mathbf{A} = (\mathbf{B} + \mathbf{B}')/2$. Matrix $\mathbf{A}$ is symmetric, so there is no loss in generality in working with symmetric matrices.

Let $Y = \mathbf{X}'\mathbf{A}\mathbf{X}$ with $\mathbf{X} \sim \mathrm{N}_n(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, $\boldsymbol{\Sigma} > 0$. Let $\boldsymbol{\Sigma}^{\frac{1}{2}}$ be a matrix such that $\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\Sigma}^{\frac{1}{2}} = \boldsymbol{\Sigma}$. Recall that $\boldsymbol{\Sigma}^{\frac{1}{2}}$ is easily computed using the spectral decomposition and is symmetric and positive-definite. Then $\boldsymbol{\Sigma}^{-\frac{1}{2}}\mathbf{X} \sim \mathrm{N}_n\left(\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\mu}, \mathbf{I}\right)$. Next, write

$$Y = \mathbf{X}'\mathbf{A}\mathbf{X} = \mathbf{X}'\mathbf{I}\mathbf{A}\mathbf{I}\mathbf{X} = \mathbf{X}'\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\Sigma}^{\frac{1}{2}}\mathbf{A}\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\Sigma}^{-\frac{1}{2}}\mathbf{X}, \tag{32.38}$$

and let the spectral decomposition of $\boldsymbol{\Sigma}^{\frac{1}{2}}\mathbf{A}\boldsymbol{\Sigma}^{\frac{1}{2}}$ be given by $\mathbf{P}\boldsymbol{\Lambda}\mathbf{P}'$, where $\mathbf{P}$ is an orthogonal matrix and $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n) = \mathrm{Eig}(\boldsymbol{\Sigma}^{\frac{1}{2}}\mathbf{A}\boldsymbol{\Sigma}^{\frac{1}{2}}) = \mathrm{Eig}(\boldsymbol{\Sigma}\mathbf{A}) = \mathrm{Eig}(\mathbf{A}\boldsymbol{\Sigma})$. Then, from (32.38),

$$F_Y(y) = \Pr\left(\mathbf{X}'\boldsymbol{\Sigma}^{-\frac{1}{2}}\mathbf{P}\boldsymbol{\Lambda}\mathbf{P}'\boldsymbol{\Sigma}^{-\frac{1}{2}}\mathbf{X} \le y\right) = \Pr\left(\mathbf{W}'\boldsymbol{\Lambda}\mathbf{W} \le y\right), \tag{32.39}$$

where

$$\mathbf{W} = \mathbf{P}'\boldsymbol{\Sigma}^{-\frac{1}{2}}\mathbf{X} \sim \mathrm{N}(\boldsymbol{\nu}, \mathbf{I}_n), \qquad \boldsymbol{\nu} = \mathbf{P}'\boldsymbol{\Sigma}^{-\frac{1}{2}}\boldsymbol{\mu} = (\nu_1, \ldots, \nu_n)'. \tag{32.40}$$

This decomposition is sometimes referred to as the principle axis theorem; see Scheffé (1959, p. 397).

Recall the definition of a noncentral χ^2 random variable: If $(X_1, \ldots, X_n) \sim \mathrm{N}_n(\boldsymbol{\mu}, \mathbf{I})$, with $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)'$, then $X = \sum_{i=1}^n X_i^2$ follows a noncentral χ^2 distribution with n degrees of freedom and noncentrality parameter $\boldsymbol{\theta} = \sum_{i=1}^n \mu_i^2$. We write $X \sim \chi^2(n, \boldsymbol{\theta})$. Its mgf is (see, e.g., Paolella 2007, p. 346)

$$\mathbb{M}_X(s) = (1-2s)^{-n/2} \exp\left\{\frac{s\boldsymbol{\theta}}{1-2s}\right\}, \quad s < 1/2. \tag{32.41}$$

From the cgf $\mathbb{K}_X(s)$, it follows that $\mathbb{K}'_X(s) = n(1-2s)^{-1} + \boldsymbol{\theta}(1-2s)^{-2}$, with higher order terms easily computed, from which we obtain

$$\kappa_i = \mathbb{K}_X^{(i)}(0) = 2^{i-1}(i-1)!\,(n + i\boldsymbol{\theta}).$$

From (32.39) and (32.40),

$$\mathbf{W}'\boldsymbol{\Lambda}\mathbf{W} = \sum_{i=1}^{\text{rank}(\mathbf{A})} \lambda_i W_i^2, \qquad W_i^2 \overset{\text{ind}}{\sim} \chi^2\left(1, \nu_i^2\right), \tag{32.42}$$

is a weighted sum of rank $(\mathbf{A})$ independent noncentral χ^2 random variables, each with one degree of freedom. Its SPA is straightforward, and we consider a more general case: Let $X_i \overset{\text{ind}}{\sim} \chi^2(n_i, \boldsymbol{\theta}_i)$ and define $X = \sum_{i=1}^k a_i X_i$, $a_i \neq 0$. From (32.41),

$$\mathbb{M}_{X_i}(a_i s) = (1 - 2a_i s)^{-n_i/2} \exp\left\{\frac{a_i s \boldsymbol{\theta}_i}{1 - 2a_i s}\right\}, \quad 1 - 2a_i s > 0, \tag{32.43}$$

so that

$$\mathbb{M}_X(s) = \prod_{i=1}^k \mathbb{M}_{X_i}(a_i s) \tag{32.44}$$

for s in a sufficiently small neighborhood of zero. For convenience, let $\vartheta_i = \vartheta_i(s) = (1 - 2sa_i)^{-1}$. Then straightforward calculation yields

$$\mathbb{K}_X(s) = \frac{1}{2}\sum_{i=1}^k n_i \ln \vartheta_i + s\sum_{i=1}^k a_i \boldsymbol{\theta}_i \vartheta_i, \quad \mathbb{K}'_X(s) = \sum_{i=1}^k a_i \vartheta_i (n_i + \boldsymbol{\theta}_i \vartheta_i)$$

and

$$\mathbb{K}''_X(s) = 2\sum_{i=1}^k a_i^2 \vartheta_i^2 (n_i + 2\boldsymbol{\theta}_i \vartheta_i), \quad \mathbb{K}'''_X(s) = 8\sum_{i=1}^k a_i^3 \vartheta_i^3 (n_i + 3\boldsymbol{\theta}_i \vartheta_i),$$

and $\mathbb{K}_X^{(4)}(s) = 48\sum_{i=1}^k a_i^4 \vartheta_i^4 (n_i + 4\boldsymbol{\theta}_i \vartheta_i)$, from which the SPA can be calculated once $\hat{s}$ is numerically determined.

Example 4. (Distribution of Sample Variance) Let $\mathbf{X} = (X_1, \ldots, X_n)' \sim \mathrm{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with $\boldsymbol{\Sigma} > 0$ and consider the sample variance of $X_1, \ldots, X_n$. Let $\mathbf{1}_n$ denote a length-n column vector of ones, $\mathbf{J}_n$ an $n \times n$ matrix of ones, and

$$\mathbf{M} = \mathbf{I}_n - \mathbf{1}_n\left(\mathbf{1}_n'\mathbf{1}_n\right)^{-1}\mathbf{1}_n' = \mathbf{I}_n - n^{-1}\mathbf{J}_n, \tag{32.45}$$

a rank $m = n - 1$ projection matrix (with one eigenvalue equal to zero and $n - 1$ eigenvalues equal to one). Thus (or easily directly confirmed), $\mathbf{M}' = \mathbf{M}$ and $\mathbf{MM} = \mathbf{M}$, so that

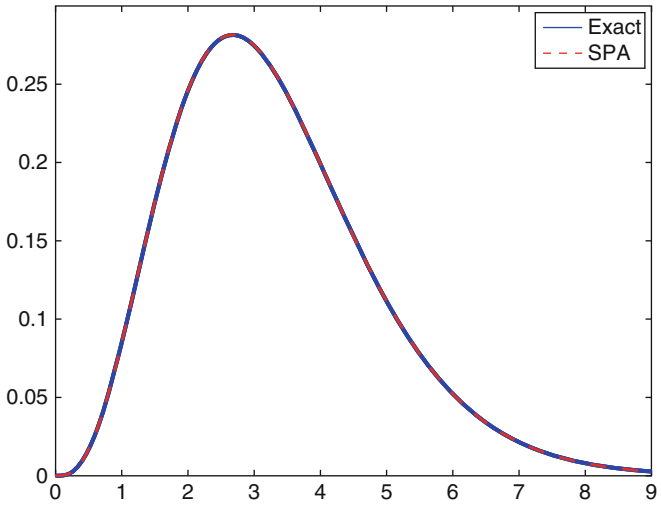


Fig. 32.2 True (*as calculated via inversion formula* (32.6)) and 2nd order SPA density of the sample variance S^2, for a sample of size 10 for $\mathbf{X} \sim \mathrm{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ with $\boldsymbol{\mu} = (-2,-1,0,1,2,2,1,0,-1,-2)'$ and $\boldsymbol{\Sigma}$ corresponding to an AR(1) process with $\rho = 0.5$. The two graphs are optically indistinguishable; the SPA is about 14 times faster to compute

$$Y = \sum_{i=1}^{n} \left(X_i - \bar{X}\right)^2 = \mathbf{X}'\mathbf{M}'\mathbf{M}\mathbf{X} = \mathbf{X}'\mathbf{M}\mathbf{X}$$

is a quadratic form. The eigenvalues $\{\lambda_i\}$ of $\boldsymbol{\Sigma}^{\frac{1}{2}}\mathbf{M}\boldsymbol{\Sigma}^{\frac{1}{2}}$ are nonnegative because $\mathbf{M}$ is positive-semidefinite and $\boldsymbol{\Sigma}$ is positive-definite. (To see this, first observe that, if $\boldsymbol{\Sigma}$ positive-definite, then $\boldsymbol{\Sigma}^{\frac{1}{2}}$ can be constructed, and is also positive-definite, so that, for any vector $\mathbf{w} \in \mathbb{R}^n \setminus \mathbf{0}$, $\mathbf{z} := \boldsymbol{\Sigma}^{\frac{1}{2}}\mathbf{w} \neq \mathbf{0}$. Next, $\mathbf{w}'\boldsymbol{\Sigma}^{\frac{1}{2}}\mathbf{M}\boldsymbol{\Sigma}^{\frac{1}{2}}\mathbf{w} = \mathbf{z}'\mathbf{M}\mathbf{z} \geq 0$ because $\mathbf{M}$ is positive-semidefinite. Of course, $\mathbf{z}'\mathbf{M}\mathbf{z} \geq 0$ also follows simply because $Y = (n-1)S^2 = \mathbf{X}'\mathbf{M}\mathbf{X}$ cannot be negative. Thus, the eigenvalues of $\boldsymbol{\Sigma}^{\frac{1}{2}}\mathbf{M}\Sigma^{\frac{1}{2}}$ are nonnegative.)

A scale transformation yields the density $f_{S^2}(s) = m f_Y(ms)$, $m = n-1$, while the cdf is given by $F_{S^2}(s) = \Pr(Y \leq ms)$. To illustrate, let $\boldsymbol{\mu} = (-2,-1,0,1,2,2,1,0,-1,-2)'$, so that $n = 10$, and $\boldsymbol{\Sigma}$ corresponding to a first-order autoregressive, or AR(1), process with parameter $\rho = 0.5$, for which the (i,j)th element of $\boldsymbol{\Sigma}$ is given by $\rho^{|i-j|}/(1-\rho^2)$. Figure 32.2 plots f_{S^2} and demonstrates the extreme accuracy of the SPA. ■

For symmetric matrices $\mathbf{A}$ and $\mathbf{B}$, a ratio of quadratic forms is given by

$$R = \frac{\mathbf{X}'\mathbf{A}\mathbf{X}}{\mathbf{X}'\mathbf{B}\mathbf{X}}, \quad \mathbf{X} \sim \mathrm{N}_n(\boldsymbol{\mu}, \boldsymbol{\Sigma}), \quad \mathbf{B} \neq \mathbf{0}, \quad \mathbf{B} \geq 0, \quad \boldsymbol{\Sigma} > 0, \tag{32.46}$$

and arises in many contexts in which quadratic forms appear. The restriction that $\mathbf{B}$ is positive-semidefinite but nonzero ensures that the denominator is positive with probability one. (Note that, if $\mathbf{B}$ has z zero eigenvalues, $0 < z < n$, then there exists a z–dimensional hyperplane $\mathcal{Z}$ in $\mathbb{R}^n$ (e.g., a line for $z = 1$, etc.) such that, for $\mathbf{X} \in \mathcal{Z}$, $\mathbf{X}'\mathbf{B}\mathbf{X} = 0$. However, $\mathcal{Z}$ has measure zero in $\mathbb{R}^n$ so that, with probability one, $\mathbf{X}'\mathbf{B}\mathbf{X} > 0$.) Let $\boldsymbol{\Sigma}^{\frac{1}{2}}$ be such that $\boldsymbol{\Sigma}^{\frac{1}{2}}\boldsymbol{\Sigma}^{\frac{1}{2}} = \boldsymbol{\Sigma}$. Then

$$R = \frac{\mathbf{X}'\mathbf{A}\mathbf{X}}{\mathbf{X}'\mathbf{B}\mathbf{X}} = \frac{\mathbf{Z}'\mathbf{A}^*\mathbf{Z}}{\mathbf{Z}'\mathbf{B}^*\mathbf{Z}},$$

where $\mathbf{A}^* = \boldsymbol{\Sigma}^{\frac{1}{2}}\mathbf{A}\boldsymbol{\Sigma}^{\frac{1}{2}}$, $\mathbf{B}^* = \boldsymbol{\Sigma}^{\frac{1}{2}}\mathbf{B}\boldsymbol{\Sigma}^{\frac{1}{2}}$, and $\mathbf{Z} = \boldsymbol{\Sigma}^{-1/2}\mathbf{X} \sim \mathrm{N}\left(\boldsymbol{\Sigma}^{-1/2}\boldsymbol{\mu}, \mathbf{I}\right)$, so that we may assume $\boldsymbol{\Sigma} = \mathbf{I}$ without loss of generality. Observe that, if $\mathbf{X} \sim \mathrm{N}\left(\mathbf{0}, \sigma^2\mathbf{I}\right)$, then σ^2 can be factored out of the numerator and denominator, so that R does not depend on σ^2.

Let $\mathbf{X} \sim \mathrm{N}_n(\boldsymbol{\mu}, \mathbf{I})$. For computing the cdf of ratio R in (32.46) at a given value r, construct the spectral decomposition

$$\mathbf{A} - r\mathbf{B} = \mathbf{P}\boldsymbol{\Lambda}\mathbf{P}', \tag{32.47}$$

$\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, and let $\mathbf{W} = \mathbf{P}'\mathbf{X} \sim \mathrm{N}(\boldsymbol{\nu}, \mathbf{I}_n)$, where $\boldsymbol{\nu} = \mathbf{P}'\boldsymbol{\mu} = (\nu_1, \ldots, \nu_n)'$. Then

$$\begin{aligned}\Pr(R \le r) &= \Pr\left(\mathbf{X}'\mathbf{A}\mathbf{X} \le r\,\mathbf{X}'\mathbf{B}\mathbf{X}\right) = \Pr\left(\mathbf{X}'(\mathbf{A} - r\mathbf{B})\mathbf{X} \le 0\right) \\ &= \Pr\left(\mathbf{X}'\mathbf{P}\boldsymbol{\Lambda}\mathbf{P}'\mathbf{X} \le 0\right) = \Pr\left(\mathbf{W}'\boldsymbol{\Lambda}\mathbf{W} \le 0\right) = F_S(0),\end{aligned} \tag{32.48}$$

where $S = \sum_{i=1}^n \lambda_i W_i^2$ and $W_i^2 \overset{\text{ind}}{\sim} \chi^2\left(1, \nu_i^2\right)$, so that S is a weighted sum of noncentral χ^2 random variables, each with one degree of freedom and noncentrality parameter ν_i^2, $i = 1, \ldots, n$. The λ_i are the eigenvalues of $\mathbf{A} - r\mathbf{B}$, some of which, depending on $\mathbf{A}$ and $\mathbf{B}$, might be zero.

If $\mathbf{B} > 0$, then both $\mathbf{B}^{1/2}$ and $\mathbf{B}^{-1/2}$ exist, so that R can be written as

$$R = \frac{\mathbf{X}'\mathbf{A}\mathbf{X}}{\mathbf{X}'\mathbf{B}\mathbf{X}} = \frac{\mathbf{X}'\mathbf{B}^{\frac{1}{2}}\mathbf{B}^{-\frac{1}{2}}\mathbf{A}\mathbf{B}^{-\frac{1}{2}}\mathbf{B}^{\frac{1}{2}}\mathbf{X}}{\mathbf{X}'\mathbf{B}^{\frac{1}{2}}\mathbf{B}^{\frac{1}{2}}\mathbf{X}} = \frac{\mathbf{Y}'\mathbf{C}\mathbf{Y}}{\mathbf{Y}'\mathbf{Y}},$$

where $\mathbf{Y} = \mathbf{B}^{1/2}\mathbf{X}$ and $\mathbf{C} = \mathbf{B}^{-1/2}\mathbf{A}\mathbf{B}^{-1/2}$. It is well known that, if $R = \mathbf{Y}'\mathbf{C}\mathbf{Y}/\mathbf{Y}'\mathbf{Y}$, then

$$c_{\min} \le R \le c_{\max}, \tag{32.49}$$

where $c_{\min}$ and $c_{\max}$ refer respectively to the smallest and largest eigenvalues of $\mathbf{C}$. (Note that they are real because $\mathbf{C}$ is symmetric.)

From (32.48), $F_R(r) = F_S(0)$, and, because the cf and mgf of S are tractable, the cdf inversion formula or the saddlepoint approximation can be applied to compute

$F_R(r)$. Note that, for each value of r, (32.47) needs to be computed to obtain the v_i and λ_i. For large n, this will be the most time consuming part of the computation.

The characteristic function of R is not tractable in general, so that direct application of the inversion formula is not possible. We show how an exact calculation can be performed, and use of the saddlepoint approximation.

For the exact solution, let N and D be continuous random variables with joint cf $\varphi_{N,D}$ and such that $\Pr(D > 0) = 1$ and $\mathbb{E}[D] < \infty$. The crucial result is from Geary (1944), who showed that the density of $R = N/D$ can be written as

$$f_R(r) = \frac{1}{2\pi i} \int_{-\infty}^{\infty} \left[\frac{\partial \varphi_{N,D}(s,t)}{\partial t} \right]_{t=-rs} \mathrm{d}s. \tag{32.50}$$

With $N = \mathbf{X}'\mathbf{A}\mathbf{X}$ and $D = \mathbf{X}'\mathbf{B}\mathbf{X}$, calculation shows that

$$\mathbb{M}_{N,D}(s,t) = |\mathbf{\Omega}|^{-\frac{1}{2}} \exp\left\{ -\tfrac{1}{2} \boldsymbol{\mu}' \left(\mathbf{I} - \mathbf{\Omega}^{-1} \right) \boldsymbol{\mu} \right\}, \quad \mathbf{\Omega} = \mathbf{I} - 2(s\mathbf{A} + t\mathbf{B}),$$

and $\varphi_{N,D}(s,t) = \mathbb{M}_{N,D}(is, it)$. Then, we can rewrite (32.50) as

$$f_R(r) = \frac{1}{\pi} \int_0^{\infty} \mathrm{Re}\left[\mathbb{M}^*(is) \right] \mathrm{d}s, \tag{32.51}$$

where $\mathbb{M}^*(s) := [\partial \mathbb{M}_{N,D}(s,t)/\partial t]_{t=-rs}$ is given in Butler and Paolella (2008) as

$$\mathbb{M}^*(s) = \left[\prod_{i=1}^{n} (1 - 2s\lambda_i)^{-1/2} \right] \exp\left\{ s \sum_{i=1}^{n} \frac{\lambda_i v_i^2}{1 - 2s\lambda_i} \right\} \left[\mathrm{tr}\, \mathbf{D}^{-1}\mathbf{H} + \boldsymbol{\nu}'\mathbf{D}^{-1}\mathbf{H}\mathbf{D}^{-1}\boldsymbol{\nu} \right],$$

with spectral decomposition $\mathbf{A} - r\mathbf{B} = \mathbf{P}\mathbf{\Lambda}\mathbf{P}'$ as in (32.47), $\mathbf{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$, $\mathbf{P} = [\mathbf{p}_1, \ldots, \mathbf{p}_n]$, $\mathbf{D} = \mathbf{I} - 2s\mathbf{\Lambda}$, $\mathbf{H} = \mathbf{P}'\mathbf{B}\mathbf{P}$, $\boldsymbol{\nu} = \mathbf{P}'\boldsymbol{\mu}$, and we have exploited the fact that $\mathrm{Re}[\mathbb{M}^*(is)]$ is an even function of s.

For use with software packages not supporting complex arithmetic, the following result can be proven; see Broda and Paolella (2009b). Let $R = \mathbf{X}'\mathbf{A}\mathbf{X}/\mathbf{X}'\mathbf{B}\mathbf{X}$, where $\mathbf{X} \sim \mathrm{N}_n(\boldsymbol{\mu}, \mathbf{I})$, $\mathbf{B} \neq 0, \mathbf{B} \geq 0$. Then the density of R is

$$f_R(r) = \frac{1}{\pi} \int_0^{\infty} \frac{\rho(u)\cos\beta(u) - u\delta(u)\sin\beta(u)}{2\gamma(u)}\, \mathrm{d}u, \tag{32.52}$$

where

$$\beta(u) = \frac{1}{2}\sum_{i=1}^{n} \arctan a_i + \frac{\theta_i a_i}{c_i}, \qquad \gamma(u) = \exp\left\{ \frac{1}{2}\sum_{i=1}^{n} \frac{\theta_i b_i}{c_i} + \frac{1}{4}\ln c_i \right\},$$

$$\rho(u) = \mathrm{tr}\,\mathbf{H}\mathbf{F}^{-1} + \boldsymbol{\nu}'\mathbf{F}^{-1}(\mathbf{H} - u^2\mathbf{\Lambda}\mathbf{H}\mathbf{\Lambda})\mathbf{F}^{-1}\boldsymbol{\nu}, \quad \delta(u) = \mathrm{tr}\,\mathbf{H}\mathbf{\Lambda}\mathbf{F}^{-1} + 2\boldsymbol{\nu}'\mathbf{F}^{-1}\mathbf{H}\mathbf{\Lambda}\mathbf{F}^{-1}\boldsymbol{\nu},$$

$a_i = \lambda_i u$, $b_i = a_i^2$, $c_i = 1 + b_i$, $\theta_i = v_i^2 = (\mathbf{p}_i'\boldsymbol{\mu})^2$, and $\mathbf{F} = \mathbf{I} + u^2\mathbf{\Lambda}^2$.

For the SPA, as before, let $R = N/D$ with $N = \mathbf{X}'\mathbf{A}\mathbf{X}$ and $D = \mathbf{X}'\mathbf{B}\mathbf{X}$. The same derivation that leads to (32.50) also shows that, with C the (constructed) random variable associated with mgf

$$\mathbb{M}_C(s) = \frac{1}{\mathbb{E}[D]} \frac{\partial}{\partial t} \mathbb{M}_{N,D}(s,t)\big|_{t=-rs}, \tag{32.53}$$

the density of R is

$$f_R(r) = \mathbb{E}[D] f_C(0), \tag{32.54}$$

where f_C is the density of C. We approximate f_C with $\hat{f}_C$, the SPA applied to $\mathbb{M}_C$, so that the SPA of the density of R is, from (32.54), $\hat{f}_R(r) = \mathbb{E}[D]\hat{f}_C(0)$, as was done in Daniels (1954, Sect. 9). This leads to (see Butler and Paolella 2008 for details)

$$\hat{f}_R(r) = \frac{J(\hat{s})}{\sqrt{2\pi K_S''(\hat{s})}} \exp\{\mathbb{K}_S(\hat{s})\}, \tag{32.55}$$

where $\mathbb{K}_S$ is the c.g.f. of S in (32.48) and $\hat{s}$ solves $\mathbb{K}_S'(\hat{s}) = 0$. Quantity $J(\hat{s})$ is computed from

$$J(s) = \operatorname{tr}(\mathbf{U}\mathbf{H}) + \boldsymbol{\nu}'\mathbf{U}\mathbf{H}\mathbf{U}\boldsymbol{\nu}, \tag{32.56}$$

with $\mathbf{U} = (\mathbf{I} - 2s\boldsymbol{\Lambda})^{-1}$, and $\mathbf{H}$, $\mathbf{P}$, $\boldsymbol{\Lambda}$ and $\boldsymbol{\nu}$ are given above.

A second order saddlepoint density approximation for the general case can be derived. In particular, from Butler (2007, p. 383),

$$\tilde{f}_R(r) = \hat{f}_R(r)(1 + O), \tag{32.57}$$

where $\hat{f}_R(r)$ is given in (32.55),

$$O = \left(\frac{\hat{\kappa}_4}{8} - \frac{5}{24}\hat{\kappa}_3^2\right) + \frac{J_r'(\hat{s})\hat{\kappa}_3}{2J_r(\hat{s})\sqrt{K_S''(\hat{s})}} - \frac{J_r''(\hat{s})}{2J_r(\hat{s})K_S''(\hat{s})}, \tag{32.58}$$

$\hat{\kappa}_i = K_S^{(i)}(\hat{s})/K_S''(\hat{s})^{i/2}$, and $J_r'(\hat{s}) = 2\operatorname{tr}(\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}\mathbf{H}) + 4\boldsymbol{\nu}'\mathbf{U}\boldsymbol{\Lambda}\mathbf{U}\mathbf{H}\mathbf{U}\boldsymbol{\nu}$. The second derivative of J_r, as required in (32.58), could also be algebraically formulated, but it is easily and accurately numerically obtained.

An important special case of the general ratio R is when $\boldsymbol{\mu} = \mathbf{0}$. Then $\boldsymbol{\nu} = \mathbf{0}$ and (32.55) easily reduces to

$$\hat{f}_R(r) = \frac{\operatorname{tr}(\mathbf{I} - 2\hat{s}\boldsymbol{\Lambda})^{-1}\mathbf{H}}{\sqrt{4\pi\sum_{i=1}^n \lambda_i^2(1 - 2\lambda_i\hat{s})^{-2}}} \prod_{i=1}^n (1 - 2\lambda_i\hat{s})^{-1/2}, \tag{32.59}$$

which was first derived by Lieberman (1994a,b).

If $\boldsymbol{\mu} = \mathbf{0}$, $\boldsymbol{\Sigma} = \sigma^2\mathbf{I}$, and $\mathbf{B} = \mathbf{I}$, then matters simplify considerably. Firstly, $\operatorname{tr}(\mathbf{I} - 2\hat{s}\boldsymbol{\Lambda})^{-1}\mathbf{H} = n$, seen by noting that $\mathbf{H} = \mathbf{P}'\mathbf{P} = \mathbf{I}$, implying

$$\operatorname{tr}\left(\mathbf{I}_n - 2\hat{s}\mathbf{\Lambda}\right)^{-1} = \sum_{i=1}^{n} \left(1 - 2\hat{s}\lambda_i\right)^{-1}.$$

Next, as the saddlepoint equation solves $0 = \mathbb{K}'_S(\hat{s}) = \sum_{i=1}^{n} \lambda_i \left(1 - 2\hat{s}\lambda_i\right)^{-1}$, it follows that $\sum_{i=1}^{n} \left(1 - 2\hat{s}\lambda_i\right)^{-1} = n$, because

$$n = \sum_{i=1}^{n} \frac{1 - 2\hat{s}\lambda_i}{1 - 2\hat{s}\lambda_i} = \sum_{i=1}^{n} \frac{1}{1 - 2\hat{s}\lambda_i} - 2\hat{s} \sum_{i=1}^{n} \frac{\lambda_i}{1 - 2\hat{s}\lambda_i}.$$

Thus, from (32.59), $\hat{f}_R(r)$ can be expressed as

$$\hat{f}_R(r) = \frac{n \prod_{i=1}^{n} \left(1 - 2\lambda_i \hat{s}\right)^{-1/2}}{\sqrt{4\pi \sum_{i=1}^{n} \lambda_i^2 \left(1 - 2\lambda_i \hat{s}\right)^{-2}}}. \tag{32.60}$$

It is easy to confirm that the eigenvalues of $(\mathbf{A} - r\mathbf{I})$ are given by

$$\lambda_i = \zeta_i - r, \qquad \boldsymbol{\zeta} = \operatorname{Eig}(\mathbf{A}). \tag{32.61}$$

These need only be calculated once, so that density (32.60) is easily computed as a function of r. This seemingly very special case often arises in various applications, typically as the distribution of a statistic under the null hypothesis. As such, we consider it further in the next example.

Example 5. (Ratio in the null case) In (32.46), let $\mathbf{X} \sim \mathrm{N}\left(\mathbf{0}, \sigma^2\mathbf{I}\right)$ and $\mathbf{B} = \mathbf{I}$. First note that σ^2 can be set to one, without loss of generality, as it can be factored out of $\mathbf{X}$ and it cancels from the numerator and denominator. Let the spectral decomposition of $\mathbf{A}$ be given by $\mathbf{A} = \mathbf{P}\mathbf{\Lambda}\mathbf{P}'$, with $\mathbf{\Lambda} = \operatorname{diag}(\lambda_1, \ldots, \lambda_n)$ the eigenvalues of $\mathbf{A}$. Then

$$R = \frac{\mathbf{X}'\mathbf{A}\mathbf{X}}{\mathbf{X}'\mathbf{X}} = \frac{\mathbf{X}'\mathbf{P}\mathbf{\Lambda}\mathbf{P}'\mathbf{X}}{\mathbf{X}'\mathbf{P}\mathbf{P}'\mathbf{X}} = \frac{\mathbf{Y}'\mathbf{\Lambda}\mathbf{Y}}{\mathbf{Y}'\mathbf{Y}}, \tag{32.62}$$

where $\mathbf{Y} = \mathbf{P}'\mathbf{X} \sim \mathrm{N}(\mathbf{0}, \mathbf{I})$. Thus, R can be expressed as

$$R = \frac{\sum_{i=1}^{n} \lambda_i \chi_i^2}{\sum_{i=1}^{n} \chi_i^2} =: \frac{U}{V}, \tag{32.63}$$

where U and V are defined to be the numerator and denominator, respectively, and the χ_i^2 are iid central chi-square with one degree of freedom. From (32.49), $\lambda_{\min} \leq R \leq \lambda_{\max}$, where $\lambda_{\min}$ and $\lambda_{\max}$ refer respectively to the smallest and largest eigenvalues of $\mathbf{A}$. Thus, R has finite support, and all positive moments must exist.

Calculation (see Paolella 2007, Example 2.22) shows that $\mathbb{E}[R] = n^{-1} \sum_{i=1}^{n} \lambda_i =: \bar{\lambda}$; and recalling the mean of χ_i^2, we see that $\mathbb{E}[R] = \mathbb{E}[U]/\mathbb{E}[V]$, or $\mathbb{E}[U] = \mathbb{E}[R]\mathbb{E}[V]$. This result would also be the case if R were independent

of V, because then, $\mathbb{E}[RV]$ would equal $\mathbb{E}[R]\mathbb{E}[V]$, so that $U = RV$ would imply $\mathbb{E}[U] = \mathbb{E}[R]\mathbb{E}[V]$. This motivates the possibility that R is actually independent of V, which turns out to be true, as shown by Pitman in 1937; see Stuart and Ord (1994, p. 529). The consequence of this is that $U = RV$ implies $\mathbb{E}[U] = \mathbb{E}[R]\mathbb{E}[V]$, and, more generally, $U^p = (RV)^p$ implies $\mathbb{E}[U^p] = \mathbb{E}[R^p]\mathbb{E}[V^p]$ for all p such that the expectations exist, i.e.,

$$\mathbb{E}\left[R^p\right] = \frac{\mathbb{E}\left[U^p\right]}{\mathbb{E}\left[V^p\right]}. \tag{32.64}$$

It is this latter fact which is critical for the ease with which the moments of R can be evaluated: calculating the raw moments of R has been reduced to deriving the raw moments of both U and V, which is straightforward.

There is another interesting consequence of this independence result. Again with $R = U/V$ for $U = \mathbf{X}'\mathbf{A}\mathbf{X}$, $V = \mathbf{X}'\mathbf{X}$, $\mathbf{X} \sim \mathrm{N}\left(\mathbf{0}, \sigma^2\mathbf{I}\right)$, and $\lambda = \mathrm{Eig}(\mathbf{A})$, the independence of R and V implies, for r such that $\min \lambda_i < r < \max \lambda_i$,

$$F_R(r) = \Pr\left(R \le r\right) = \Pr\left(R \le r \mid V = 1\right) = \Pr\left(U \le r \mid V = 1\right).$$

The joint mgf of U and V is

$$\begin{aligned}
\mathbb{M}_{U,V}\left(s,t\right) &= \mathbb{E}\left[\exp\{sU + tV\}\right] \\
&= \mathbb{E}\left[\exp\left\{s\sum_{i=1}^{n}\lambda_i\chi_i^2 + t\sum_{i=1}^{n}\chi_i^2\right\}\right] = \mathbb{E}\left[\exp\left\{\sum_{i=1}^{n}\left(s\lambda_i + t\right)\chi_i^2\right\}\right] \\
&= \prod_{i=1}^{n}\left[1 - 2\left(s\lambda_i + t\right)\right]^{-1/2}.
\end{aligned} \tag{32.65}$$

Based on this, the conditional saddlepoint approximation discussed in Sect. 32.4.1 is applicable, and it would be of interest to compare the accuracy of the cdf approximation from its use with the one discussed above. However, it turns out that they are identical, as proven in Butler and Paolella (1998) in a more general setting. ■

32.6 A Finance Application

The number of papers using saddlepoint methods in finance has increased enormously in the last decade, covering, among others, applications in options pricing, credit risk and Collateralized Debt Obligations (CDOs), portfolio allocation and risk management. A non-exhaustive collection of papers includes Rogers and Zane (1999), Martin et al. (2001), Duffie and Pan (2001), Gordy (2002), Collin-Dufresne and Goldstein (2002), Dembo et al. (2004), Glasserman (2004), Xiong et al.

(2005), Yang et al. (2006), Aït-Sahalia and Yu (2006), Veilex (2007), Wong (2008), Glasserman and Kim (2009) and Broda and Paolella (2009a).

In this section, we will consider an application to risk management. In particular, the task is to quantify the risk associated with a portfolio of financial assets. The first issue that needs to be addressed concerns the choice of risk measure. The most common choice, and which is officially endorsed by the Basle committee on banking supervision, is the so-called Value-at-Risk, or VaR. Denote the portfolio loss as L, and suppose that L is continuous. Then, for a given confidence level $(1-\alpha)$, the $100(1-\alpha)\%$ VaR is defined as the $(1-\alpha)\%$ quantile of the distribution of L, i.e.,

$$F_L\left(\text{VaR}_L^{(1-\alpha)}\right) = (1-\alpha). \tag{32.66}$$

A common choice for α is 0.01. Intuitively, the 99% VaR is the maximum loss that will not be exceeded in 99% of cases.

Recently, a substantial body of literature has emerged which criticizes the use of VaR as a risk measure. The criticism is mostly based on the fact that VaR is not subadditive, i.e., the VaR of a portfolio of assets is not guaranteed to be smaller than or equal to the sum of the VaRs of the individual assets. This is counterintuitive because it may discourage diversification. An alternative risk measure which does not suffer from this drawback is the expected shortfall, or ES. For continuous L and a given confidence level $(1-\alpha)$, it is defined as

$$\text{ES}_L^{(1-\alpha)} := \mathbb{E}\left[L \mid L > \text{VaR}_L^{(1-\alpha)}\right],$$

i.e., the expected loss, conditional on the loss exceeding the VaR. Note that the expected shortfall is related to the partial expectation via

$$\text{ES}_L^{(1-\alpha)} = \frac{\mathbb{E}[L]}{\alpha} - \frac{1}{\alpha} G_L\left(\text{VaR}_L^{(1-\alpha)}\right). \tag{32.67}$$

More details on the ES can be found in Broda and Paolella (2011).

As is clear from the definitions of these risk measures, their evaluation requires a means of computing the distribution and partial expectation of L. The caveat here is that most real portfolios contain securities which depend nonlinearly on the risk factors (e.g., stock options and other derivatives). Replicating these nonlinearities exactly would require a simulation approach for estimating the loss distribution, because the exact distribution is intractable. However, reliable estimates far into the tails require large numbers of replications, which is impractical, especially if the securities themselves cannot be valued analytically. A trade-off often used in practice is the so-called Delta-Gamma approximation. Suppose the portfolio value depends on n risk factors $S = (S_1, \ldots, S_n)'$. Typical risk factors are stock prices, commodity prices, or exchange rates. Defining the sensitivities $\delta_j = \partial X/\partial S_j$, $\Gamma_{jk} = \partial^2 X/\partial S_j \partial S_k$, the portfolio loss can be locally approximated by the quadratic

$$Q := d + \mathbf{a}'\Delta S + \Delta S'\mathbf{A}\Delta S,$$

where $\Delta S := S_T - S_0$ are the changes in the risk factors, $\mathbf{a} = -\boldsymbol{\delta}$, and $\mathbf{A} = -\frac{1}{2}\boldsymbol{\Gamma}$.

Now assume that $\Delta S \sim \mathrm{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, and note that for $d = 0$ and $\mathbf{a} = \mathbf{0}$, Q is just a quadratic form in normal variables. Generalizing (32.44), the mgf of Q in the general case is derived in the appendix. It is given by

$$\mathbb{M}_Q(s) = \exp\left\{ s\left(d + \boldsymbol{\mu}'\mathbf{A}\boldsymbol{\mu} + \mathbf{a}'\boldsymbol{\mu}\right) + s^2 \sum_{i=1}^{n} \frac{c_i^2}{1 - 2s\lambda_i} \right\} \prod_{i=1}^{n} (1 - 2s\lambda_i)^{-1/2}, \tag{32.68}$$

where $\boldsymbol{\Sigma}^{1/2}\mathbf{A}\boldsymbol{\Sigma}^{1/2} = \mathbf{P}\boldsymbol{\Lambda}\mathbf{P}'$ with $\mathbf{P}$ orthogonal, $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ and

$$(c_1, \ldots, c_n)' = \mathbf{P}'\left(\boldsymbol{\Sigma}^{1/2}\mathbf{a}/2 + \boldsymbol{\Sigma}^{1/2}\mathbf{A}\boldsymbol{\mu}\right).$$

With the mgf of Q available, the cdf (and hence the VaR) can be computed by means of (32.8). This has first been pursued by Rouvinez (1997) and subsequently generalized to the case of multivariate t risk factors by Glasserman et al. (2002). Similarly, partial expectations (and thus the expected shortfall) can be computed from (32.18), as in Yueh and Wong (2010). Broda (forthcoming) generalized this to the case of multivariate t risk factors, in order to account for the well-documented heavy tails of asset returns.

Now consider the saddlepoint approximation. The cgf of Q is given by

$$\mathbb{K}_Q(s) = sm + s^2 \sum_{i=1}^{n} c_i^2 \vartheta_i + \frac{1}{2} \sum_{i=1}^{n} \ln \vartheta_i,$$

where $m = d + \boldsymbol{\mu}'\mathbf{A}\boldsymbol{\mu} + \mathbf{a}'\boldsymbol{\mu}$ and $\vartheta_i = \vartheta_i(s) = (1 - 2s\lambda_i)^{-1}$. The derivatives are easily determined to be

$$\begin{aligned} \mathbb{K}_Q'(s) &= m + \sum_{i=1}^{n} s c_i^2 \vartheta_i + s^2 c_i^2 \vartheta_i^2 \lambda_i + \vartheta_i \lambda_i, \quad \mathbb{K}_Q''(s) \\ &= \sum_{i=1}^{n} c_i^2 \vartheta_i + 4s c_i^2 \vartheta_i^2 \lambda_i + 4s^2 c_i^2 \vartheta_i^3 \lambda_i^2 + 2\vartheta_i^2 \lambda_i^2, \end{aligned}$$

$$\mathbb{K}_Q'''(s) = \sum_{i=1}^{n} 6c_i^2 \vartheta_i^2 \lambda_i + 24s c_i^2 \vartheta_i^3 \lambda_i^2 + 24s^2 c_i^2 \vartheta_i^4 \lambda_i^3 + 8\vartheta_i^3 \lambda_i^3,$$

and

$$\mathbb{K}_Q''''(s) = \sum_{i=1}^{n} 48c_i^2 \vartheta_i^3 \lambda_i^2 + 192s c_i^2 \vartheta_i^4 \lambda_i^3 + 192s^2 c_i^2 \vartheta_i^5 \lambda_i^4 + 48\vartheta_i^4 \lambda_i^4.$$

With these, the saddlepoint approximation to the cdf is straightforwardly computed from (32.9), as in Feuerverger and Wong (2000), or from (32.15) for a second order approximation. This has been extended to the multivariate t setting in Broda (forthcoming) as well.

For illustration, we consider a portfolio with a stock as single risk factor. Because it is an asymptotic expansion, the accuracy of the SPA improves as the number of risk factors grows, so that the case of a single risk factor can serve as a worst-case scenario. To be specific, we consider

$$Q = \Delta S + \Delta S^2, \quad \Delta S \sim \mathrm{N}(0,1),$$

which corresponds to being short a number of slightly in-the-money European Calls on the same stock, close to expiry. The reason for considering a short position is that the loss is bounded for long positions.

The left panel of Fig. 32.3 illustrates the SPA to the cdf of Q. The first order (dashes) and second order (circles) SPAs are graphically almost indistinguishable from the true cdf (solid). A clearer picture emerges from the bottom left panel,

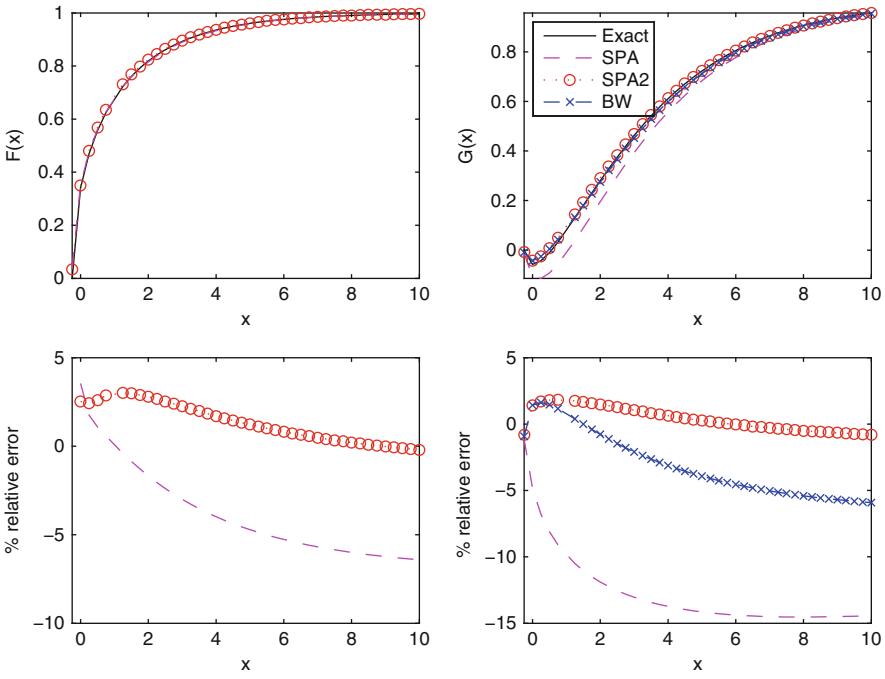


Fig. 32.3 Approximations to the cdf (*top left*) and partial expectation (*top right*) of a quadratic approximation to a short position in European Calls. The bottom row shows the relative error, defined as $100(\hat{F} - F)/\min(F, 1-F)$ and $100(\hat{G} - G)/(\mathbb{E}[Q] - G)$, respectively. BW refers to approximation (32.20)

which shows the relative error, computed as $(\hat{F}_Q - F_Q)/\min(F_Q, 1 - F_Q)$. The second order approximation shows less than 5% error across the entire support and thus has a clear advantage over its first order counterpart, especially in the tail. The right panel of the same figure shows the corresponding results for the partial expectation G_Q. In addition to the first and second order SPAs, the crosses represent approximation (32.20), which ranges between the other two in terms of accuracy. Note also that the overall accuracy of the approximations is lower than for the cdf, indicating that the partial expectation is a more difficult target for approximation.

Next, consider Fig. 32.4, which illustrates the approximations to the VaR (left) and ES (right). The VaR is computed by replacing F_Q with its first or second order SPA in (32.66), while the ES is obtained from (32.67), after plugging in the corresponding approximate VaR. For the ES, when using (32.20) to approximate the partial expectation, we plug in the first-order VaR approximation. Interestingly, the first order approximation to the VaR outperforms the second order SPA except in the far tails. For the ES approximation however, the results are reversed, with the second order approximation dominating, at under 2% relative error. Somewhat surprisingly, the approximation based on (32.20) performs very similarly to the second order

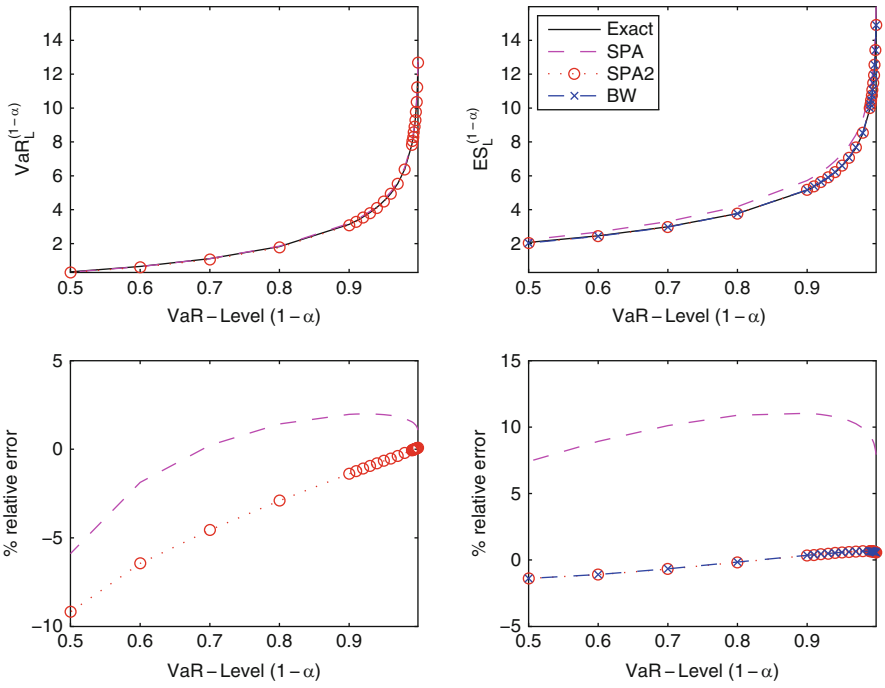


Fig. 32.4 Approximations to the VaR (*top left*) and ES (*top right*) of a quadratic approximation to a short position in European Calls. The bottom row shows the relative error, defined as $100(\hat{\text{VaR}} - \text{VaR})/\text{VaR}$ and $100(\hat{\text{ES}} - \text{ES})/\text{ES}$, respectively

SPA, despite the fact that both quantities which enter the approximation (the VaR and the partial expectation) are only approximated to first order. This would seem to suggest that a form of error cancelation is taking place.

A.1 Derivation of the mgf of Q

The mgf of $Q = \mathbf{X}'\mathbf{A}\mathbf{X} + \mathbf{a}'\mathbf{X} + d$, where $\mathbf{X} \sim \mathrm{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$, is

$$\mathbb{E}\left[e^{sQ}\right] = \frac{1}{(2\pi)^{n/2}|\boldsymbol{\Sigma}|^{1/2}} \int_{\mathbb{R}^n} \exp\Big\{ s\mathbf{x}'\mathbf{A}\mathbf{x} + s\mathbf{a}'\mathbf{x} + sd - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\Big\}\, d\mathbf{x},$$

and the exponent can be rearranged as

$$\begin{aligned} & s\mathbf{x}'\mathbf{A}\mathbf{x} + s\mathbf{a}'\mathbf{x} + sd - \frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})'\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu}) \\ &= -\frac{1}{2}\left(\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - 2sd\right) + \frac{1}{2}(\boldsymbol{\mu} + s\boldsymbol{\Sigma}\mathbf{a})'\left(\mathbf{I} - 2s\mathbf{A}\boldsymbol{\Sigma}\right)^{-1}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} + s\boldsymbol{\Sigma}\mathbf{a}) \\ &- \frac{1}{2}(\mathbf{x}-\mathbf{m})'(\boldsymbol{\Sigma}^{-1} - 2s\mathbf{A})(\mathbf{x}-\mathbf{m}), \end{aligned}$$

where $\mathbf{m} = (\boldsymbol{\Sigma}^{-1} - 2s\mathbf{A})^{-1}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} + s\boldsymbol{\Sigma}\mathbf{a})$. As $\boldsymbol{\Sigma} > 0$ and $\mathbf{A}$ is finite, there exists a neighborhood N_0 around zero such that, for $s \in N_0$, $\boldsymbol{\Sigma}^{-1} - 2s\mathbf{A} > 0$. Recognizing the kernel of the multivariate normal distribution,

$$\int_{\mathbb{R}^n} \exp\left[-\frac{1}{2}(\mathbf{x}-\mathbf{m})'(\boldsymbol{\Sigma}^{-1} - 2s\mathbf{A})(\mathbf{x}-\mathbf{m})\right] d\mathbf{x} = (2\pi)^{n/2}\left|(\boldsymbol{\Sigma}^{-1} - 2s\mathbf{A})\right|^{-1/2},$$

the integral becomes

$$\mathbb{M}_Q(s) = |\mathbf{I} - 2s\mathbf{A}\boldsymbol{\Sigma}|^{-1/2} \times \exp\{E\}, \tag{A.1}$$

where

$$E := -\frac{1}{2}\left(\boldsymbol{\mu}'\boldsymbol{\Sigma}^{-1}\boldsymbol{\mu} - 2sd\right) + \frac{1}{2}(\boldsymbol{\mu} + s\boldsymbol{\Sigma}\mathbf{a})'\left(\mathbf{I} - 2s\mathbf{A}\boldsymbol{\Sigma}\right)^{-1}\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} + s\boldsymbol{\Sigma}\mathbf{a}). \tag{A.2}$$

Now let $\boldsymbol{\Sigma}^{1/2}$ be the symmetric square root of $\boldsymbol{\Sigma}$ and set $\boldsymbol{\Sigma}^{1/2}\mathbf{A}\boldsymbol{\Sigma}^{1/2} = \mathbf{P}\boldsymbol{\Lambda}\mathbf{P}'$ with $\mathbf{P}$ orthogonal, and $\boldsymbol{\Lambda} = \mathrm{diag}(\lambda_1, \ldots, \lambda_n)$ the eigenvalues of $\boldsymbol{\Sigma}^{1/2}\mathbf{A}\boldsymbol{\Sigma}^{1/2}$, the nonzero ones of which are the same as those of $\mathbf{A}\boldsymbol{\Sigma}$. Then, with $|\mathbf{P}'\mathbf{P}| = |\mathbf{I}| = 1$ and recalling that the determinant of a product is the product of the determinants,

$$\begin{aligned}
\left|\mathbf{I}-2s\mathbf{A}\boldsymbol{\Sigma}\right| &= \left|\boldsymbol{\Sigma}^{-1/2}\boldsymbol{\Sigma}^{1/2}\right|\left|\mathbf{I}-2s\mathbf{A}\boldsymbol{\Sigma}\right| = \left|\boldsymbol{\Sigma}^{-1/2}\right|\left|\boldsymbol{\Sigma}^{1/2}\right|\left|\mathbf{I}-2s\mathbf{A}\boldsymbol{\Sigma}\right| \\
&= \left|\boldsymbol{\Sigma}^{1/2}\right|\left|\mathbf{I}-2s\mathbf{A}\boldsymbol{\Sigma}\right|\left|\boldsymbol{\Sigma}^{-1/2}\right| = \left|\boldsymbol{\Sigma}^{1/2}\boldsymbol{\Sigma}^{-1/2}-2s\boldsymbol{\Sigma}^{1/2}\mathbf{A}\boldsymbol{\Sigma}\,\boldsymbol{\Sigma}^{-1/2}\right| \\
&= \left|\mathbf{I}-2s\boldsymbol{\Sigma}^{1/2}\mathbf{A}\boldsymbol{\Sigma}^{1/2}\right| = \left|\mathbf{I}-2s\mathbf{P}\boldsymbol{\Lambda}\mathbf{P}'\right| = \left|\mathbf{P}\mathbf{P}'-2s\mathbf{P}\boldsymbol{\Lambda}\mathbf{P}'\right| \\
&= \left|\mathbf{P}\right|\left|\mathbf{I}-2s\boldsymbol{\Lambda}\right|\left|\mathbf{P}'\right| = \left|\mathbf{P}'\right|\left|\mathbf{P}\right|\left|\mathbf{I}-2s\boldsymbol{\Lambda}\right| = \left|\mathbf{P}'\mathbf{P}\right|\left|\mathbf{I}-2s\boldsymbol{\Lambda}\right| \\
&= \left|\mathbf{I}-2s\boldsymbol{\Lambda}\right| = \prod_{i=1}^{n}(1-2s\lambda_i),
\end{aligned}$$

so that

$$\left|\mathbf{I}-2s\mathbf{A}\boldsymbol{\Sigma}\right|^{-1/2} = \prod_{i=1}^{n}(1-2s\lambda_i)^{-1/2}. \tag{A.3}$$

Next, E in (A.2) will be simplified. First recall that $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$, so that

$$\begin{aligned}
(\mathbf{I}-2s\mathbf{A}\boldsymbol{\Sigma})^{-1}\,\boldsymbol{\Sigma}^{-1} &= [\boldsymbol{\Sigma}\,(\mathbf{I}-2s\mathbf{A}\boldsymbol{\Sigma})]^{-1} = (\boldsymbol{\Sigma}-2s\boldsymbol{\Sigma}\mathbf{A}\boldsymbol{\Sigma})^{-1} \\
&= \left[\boldsymbol{\Sigma}^{1/2}\left(\mathbf{I}-2s\boldsymbol{\Sigma}^{1/2}\mathbf{A}\boldsymbol{\Sigma}^{1/2}\right)\boldsymbol{\Sigma}^{1/2}\right]^{-1} \\
&= \boldsymbol{\Sigma}^{-1/2}\left(\mathbf{I}-2s\boldsymbol{\Sigma}^{1/2}\mathbf{A}\boldsymbol{\Sigma}^{1/2}\right)^{-1}\boldsymbol{\Sigma}^{-1/2}.
\end{aligned}$$

Then

$$\begin{aligned}
E &= -\tfrac{1}{2}\left[(\boldsymbol{\Sigma}^{-1/2}\boldsymbol{\mu})'(\boldsymbol{\mu}\boldsymbol{\Sigma}^{-1/2}) - 2sd\right] \\
&\quad + \tfrac{1}{2}(\boldsymbol{\Sigma}^{-1/2}\boldsymbol{\mu}+s\boldsymbol{\Sigma}^{1/2}\mathbf{a})'(\mathbf{I}-2s\boldsymbol{\Sigma}^{1/2}\mathbf{A}\boldsymbol{\Sigma}^{1/2})^{-1}(\boldsymbol{\Sigma}^{-1/2}\boldsymbol{\mu}+s\boldsymbol{\Sigma}^{1/2}\mathbf{a}) \\
&= s(d+\boldsymbol{\mu}'\mathbf{A}\boldsymbol{\mu}+\mathbf{a}'\boldsymbol{\mu}) \\
&\quad + \tfrac{s^2}{2}(\boldsymbol{\Sigma}^{1/2}\mathbf{a}+2\boldsymbol{\Sigma}^{1/2}\mathbf{A}\boldsymbol{\mu})'(\mathbf{I}-2s\boldsymbol{\Sigma}^{1/2}\mathbf{A}\boldsymbol{\Sigma}^{1/2})^{-1}(\boldsymbol{\Sigma}^{1/2}\mathbf{a}+2\boldsymbol{\Sigma}^{1/2}\mathbf{A}\boldsymbol{\mu})
\end{aligned}$$

or

$$\begin{aligned}
E &= s(d+\boldsymbol{\mu}'\mathbf{A}\boldsymbol{\mu}+\mathbf{a}'\boldsymbol{\mu}) \\
&\quad + \tfrac{s^2}{2}(\boldsymbol{\Sigma}^{1/2}\mathbf{a}+2\boldsymbol{\Sigma}^{1/2}\mathbf{A}\boldsymbol{\mu})'\mathbf{P}\mathbf{P}'(\mathbf{I}-2s\boldsymbol{\Sigma}^{1/2}\mathbf{A}\boldsymbol{\Sigma}^{1/2})^{-1}\mathbf{P}\mathbf{P}'(\boldsymbol{\Sigma}^{1/2}\mathbf{a}+2\boldsymbol{\Sigma}^{1/2}\mathbf{A}\boldsymbol{\mu}) \\
&= s(d+\boldsymbol{\mu}'\mathbf{A}\boldsymbol{\mu}+\mathbf{a}'\boldsymbol{\mu}) \\
&\quad + \tfrac{s^2}{2}(\boldsymbol{\Sigma}^{1/2}\mathbf{a}+2\boldsymbol{\Sigma}^{1/2}\mathbf{A}\boldsymbol{\mu})'\mathbf{P}(\mathbf{P}'\mathbf{P}-2s\mathbf{P}'\boldsymbol{\Sigma}^{1/2}\mathbf{A}\boldsymbol{\Sigma}^{1/2}\mathbf{P})^{-1}\mathbf{P}'(\boldsymbol{\Sigma}^{1/2}\mathbf{a}+2\boldsymbol{\Sigma}^{1/2}\mathbf{A}\boldsymbol{\mu})
\end{aligned}$$

or, with $\mathbf{c} = (c_1,\ldots,c_n)' = \mathbf{P}'(\boldsymbol{\Sigma}^{1/2}\mathbf{a}+2\boldsymbol{\Sigma}^{1/2}\mathbf{A}\boldsymbol{\mu})$,

$$E = s(d+\boldsymbol{\mu}'\mathbf{A}\boldsymbol{\mu}+\mathbf{a}'\boldsymbol{\mu}) + \frac{1}{2}s^2\mathbf{c}'(\mathbf{I}-2s\boldsymbol{\Lambda})^{-1}\mathbf{c}.$$

Putting this together with (A.1), (A.2) and (A.3) gives (32.68).

References

Aït-Sahalia, Y., Yu, J.: Saddlepoint Approximations for continuous-time markov processes. J. Econometrics **134**, 507–551 (2006)

Broda, S.A.: The expected shortfall of quadratic portfolios with heavy-tailed risk factors. Math. Finance (forthcoming)

Broda, S.A., Paolella, M.S.: Saddlepoint Approximations for the doubly noncentral t distribution. Comput. Stat. Data Anal. **51**, 2907–2918 (2007)

Broda, S.A., Paolella, M.S.: CHICAGO: A fast and accurate method for portfolio risk calculation. J. Financ. Econometrics **7**(4), 412–436 (2009a)

Broda, S.A., Paolella, M.S.: Evaluating the density of ratios of noncentral quadratic forms in normal variables. Comput. Stat. Data Anal. **53**(4), 1264–1270 (2009b)

Broda, S.A., Paolella, M.S.: Saddlepoint Approximation of Expected Shortfall for Transformed Means, Mimeo, UvA Econometrics Discussion Paper 2010/08, University of Amsterdam (2009c)

Broda, S.A., Paolella, M.S.: Expected shortfall for distributions in finance. In: Cizek, P., Härdle, W., Weron, R. (eds.) Statistical Tools for Finance and Insurance, (2nd ed.), Springer, Berlin (2011)

Butler, R.W.: An Introduction to Saddlepoint Methods, Cambridge University Press, Cambridge (2007)

Butler, R.W., Paolella,M.S.: Approximate Distributions for the various serial correlograms. Bernoulli **4**(4), 497–518 (1998)

Butler, R.W., Paolella, M.S.: Uniform Saddlepoint Approximations for ratios of quadratic forms. Bernoulli **14**(1), 140–154 (2008)

Butler, R.W., Wood, A.T.A.: Saddlepoint Approximation for moment generating functions of truncated random variables. Ann. Stat. **32**, 2712–2730 (2004)

Coleman, M.S., Mansour, A.: Real Estate in the Real World: Dealing with Non-normality and risk in an asset allocation model. J. Real Estate Portfolio Manag. **11**(1), 37–54 (2005)

Collin-Dufresne, P., Goldstein, R.S.: Pricing Swaptions within an affine framework. J. Derivatives **10**, 9–26 (2002)

Daniels, H.E.: Saddlepoint approximation in statistics. Ann. Math. Statist. **25**, 631–650 (1954)

Daniels, H.E.: Tail probability approximation. Int. Stat. Rev. **55**, 37–48 (1987)

Daniels, H.E., Young, G.A.: Saddlepoint approximation for the studentized mean, with an application to the bootstrap. Biometrika **78**, 169–179 (1991)

Dembo, A., Deuschel, J.-D., Duffie, D.: Large Portfolio Losses. Finance Stochast. **8**, 3–16 (2004)

DiCiccio, T.J., Martin, M.A.: Approximations of Marginal tail probabilities for a class of smooth functions with applications to bayesian and conditional inference. Biometrika **78**, 891–902 (1991)

Duffie, D., Pan, J.: Analytical value-at-risk with jumps and credit risk. Finance Stochast. **5**, 155–180 (2001)

Feuerverger, A., Wong, A.C.M.: Computation of value-at-risk for nonlinear portfolios. J. Risk **3**(1), 37–55 (2000)

Geary, R.C.: Extension of a theorem by harald cramér on the frequency distribution of the quotient of two variables. J. Roy. Stat. Soc. **17**, 56–57 (1944)

Gil–Pelaez, J.: Note on the inversion theorem. Biometrika **38**, 481–482 (1951)

Glasserman, P.: Tail approximations for portfolio credit risk. J. Derivatives **12**(2), 24–42 (2004)

Glasserman, P., Heidelberger, P., Shahabuddin, P.: Portfolio value-at-risk with heavy-tailed risk factors. Math. Fin. **12**, 239–269 (2002)

Glasserman, P., Kim, K.-K.: Saddlepoint approximations for affine jump-diffusion models. J. Econ. Dynam. Contr. **33**, 15–36 (2009)

Gordy, M.: Saddlepoint approximation of credit risk. J. Bank. Finance **26**, 1335–1353 (2002)

Harvey, C.R., Siddique, A.: Autoregressive conditional skewness. J. Financ. Quant. Anal. **34**(4), 465–487 (1999)

Jensen, J.L.: Saddlepoint Approximations. Oxford University Press, Oxford (1995)
Kolassa, J.E.: Multivariate saddlepoint tail probability approximations, Ann. Stat. **31**(1), 274–286 (2003)
Lieberman, O.: Saddlepoint Approximation for the Distribution of a ratio of quadratic forms in normal variables. J. Am. Stat. Assoc. **89**(427), 924–928 (1994a)
Lieberman, O.: Saddlepoint Approximation for the least squares estimator in first-order autoregression. Biometrika **81**(4), 807–11 (1994b)
Lugannani, R., Rice, S.O.: Saddlepoint Approximations for the distribution of sums of independent random variables. Adv. Appl. Prob. **12**, 475–490 (1980)
Martin, R.: The Saddlepoint method and portfolio optionalities. Risk Magazine **19**(12), 93–95 (2006)
Martin, R.J., Thompson, K.E., Browne, C.J.: Taking to the saddle. Risk **14**, 91–94 (2001)
Paolella, M.S.: Intermediate Probability: A Computational Approach, Wiley, Chichester (2007)
Premaratne, G., Bera, A.K.: Modeling Asymmetry and Excess Kurtosis in Stock Return Data, Illinois Research & Reference Working Paper 00-123, Department of Economics, University of Illinois (2000)
Rogers, L.C.G., Zane, O.: Saddlepoint Approximations to option prices. Ann. Appl. Probab. **9**(2), 493–503 (1999)
Rouvinez, C.: Going greek with VaR. Risk **10**(2), 57–65 (1997)
Scheffé, H.: The analysis of variance. Wiley, New York (1959)
Skovgaard, I.M.: Saddlepoint expansions for conditional distributions. J. Appl. Prob. **24**, 275–287 (1987)
Stuart, A., Ord, J.K.: Kendall's Advanced Theory of Statistics. vol. 1, Distribution Theory, (6th edn.), Edward Arnold, London (1994)
Temme, N.M.: The uniform asymptotic expansion of a class of integrals related to cumulative distribution functions. SIAM J. Math. Anal. **13**, 239–253 (1982)
Tsionas, E.G.: Bayesian inference in the noncentral student-*t* model. J. Comput. Graph. Stat. **11**(1), 208–221 (2002)
Veilex, L.: Higher Order Large Deviation Approximations Applied to CDO Pricing. Credit Suisse Group. Available on SSRN (2007)
Wang, S.: Saddlepoint approximations for bivariate distributions. J. Appl. Probab. **27**, 586–597 (1990)
Wong, W.K.: Backtesting trading risk of commercial banks using expected shortfall. J. Bank. Financ. **32**, 1404–1415 (2008)
Xiong, J., Wong, A., Salopek, D.: Saddlepoint approximations to option price in a general equilibrium model. Stat. Probab. Lett. **71**(4), 361–369 (2005)
Yang, J., Hurd, T.R., Zhang, X.: Saddlepoint approximation method for pricing CDOs. J. Comput. Finance **10**, 1–20 (2006)
Yueh, M.-L., Wong, M.C.W.: Analytical VaR and expected shortfall for quadratic portfolios. J. Derivatives **17**, 33–44 (2010)

Chapter 33
Bagging, Boosting and Ensemble Methods

Peter Bühlmann

33.1 An Introduction to Ensemble Methods

Ensemble methods aim at improving the predictive performance of a given statistical learning or model fitting technique. The general principle of ensemble methods is to construct a linear combination of some model fitting method, instead of using a single fit of the method.

More precisely, consider for simplicity the framework of function estimation. We are interested in estimating a real-valued function

$$g : \mathbb{R}^d \to \mathbb{R}$$

based on data $(X_1, Y_1), \ldots, (X_n, Y_n)$ where X is a d-dimensional predictor variable and Y a univariate response. Generalizations to other functions $g(\cdot)$ and other data-types are possible. We assume to have specified a *base procedure* which, given some input data (as above), yields an estimated function $\hat{g}(\cdot)$. For example, the base procedure could be a nonparametric kernel estimator (if d is small) or a nonparametric statistical method with some structural restrictions (for $d \geq 2$) such as a regression tree (or class-probability estimates from a classification tree).

We can run a base procedure many times when changing the input data: the original idea of ensemble methods is to use reweighted original data to obtain different estimates $\hat{g}_1(\cdot), \hat{g}_2(\cdot), \hat{g}_3(\cdot), \ldots$ based on different reweighted input data. We can then construct an ensemble-based function estimate $g_{ens}(\cdot)$ by taking linear combinations of the individual function estimates $\hat{g}_k(\cdot)$:

P. Bühlmann (✉)
ETH Zürich, Seminar für Statistik, Zürich, Switzerland
e-mail: buhlmann@stat.math.ethz.ch

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_33,

$$\hat{g}_{ens}(\cdot) = \sum_{k=1}^{M} c_k \hat{g}_k(\cdot), \tag{33.1}$$

where the $\hat{g}_k(\cdot)$ are obtained from the base procedure based on the kth reweighted data-set. For some ensemble methods, e.g. for bagging (see Sect. 35.2), the linear combination coefficients $c_k \equiv 1/M$ are averaging weights; for other methods, e.g. for boosting (see Sect. 35.3), $\sum_{k=1}^{M} c_k$ increases as M gets larger.

Ensemble methods became popular as a relatively simple device to improve the predictive performance of a base procedure. There are different reasons for this: the bagging procedure turns out to be a variance reduction scheme, at least for some base procedures. On the other hand, boosting methods are primarily reducing the (model) bias of the base procedure. This already indicates that bagging and boosting are very different ensemble methods. We will argue in Sects. 33.4.1 and 33.4.7 that boosting may be even viewed as a non-ensemble method which has tremendous advantages over ensemble (or multiple prediction) methods in terms of interpretation.

Random forests (Breiman 2001) is a very different ensemble method than bagging or boosting. The earliest random forest proposal is from Amit and Geman (Amit and Geman 1997). From the perspective of prediction, random forests is about as good as boosting, and often better than bagging. Section 33.4.12 highlights a few more aspects.

Some rather different exposition about bagging and boosting which describes these methods in the much broader context of many other modern statistical methods can be found in Hastie et al. (2001).

33.2 Bagging and Related Methods

Bagging Breiman (1996a), a sobriquet for **b**ootstrap **agg**rega**ting**, is an ensemble method for improving unstable estimation or classification schemes. Breiman Breiman (1996a) motivated bagging as a variance reduction technique for a given base procedure, such as decision trees or methods that do variable selection and fitting in a linear model. It has attracted much attention, probably due to its implementational simplicity and the popularity of the bootstrap methodology. At the time of its invention, only heuristic arguments were presented why bagging would work. Later, it has been shown in Bühlmann and Yu (2002) that bagging is a smoothing operation which turns out to be advantageous when aiming to improve the predictive performance of regression or classification trees. In case of decision trees, the theory in Bühlmann and Yu (2002) confirms Breiman's intuition that bagging is a variance reduction technique, reducing also the mean squared error (MSE). The same also holds for subagging (**sub**sample **agg**rega**ting**), defined in Sect. 33.2.3, which is a computationally cheaper version than bagging. However,

for other (even "complex") base procedures, the variance and MSE reduction effect of bagging is not necessarily true; this has also been shown in Buja and Stuetzle (2006) for the simple case where the estimator is a U-statistics.

33.2.1 Bagging

Consider the regression or classification setting. The data is given as in Sect. 35.1: we have pairs (X_i, Y_i) $(i = 1, \ldots, n)$, where $X_i \in \mathbb{R}^d$ denotes the d-dimensional predictor variable and the response $Y_i \in \mathbb{R}$ (regression) or $Y_i \in \{0, 1, \ldots, J-1\}$ (classification with J classes). The target function of interest is usually $\mathbb{E}[Y|X = x]$ for regression or the multivariate function $\mathbb{P}[Y = j|X = x]$ $(j = 0, \ldots, J-1)$ for classification. The function estimator, which is the result from a given base procedure, is

$$\hat{g}(\cdot) = h_n((X_1, Y_1), \ldots, (X_n, Y_n))(\cdot) : \mathbb{R}^d \to \mathbb{R},$$

where the function $h_n(\cdot)$ defines the estimator as a function of the data.

Bagging is defined as follows.

Bagging Algorithm

Step 1. Construct a bootstrap sample $(X_1^*, Y_1^*), \ldots, (X_n^*, Y_n^*)$ by randomly drawing n times with replacement from the data $(X_1, Y_1), \ldots, (X_n, Y_n)$.

Step 2. Compute the bootstrapped estimator $\hat{g}^*(\cdot)$ by the plug-in principle: $\hat{g}^*(\cdot) = h_n((X_1^*, Y_1^*), \ldots, (X_n^*, Y_n^*))(\cdot)$.

Step3. Repeat steps 1 and 2 M times, where M is often chosen as 50 or 100, yielding $\hat{g}^{*k}(\cdot)$ $(k = 1, \ldots, M)$. The bagged estimator is $\hat{g}_{Bag}(\cdot) = M^{-1} \sum_{k=1}^{M} \hat{g}^{*k}(\cdot)$.

In theory, the bagged estimator is

$$\hat{g}_{Bag}(\cdot) = \mathbb{E}^*[\hat{g}^*(\cdot)]. \tag{33.2}$$

The theoretical quantity in (33.2) corresponds to $M = \infty$: the finite number M in practice governs the accuracy of the Monte Carlo approximation but otherwise, it shouldn't be viewed as a tuning parameter for bagging. Whenever we discuss properties of bagging, we think about the theoretical version in (33.2).

This is exactly Breiman's Breiman (1996a) definition for bagging regression estimators. For classification, we propose to average the bootstrapped probabilities $\hat{g}_j^{*k}(\cdot) = \hat{\mathbf{P}}^*[Y^{*k} = j|X^{*k} = \cdot]$ $(j = 0, \ldots, J-1)$ yielding an estimator for $\mathbf{P}[Y = j|X = \cdot]$, whereas Breiman Breiman (1996a) proposed to vote among classifiers for constructing the bagged classifier.

The empirical fact that bagging improves the predictive performance of regression and classification trees is nowadays widely documented (Borra and Di Ciaccio 2002; Breiman 1996a,b; Bühlmann and Yu 2002; Buja and Stuetzle 2006). To give an idea about the gain in performance, we cite some of the results of Breiman's pioneering paper Breiman (1996a): for 7 classification problems, bagging a classification tree improved over a single classification tree (in terms of cross-validated misclassification error) by

$$33\%, 47\%, 30\%, 23\%, 20\%, 22\%, 27\%;$$

in case of 5 regression data sets, bagging regression trees improved over a single regression tree (in terms of cross-validated squared error) by

$$39\%, 22\%, 46\%, 30\%, 38\%.$$

In both cases, the size of the single decision tree and of the bootstrapped trees was chosen by optimizing a tenfold cross-validated error, i.e. using the "usual" kind of tree procedure. Besides that the reported improvement in percentages is quite impressive, it is worth pointing out that bagging a decision tree is almost never worse (in terms of predictive power) than a single tree.

A trivial equality indicates the somewhat unusual approach of using the bootstrap methodology:

$$\hat{g}_{Bag}(\cdot) = \hat{g}(\cdot) + (\mathbb{E}^*[\hat{g}^*(\cdot)] - \hat{g}(\cdot)) = \hat{g}(\cdot) + \mathrm{Bias}^*(\cdot),$$

where $\mathrm{Bias}^*(\cdot)$ is the bootstrap bias estimate of $\hat{g}(\cdot)$. Instead of the usual bias correction with a negative sign, bagging comes along with the wrong sign and *adds* the bootstrap bias estimate. Thus, we would expect that bagging has a higher bias than $\hat{g}(\cdot)$, which we will argue to be true in some sense, see Sect. 33.2.2. But according to the usual interplay between bias and variance in nonparametric statistics, the hope is to gain more by reducing the variance than increasing the bias, so that overall, bagging would pay-off in terms of the MSE. Again, this hope turns out to be true for some base procedures. In fact, Breiman Breiman (1996a) described heuristically the performance of bagging as follows: the variance of the bagged estimator $\hat{g}_{Bag}(\cdot)$ should be equal or smaller than that for the original estimator $\hat{g}(\cdot)$; and there can be a drastic variance reduction if the original estimator is "unstable".

33.2.2 Unstable Estimators with Hard Decision Indicator

Instability often occurs when hard decisions with indicator functions are involved as in regression or classification trees. One of the main underlying ideas why bagging works can be demonstrated by a simple example.

Toy Example: A Simple, Instructive Analysis

Consider the estimator

$$\hat{g}(x) = \mathbf{1}_{[\overline{Y}_n \le x]}, \ x \in \mathbb{R}, \tag{33.3}$$

where $\overline{Y}_n = n^{-1} \sum_{i=1}^n Y_i$ with $Y_1, \ldots, Y_n$ i.i.d. (no predictor variables X_i are used for this example). The target we have in mind is $g(x) = \lim_{n\to\infty} \mathbb{E}[\hat{g}(x)]$. A simple yet precise analysis below shows that bagging is a smoothing operation. Due to the central limit theorem we have

$$n^{1/2}(\overline{Y}_n - \mu) \to_D \mathcal{N}(0, \sigma^2) \ (n \to \infty) \tag{33.4}$$

with $\mu = \mathbb{E}[Y_1]$ and $\sigma^2 = Var(Y_1)$. Then, for x in a $n^{-1/2}$-neighborhood of μ,

$$x = x_n(c) = \mu + c\sigma n^{-1/2}, \tag{33.5}$$

we have the distributional approximation

$$\hat{g}(x_n(c)) \to_D L(Z) = \mathbf{1}_{[Z \le c]} \ (n \to \infty), \ Z \sim \mathcal{N}(0, 1). \tag{33.6}$$

Obviously, for a fixed c, this is a hard decision function of Z. On the other hand, averaging for the bagged estimator looks as follows. Denote by $\Phi(\cdot)$ the c.d.f. of a standard normal distribution:

$$\begin{aligned} \hat{g}_{Bag}(x_n(c)) &= \mathbb{E}^*[\mathbf{1}_{[\overline{Y}_n^* \le x_n(c)]}] = \mathbb{E}^*[\mathbf{1}_{[n^{1/2}(\overline{Y}_n^* - \overline{Y}_n)/\sigma \le n^{1/2}(x_n(c) - \overline{Y}_n)/\sigma]}] \\ &= \Phi(n^{1/2}(x_n(c) - \overline{Y}_n)/\sigma) + o_P(1) \\ &\to_D L_{Bag}(Z) = \Phi(c - Z) \ (n \to \infty), \ Z \sim \mathcal{N}(0, 1), \end{aligned} \tag{33.7}$$

where the first approximation (second line) follows because the bootstrap is consistent for the arithmetic mean $\overline{Y}_n$, i.e.,

$$\sup_{x \in \mathbb{R}} |\mathbb{P}^*[n^{1/2}(\overline{Y}_n^* - \overline{Y}_n)/\sigma \le x] - \Phi(x)| = o_P(1) \ (n \to \infty), \tag{33.8}$$

and the second approximation (third line in (33.7)) holds, because of (33.4) and the definition of $x_n(c)$ in (33.5). Comparing with (33.6), bagging produces a soft decision function $L_{Bag}(\cdot)$ of Z: it is a shifted inverse probit, similar to a sigmoid-type function. Figure 33.1 illustrates the two functions $L(\cdot)$ and $L_{Bag}(\cdot)$.

We see that bagging is a smoothing operation. The amount of smoothing is determined "automatically" and turns out to be very reasonable (we are not claiming any optimality here). The effect of smoothing is that bagging reduces variance due to a soft- instead of a hard-thresholding operation.

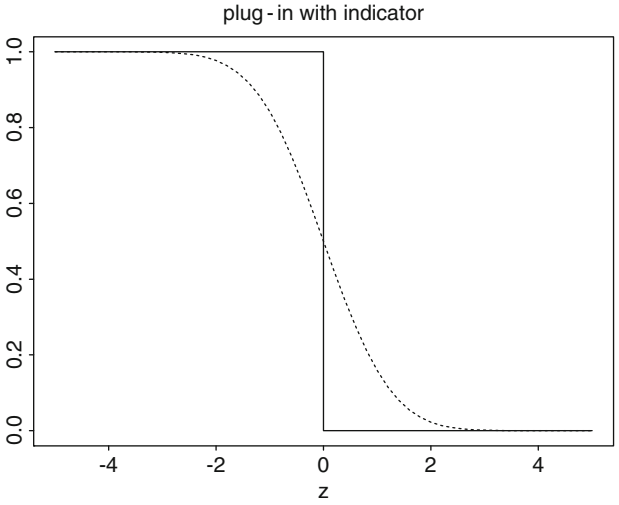


Fig. 33.1 Indicator estimator from (33.3) at $x = x_n(0)$ as in (33.5). Function $L(z) = \mathbf{1}_{[z \le 0]}$ (*solid line*) and $L_{Bag}(z)$ (*dotted line*) defining the asymptotics of the estimator in (33.6) and its bagged version in (33.7)

We can compute the first two asymptotic moments in the unstable region with $x = x_n(c)$.

Numerical evaluations of these first two moments and the mean squared error (MSE) are given in Fig. 33.2. We see that in the approximate range where $|c| \le 2.3$, bagging improves the asymptotic MSE. The biggest gain, by a factor 3, is at the most unstable point $x = \mu = \mathbb{E}[Y_1]$, corresponding to $c = 0$. The squared bias with bagging has only a negligible effect on the MSE (note the different scales in Fig. 33.2). Note that we always give an a-priori advantage to the original estimator which is asymptotically unbiased for the target as defined.

In Bühlmann and Yu (2002), this kind of analysis has been given for more general estimators than $\overline{Y}_n$ in (33.3) and also for estimation in linear models after testing. Hard decision indicator functions are involved there as well and bagging reduces variance due to its smoothing effect. The key to derive this property is always the fact that the bootstrap is asymptotically consistent as in (33.8).

Regression Trees

We address here the effect of bagging in the case of decision trees which are most often used in practice in conjunction with bagging. Decision trees consist of piecewise constant fitted functions whose supports (for the piecewise constants) are given by indicator functions similar to (33.3). Hence we expect bagging to bring a significant variance reduction as in the toy example above.

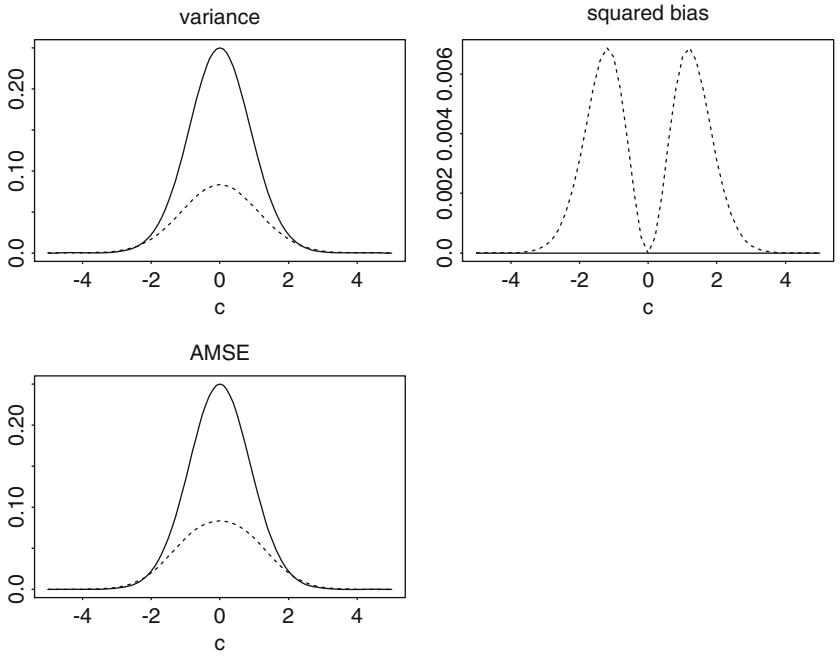


Fig. 33.2 Indicator estimator from (33.3) at $x = x_n(c)$ as in (33.5). Asymptotic variance, squared bias and mean squared error (*AMSE*) (the target is $\lim_{n\to\infty} \mathbb{E}[\hat{g}(x)]$) for the estimator $\hat{g}(x_n(c))$ from (33.3) (*solid line*) and for the bagged estimator $\hat{g}_{Bag}(x_n(c))$ (*dotted line*) as a function of c

For simplicity of exposition, we consider first a one-dimensional predictor space and a so-called regression stump which is a regression tree with one split and two terminal nodes. The stump estimator (or algorithm) is then defined as the decision tree,

$$\hat{g}(x) = \hat{\beta}_\ell \mathbf{1}_{[x<\hat{d}]} + \hat{\beta}_u \mathbf{1}_{[x\geq\hat{d}]} = \hat{\beta}_\ell + (\hat{\beta}_u - \hat{\beta}_\ell)\mathbf{1}_{[\hat{d}\leq x]}, \tag{33.9}$$

where the estimates are obtained by least squares as

$$(\hat{\beta}_\ell, \hat{\beta}_u, \hat{d}) = \text{argmin}_{\beta_\ell,\beta_u,d} \sum_{i=1}^{n} (Y_i - \beta_\ell \mathbf{1}_{[X_i<d]} - \beta_u \mathbf{1}_{[X_i\geq d]})^2.$$

These values are estimates for the best projected parameters defined by

$$(\beta_\ell^0, \beta_u^0, d^0) = \text{argmin}_{\beta_\ell,\beta_u,d} \mathbb{E}[(Y - \beta_\ell \mathbf{1}_{[X<d]} - \beta_u \mathbf{1}_{[X\geq d]})^2]. \tag{33.10}$$

The main mathematical difference of the stump in (33.9) to the toy estimator in (33.3) is the behavior of $\hat{d}$ in comparison to the behavior of $\overline{Y}_n$ (and not the

constants $\hat{\beta}_\ell$ and $\hat{\beta}_u$ involved in the stump). It is shown in Bühlmann and Yu (2002) that $\hat{d}$ has convergence rate $n^{-1/3}$ (in case of a smooth regression function) and a limiting distribution which is non-Gaussian. This also explains that the bootstrap is not consistent, but consistency as in (33.8) turned out to be crucial in our analysis above. Bagging is still doing some kind of smoothing, but it is not known how this behaves quantitatively. However, a computationally attractive version of bagging, which has been found to perform often as good as bagging, turns out to be more tractable from a theoretical point of view.

33.2.3 Subagging

Subagging is a sobriquet for **sub**sample **agg**regat**ing** where subsampling is used instead of the bootstrap for the aggregation. An estimator $\hat{g}(\cdot) = h_n((X_1, Y_1), \ldots, (X_n, Y_n))(\cdot)$ is aggregated as follows:

$$\hat{g}_{SB(m)}(\cdot) = \binom{n}{m}^{-1} \sum_{(i_1,\ldots,i_m)\in\mathcal{I}} h_m((X_{i_1}, Y_{i_1}), \ldots, (X_{i_m}, Y_{i_m}))(\cdot),$$

where $\mathcal{I}$ is the set of m-tuples ($m < n$) whose elements in $\{1, \ldots, n\}$ are all distinct. This aggregation can be approximated by a stochastic computation. The subagging algorithm is as follows.

Subagging Algorithm

Step 1. For $k = 1, \ldots, M$ (e.g. $M = 50$ or 100) do:

(i) Generate a random subsample $(X_1^{*k}, Y_1^{*k}), \ldots, (X_m^{*k}, Y_m^{*k})$ by randomly drawing m times without replacement from the data $(X_1, Y_1), \ldots, (X_n, Y_n)$ (instead of resampling with replacement in bagging).
(ii) Compute the subsampled estimator $\hat{g}_{(m)}^{*k}(\cdot) = h_m((X_1^{*k}, Y_1^{*k}), \ldots, (X_m^{*k}, Y_m^{*k}))(\cdot)$.

Step 2. Average the subsampled estimators to approximate $\hat{g}_{SB(m)}(\cdot) \approx M^{-1} \sum_{k=1}^{M} \hat{g}_{(m)}^{*k}(\cdot)$.

As indicated in the notation, subagging depends on the subsample size m which is a tuning parameter (in contrast to M).

An interesting case is *half subagging* with $m = \lfloor n/2 \rfloor$. More generally, we could also use $m = \lfloor an \rfloor$ with $0 < a < 1$ (i.e. m a fraction of n) and we will argue why the usual choice $m = o(n)$ in subsampling for distribution estimation Politis et al. (1999) is a bad choice. Half subagging with $m = [n/2]$ has been studied

also in Buja and Stuetzle (2006): in case where $\hat{g}$ is a U-statistic, half subagging is exactly equivalent to bagging, and subagging yields very similar empirical results to bagging when the estimator $\hat{g}(\cdot)$ is a decision tree. Thus, if we don't want to optimize over the tuning parameter m, a good choice in practice is very often $m = \lfloor n/2 \rfloor$. Consequently, half subagging typically saves more than half of the computing time because the computational order of an estimator $\hat{g} = \hat{g}_{(n)}$ is usually at least linear in n.

Subagging Regression Trees

We describe here in a non-technical way the main mathematical result from Bühlmann and Yu (2002) about subagging regression trees.

The underlying assumptions for some mathematical theory are as follows. The data generating regression model is

$$Y_i = g(X_i) + \varepsilon_i,\ i = 1, \ldots, n,$$

where $X_1, \ldots, X_n$ and $\varepsilon_1, \ldots, \varepsilon_n$ are i.i.d. variables, independent from each other, and $\mathbb{E}[\varepsilon_1] = 0, \mathbb{E}|\varepsilon_1|^2 < \infty$. The regression function $g(\cdot)$ is assumed to be smooth and the distribution of X_i and ε_i are assumed to have suitably regular densities.

It is then shown in Bühlmann and Yu (2002) that for $m = \lfloor an \rfloor$ $(0 < a < 1)$,

$$\limsup_{n \to \infty} \frac{\mathbb{E}[(\hat{g}_{SB(m)}(x) - g(x))^2]}{\mathbb{E}[(\hat{g}_n(x) - g(x))^2]} < 1,$$

for x in suitable neighborhoods (depending on the fraction a) around the best projected split points of a regression tree (e.g. the parameter d^0 in (33.10) for a stump), and where $g(x) = \lim_{n \to \infty} \mathbb{E}[\hat{g}(x)]$. That is, subagging asymptotically reduces the MSE for x in neighborhoods around the unstable split points, a fact which we may also compare with Fig. 33.2. Moreover, one can argue that globally,

$$\mathbb{E}[(\hat{g}_{SB(m)}(X) - g(X))^2] \overset{\text{approx.}}{<} \mathbb{E}[(\hat{g}(X) - g(X))^2]$$

for n large, and where the expectations are taken also over (new) predictors X.

For subagging with small order $m = o(n)$, such a result is no longer true: the reason is that small order subagging will then be dominated by a large bias (while variance reduction is even better than for fraction subagging with $m = \lfloor an \rfloor$, $0 < a < 1$).

Similarly as for the toy example in Sect. 33.2.2, subagging smoothes the hard decisions in a regression tree resulting in reduced variance and MSE.

33.2.4 Bagging More "Smooth" Base Procedures and Bragging

As discussed in Sects. 33.2.2 and 33.2.3, (su-)bagging smoothes out indicator functions which are inherent in some base procedures such as decision trees. For base procedures which are "smoother", e.g. which do not involve hard decision indicators, the smoothing effect of bagging is expected to cause only small effects.

For example, in Buja and Stuetzle (2006) it is proved that the effect of bagging on the MSE is only in the second order term if the base procedure is a U-statistic. Similarly, citing Chen and Hall (2003): "... when bagging is applied to relatively conventional statistical problems, it cannot reliably be expected to improve performance". On the other hand, we routinely use nowadays "non-conventional" methods: a simple example is variable selection and fitting in a linear model where bagging has been demonstrated to improve predictive performance (Breiman 1996a).

In Borra and Di Ciaccio (2002), the performance of bagging has been studied for MARS, projection pursuit regression and regression tree base procedures: most improvements of bagging are reported for decision trees. In Bühlmann and Yu (2002), it is shown that bagging the basis function in MARS essentially doesn't change the asymptotic MSE. In Bühlmann (2003) it is empirically demonstrated in greater detail that for finite samples, bagging MARS is by far less effective - and sometimes very destructive - than bagging decision trees.

(Su-)bagging may also have a positive effect due to averaging over different selected predictor variables; this is an additional effect besides smoothing out indicator functions. In case of MARS, we could also envision that such an averaging over different selected predictor variables would have a positive effect: in the empirical analysis in Bühlmann (2003), this has been found to be only true when using a robust version of aggregation, see below.

33.2.5 Bragging

Bragging stands for **b**ootstrap **r**obust **agg**regat**ing** (Bühlmann 2003): it uses the sample median over the M bootstrap estimates $\hat{g}^{*k}(\cdot)$, instead of the sample mean in Step 3 of the bagging algorithm.

While bragging regression trees was often found to be slightly less improving than bagging, bragging MARS seems better than the original MARS and much better than bagging MARS.

33.2.6 Out-of-bag Error Estimation

Bagging "automatically" yields an estimate of the out-of-sample error, sometimes referred to as the generalization error. Consider a loss $\rho(Y, \hat{g}(X))$, measuring the

discrepancy between an estimated function $\hat{g}$, evaluated at X, and the corresponding response Y, e.g. $\rho(Y, \hat{g}(X)) = |Y - \hat{g}(X)|^2$. The generalization error is then

$$err = \mathbb{E}[\rho(Y, \hat{g}(X))],$$

where the expectation $\mathbb{E}$ is over the training data $(X_1, Y_1), \ldots, (X_n, Y_n)$ (i.i.d. or stationary pairs), $\hat{g}(\cdot)$ a function of the training data, and (X, Y) is a new test observation, independent from the training data but having the same distribution as one training sample point (X_i, Y_i).

In a bootstrap sample (in the bagging procedure), roughly $exp(-1) \approx 37\%$ of the original observations are left out: they are called "out-of-bag" observations (Breiman 1996b). Denote by $\mathcal{B}oot^k$ the original sample indices which were resampled in the kth bootstrap sample; note that the out-of-bag sample observations (in the kth bootstrap resampling stage) are then given by $\{1, \ldots, n\} \setminus \mathcal{B}oot^k$ which can be used as test sets. The out-of-bag error estimate of bagging is then defined as

$$\widehat{err}_{OB} = n^{-1} \sum_{i=1}^{n} N_M^{-1} \sum_{k=1}^{M} \mathbf{1}_{[(X_i, Y_i) \notin \mathcal{B}oot^k]} \rho(Y_i, \hat{g}^{*k}(X_i)),$$

$$N_M = \sum_{k=1}^{M} \mathbf{1}_{[(X_i, Y_i) \notin \mathcal{B}oot^k]}.$$

In Bylander (2002), a correction of the out-of-bag error estimate is proposed. Out-of-bag estimation can also be used for other tasks, e.g. for more honest class probability estimates in classification when bagging trees (Breiman 1996b).

33.2.7 Disadvantages

The main disadvantage of bagging, and other ensemble algorithms, is the lack of interpretation. A linear combination of decision trees is much harder to interpret than a single tree. Likewise: bagging a variable selection - fitting algorithm for linear models (e.g. selecting the variables using the AIC criterion within the least-squares estimation framework) gives little clues which of the predictor variables are actually important.

One way out of this lack of interpretation is sometimes given within the framework of bagging. In Efron and Tibshirani (1998), the bootstrap has been justified to judge the importance of automatically selected variables by looking at relative appearance-frequencies in the bootstrap runs. The bagging estimator is the average of the fitted bootstrap functions, while the appearance frequencies of selected variables or interactions may serve for interpretation.

33.2.8 Other References

Bagging may also be useful as a "module" in other algorithms: BagBoosting Bühlmann and Yu (2000) is a boosting algorithm (see Sect. 35.3) with a bagged base-procedure, often a bagged regression tree. The theory about bagging supports the finding that BagBoosting using bagged regression trees, which have smaller asymptotic MSEs than trees, is often better than boosting with regression trees. This is empirically demonstrated for a problem about tumor classification using microarray gene expression predictors (Dettling 2004).

In Ridgeway (2002), bagging is used in conjunction with boosting (namely for stopping boosting iterations) for density estimation. In Dudoit and Fridlyand (2003), bagging is used in the unsupervised context of cluster analysis, reporting improvements when using bagged clusters instead of original cluster-outputs.

33.3 Stability Selection

Subsampling or bootstrapping are simple but effective techniques for increasing "stability" of a method. In Sect. 35.2 we discussed bagging to potentially improve the prediction performance of an algorithm or statistical estimator. Here, we will briefly argue that subsampling or bootstrapping and aggregation leads to increased power for variable selection and for controlling the expected number of false positive selections.

To simplify the exposition, we consider data

$$(X_1, Y_1), \ldots, (X_n, Y_n) \text{ i.i.d.},$$

where X_i is a d-dimensional covariate and Y_i a univariate response. The goal is to select the set of active variables

$$S = \{1 \le j \le d;\ X^{(j)} \text{ is associated with } Y\}. \tag{33.11}$$

Here and in the sequel, $x^{(j)}$ denotes the jth component of the vector x. The wording "associated to" is very loose, of course. Depending on the context, we can use different definitions. For example, in a linear model

$$Y = \sum_{j=1}^{p} \beta_j X^{(j)} + \varepsilon,$$

we would define $S = \{1 \le j \le d;\ \beta_j \ne 0\}$. Similarly, we can use the same definition for S in a generalized linear model with regression coefficients $\beta_1, \ldots, \beta_d$.

33.3.1 Subsampling of Selection Procedure

We assume that we have specified an active set S as in (33.11) and we consider a statistical method or algorithm $\hat{S}$ for estimating S. As in Sect. 33.2.3, we use subsampling with subsample size $\lfloor n/2 \rfloor$. This yields a subsampled selection estimate $\hat{S}^*$ and we can compute the selection probability from subsampling, for each variable $j \in \{1, \ldots, d\}$:

$$\hat{\pi}_j = \mathbf{P}^*[j \in \hat{S}^*], \ j = 1, \ldots, d.$$

As in Sect. 33.2.3, we compute $\hat{\pi}_j$ by a stochastic approximation. Run the subsampling M times, producing $\hat{S}^{*1}, \ldots, \hat{S}^{*M}$ and use the right-hand side of the following formula

$$\hat{\pi}_j \approx M^{-1} \sum_{b=1}^{M} \mathbf{1}_{[j \in \hat{S}^{*b}]},$$

as an approximation for the left-hand side. Thus, the selection probabilities $\hat{\pi}_j$ are obtained by aggregating the individual selectors $\hat{S}^{*b}$ from many subsampling runs $b = 1, \ldots, M$, where M is large, e.g. $M = 100$.

The set of stable selections is defined as:

$$\hat{S}_{stable}(\pi_{thr}) = \{1 \le j \le d; \ \hat{\pi}_j \ge \pi_{thr}\}, \tag{33.12}$$

where π_{thr} is a tuning parameter to be chosen. We refer to $\hat{S}_{stable}(\pi_{thr})$ also as "stability selection" (Meinshausen and Bühlmann 2010).

As described next, the choice of the tuning parameter should be governed by controlling some false positive error measure.

33.3.2 Controlling False Positive Selections

Denote by $V = V(\pi_{thr}) = \hat{S}_{stable}(\pi_{thr}) \cap S^c$ the number of false positives with stability selection. Assuming some exchangeability condition on the design or covariates, which is rather restrictive, and requiring that the selection procedure $\hat{S}$ is performing better than random guessing, a very simple formula controls the expected number of false positive selections:

$$\mathbb{E}[V(\pi_{thr})] \le frac12\pi_{thr} - 1\frac{q^2}{d},$$

where q is an upper bound for the selection algorithm $|\hat{S}_{\lfloor n/2 \rfloor}| \leq q$ based on $\lfloor n/2 \rfloor$ observations. For example, the selector $\hat{S}$ is a forward selection algorithm which stops when the first q variables have been selected. More details are given in Meinshausen and Bühlmann (2010).

33.3.3 Related Work

Theoretical and empirical results are derived in Meinshausen and Bühlmann (2010) showing that randomizing covariates is often beneficial for improved variable or feature selection. We note that randomizing covariates has also been successfully used in Random Forests Breiman (2001). Another relation to subsampling as described in Sect. 33.3.1 is given by multiple sample-splitting: this technique has been used for deriving p-values in high-dimensional regression models (Meinshausen et al. 2009).

33.4 Boosting

Boosting algorithms have been proposed in the machine learning literature by Schapire (Schapire 1990) and Freund (Freund 1995; Freund and Schapire 1996), see also Schapire (2002). These first algorithms have been developed as ensemble methods. Unlike bagging which is a parallel ensemble method, boosting methods are sequential ensemble algorithms where the weights c_k in (33.1) are depending on the previous fitted functions $\hat{g}_1, \ldots, \hat{g}_{k-1}$. Boosting has been empirically demonstrated to be very accurate in terms of classification, notably the so-called AdaBoost algorithm (Freund and Schapire 1996). A review of boosting from a statistical perspective is given in Bühlmann and Hothorn (2007) where many of the concepts and algorithms are illustrated with the `R`-software package `mboost` (Hothorn et al. 2010).

We will explain below that boosting can be viewed as a nonparametric optimization algorithm in function space, as first pointed out by Breiman (Breiman 1998, 1999). This view turns out to be very fruitful to adapt boosting for other problems than classification, including regression and survival analysis.

Maybe it is worth mentioning here that boosting algorithms have often better predictive power than bagging, cf. Breiman (1998); of course, such a statement has to be read with caution, and methods should be tried out on individual data-sets, including e.g. cross-validation, before selecting one among a few methods.

To give an idea, we report here some empirical results from Breiman (1998) for classification: we show below the gains (in percentage) of boosting trees over bagging trees:

"normal" size data-sets: 64.3%, 10.8%, 20.3%, −4.6%, 6.9%, 16.2%,

large data-sets: 37.5%, 12.6%, −50.0%, 4.0%, 28.6%.

For all data-sets, boosting trees was better than a single classification tree. The biggest loss of 50% for boosting in comparison with bagging is for a data-set with very low misclassification error, where bagging achieves 0.014% and boosting 0.021%.

There is a striking similarity between gradient based boosting and the Lasso in linear or generalized linear models, as we will describe in Sect. 33.4.10. Thus, despite substantial conceptual differences, boosting-type algorithms are implicitly related to ℓ_1-regularization.

33.4.1 Boosting as Functional Gradient Descent

Rather than looking through the lenses of ensemble methods, boosting algorithms can be seen as functional gradient descent techniques (Breiman 1998, 1999). The goal is to estimate a function $g : \mathbb{R}^d \to \mathbb{R}$, minimizing an expected loss

$$\mathbb{E}[\rho(Y, g(X))], \ \rho(\cdot, \cdot) : \mathbb{R} \times \mathbb{R} \to \mathbb{R}^+, \tag{33.13}$$

based on data (X_i, Y_i) $(i = 1, \ldots n)$ as in Sect. 33.2.1. The loss function ρ is typically assumed to be convex in the second argument. We consider here both cases where the univariate response Y is continuous (regression problem) or discrete (classification problem), since boosting is potentially useful in both cases.

As we will see in Sect. 33.4.2, boosting algorithms are pursuing a "small" empirical risk

$$n^{-1} \sum_{i=1}^{n} \rho(Y_i, g(X_i))$$

by selecting a g in the linear hull of some function class, i.e. $g(\cdot) = \sum_k c_k g_k(\cdot)$ with $g_k(\cdot)$'s from a function class such as trees.

The most popular loss functions, for regression and binary classification, are given in Table 33.1.

Table 33.1 The squared error, binomial negative log-likelihood and exponential loss functions and their population minimizers; $\text{logit}(p) = \log(p/(1-p))$

Boosting	Loss function	Population minimizer for (33.13)
L_2Boost	$\rho(y, g) = (y - g)^2$	$g(x) = \mathbb{E}[Y \vert X = x]$
LogitBoost	$\rho(y, g) = log_2(1 + \exp(-2(y - 1)g))$	$g(x) = 0.5 \cdot \text{logit}(\mathbf{P}[Y = 1 \vert X = x])$
AdaBoost	$\rho(y, g) = \exp(-(2y - 1)g)$	$g(x) = 0.5 \cdot \text{logit}(\mathbf{P}[Y = 1 \vert X = x])$

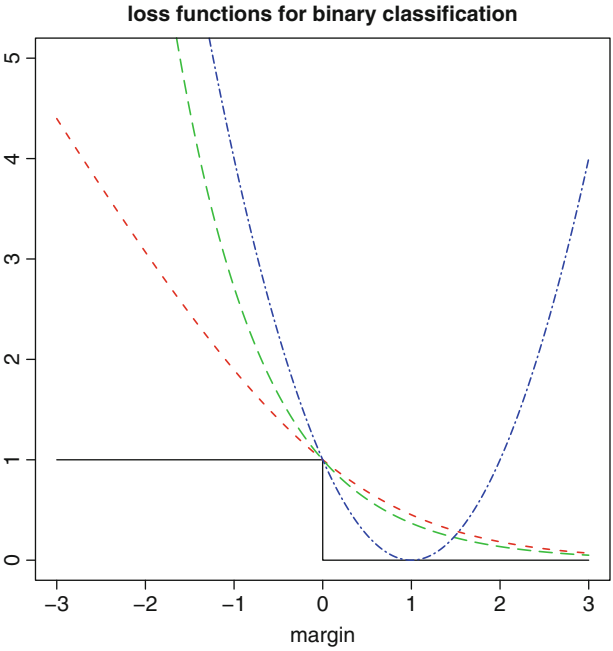


Fig. 33.3 Loss functions of the margin for binary classification. Zero-one misclassification loss (*black*), log-likelihood loss (*red*), exponential loss (*green*), squared error loss (*blue*). The loss-functions are described in Table 33.1

While the squared error loss is mainly used for regression (see Bühlmann and Yu (2003) for classification with the squared error loss), the log-likelihood and the exponential loss are for binary classification only.

The Margin for Classification

The form of the log-likelihood loss may be somewhat unusual: we norm it, by using the base 2 so that it "touches" the misclassification error as an upper bound (see Fig. 33.3), and we write it as a function of the so-called *margin* $\tilde{y}g$, where $\tilde{y} = 2y - 1 \in \{-1, 1\}$ is the usual labeling from the machine learning community. Thus, the loss is a function of the margin $\tilde{y}g$ only; and the same is true with the exponential loss and also the squared error loss for classification since

$$(\tilde{y} - g)^2 = \tilde{y}^2 - 2\tilde{y}g + g^2 = 1 - 2\tilde{y}g + (\tilde{y}g)^2,$$

using $\tilde{y}^2 = 1$.

The misclassification loss, or zero-one loss, is $\mathbf{1}_{[\tilde{y}g<0]}$, again a function of the margin, whose population minimizer is $g(x) = \mathbf{1}_{[\mathbb{P}[Y=1|X=x]>1/2]}$. For readers less

familiar with the concept of the margin, this can also be understood as follows: the Bayes-classifier which minimizes the misclassification risk is

$$g_{Bayes}(x) = \mathbf{1}_{[\mathbb{P}[Y=1|X=x]>1/2]}.$$

We can now see that a misclassification occurs, if $y = 0$, $g_{Bayes}(x) = 1$ or $y = 1$, $g_{Bayes}(x) = 0$, which is equivalent to $2(y-1)g_{Bayes}(x) < 0$ or $\tilde{y}g_{Bayes}(x) < 0$.

The (surrogate) loss functions given in Table 33.1 are all convex functions of the margin $\tilde{y}g$ which bound the zero-one misclassification loss from above, see Fig. 33.3. The convexity of these surrogate loss functions is computationally important for empirical risk minimization; minimizing the empirical zero-one loss is computationally intractable.

33.4.2 The Generic Boosting Algorithm

Estimation of the function $g(\cdot)$, which minimizes an expected loss in (33.13), is pursued by a constrained minimization of the empirical risk $n^{-1}\sum_{i=1}^{n}\rho(Y_i, g(X_i))$. The constraint comes in algorithmically (and not explicitly), by the way we are attempting to minimize the empirical risk, with a so-called functional gradient descent. This gradient descent view has been recognized and refined by various authors (cf. Breiman 1998, 1999; Bühlmann and Yu 2003; Friedman 2001; Friedman et al. 2000; Mason et al. 2000). In summary, the minimizer of the empirical risk is imposed to satisfy a "smoothness" constraint in terms of a linear expansion of ("simple") fits from a real-valued base procedure function estimate.

Generic Functional Gradient Descent

Step 1 (initialization). Given data $\{(X_i, Y_i); i = 1, \ldots, n\}$, apply the base procedure yielding the function estimate

$$\hat{F}_1(\cdot) = \hat{g}(\cdot),$$

where $\hat{g} = \hat{g}_{X,Y} = h_n((X_1, Y_1), \ldots, (X_n, Y_n))$ is a function of the original data. Set $m = 1$.

Step 2 (projecting gradient to learner). Compute the negative gradient vector

$$U_i = -\frac{\partial\rho(Y_i, g)}{\partial g}\Big|_{g=\hat{F}_m(X_i)},\ i = 1, \ldots, n,$$

evaluated at the current $\hat{F}_m(\cdot)$. Then, apply the base procedure to the gradient vector

$$\hat{g}_{m+1}(\cdot),$$

where $\hat{g}_{m+1} = \hat{g}_{X,U} = h_n((X_1, U_1), \ldots, (X_n, U_n))$ is a function of the original predictor variables and the current negative gradient vector as pseudo-response.

Step 3 (line search). Do a one-dimensional numerical search for the best step-size

$$\hat{s}_{m+1} = \text{argmin}_s \sum_{i=1}^{n} \rho(Y_i, \hat{F}_m(X_i) + s\hat{g}_{m+1}(X_i)).$$

Update,

$$\hat{F}_{m+1}(\cdot) = \hat{F}_m(\cdot) + \hat{s}_{m+1}\hat{g}_{m+1}(\cdot).$$

Step 4 (iteration). Increase m by one and repeat Steps 2 and 3 until a stopping iteration M is achieved.

The number of iterations M is the tuning parameter of boosting. The larger it is, the more complex the estimator. But the complexity, for example the variance of the estimator, is not linearly increasing in M: instead, it increases very slowly as M gets larger, see also Fig. 33.4 in Sect. 33.4.6.

Obviously, the choice of the base procedure influences the boosting estimate. Originally, boosting has been mainly used with tree-type base procedures, typically with small trees such as stumps (two terminal nodes) or trees having say 8 terminal nodes (cf. Bauer and Kohavi 1999; Breiman 1998, 2004; Dettling and Bühlmann 2003; Friedman et al. 2000); see also Sect. 33.4.9. But we will demonstrate in Sect. 33.4.7 that boosting may be very worthwhile within the class of linear, additive or interaction models, allowing for good model interpretation.

The function estimate $\hat{g}_{m+1}$ in Step 2 can be viewed as an estimate of $\mathbb{E}[U_i|X = x]$, the expected negative gradient given the predictor X, and takes values in $\mathbb{R}$, even in case of a classification problem with Y_i in a finite set (this is different from the AdaBoost algorithm, see below).

We call $\hat{F}_M(\cdot)$ the L_2Boost-, LogitBoost- or AdaBoost-estimate, according to the implementing loss function $(y - g)^2$, $log_2(1 + \exp(-2(y - 1)g))$ or $\rho(y, g) = \exp(-(2y - 1)g)$, respectively; see Table 33.1.

The original AdaBoost algorithm for classification is actually a bit different: the base procedure fit is a classifier, and not a real-valued estimator for the conditional probability of Y given X; and Steps 2 and 3 are also somewhat different. Since AdaBoost's implementing exponential loss function is not well established in statistics, we refer for a detailed discussion to Friedman et al. (2000). From a statistical perspective, the squared error loss and log-likelihood loss functions are most prominent and we describe below the corresponding boosting algorithms in detail.

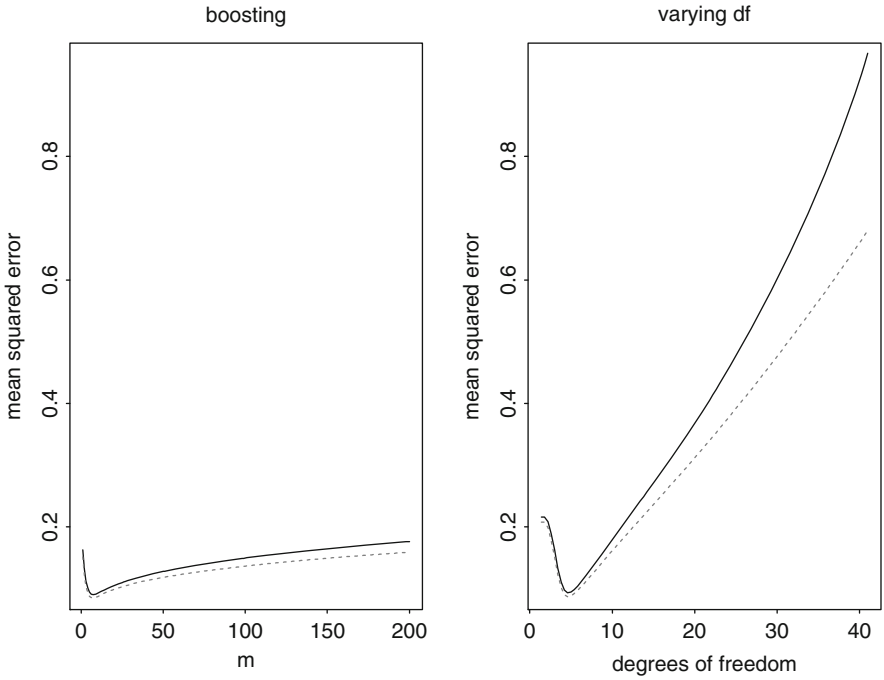


Fig. 33.4 Mean squared error $\mathbb{E}[(g(X) - \hat{g}(X))^2]$ for new predictor X (*solid line*) and $n^{-1}\sum_{i=1}^{n}\mathbb{E}[(\hat{F}_m(X_i) - g(X_i))^2]$ (*dotted line*) from 100 simulations of a nonparametric regression model with smooth regression function and Unif.$[-1/2, 1/2]$-distributed design points. Sample size is $n = 100$. Left: L_2Boost with cubic smoothing spline having $df = 3$, as a function of boosting iterations m. Right: Cubic smoothing spline for various degrees of freedom (*various amount of smoothing*)

Alternative Formulation in Function Space

In Steps 2 and 3 of the generic FGD algorithm, we associated with $U_1, \ldots, U_n$ a negative gradient vector. A reason for this can be seen from the following formulation in function space.

Consider the empirical risk functional $C(f) = n^{-1}\sum_{i=1}^{n}\rho(f(X_i), Y_i)$ and the inner product $(f, g)_n = n^{-1}\sum_{i=1}^{n} f(X_i)g(X_i)$. We can then calculate the negative (functional) Gâteaux derivative $dC(\cdot)$ of the functional $C(\cdot)$,

$$-dC(f)(x) = -\frac{\partial}{\partial\alpha}C(f + \alpha\delta_x)|_{\alpha=0}, \quad f : \mathbb{R}^p \to \mathbb{R},\ x \in \mathbb{R}^p,$$

where δ_x denotes the delta- (or indicator-) function at $x \in \mathbb{R}^p$. In particular, when evaluating the derivative $-dC$ at $\hat{f}^{[m-1]}$ and X_i, we get

$$-dC(\hat{f}^{[m-1]})(X_i) = n^{-1}U_i, \tag{33.14}$$

with $U_1, ..., U_n$ exactly as in steps 2 and 3 of the generic FGD algorithm. Thus, the negative gradient vector $U_1, \ldots, U_n$ can be interpreted as a functional (Gâteaux) derivative evaluated at the data points.

L_2Boosting

Boosting using the squared error loss, L_2Boost, has a simple structure: the negative gradient in Step 2 is the classical residual vector and the line search in Step 3 is trivial when using a base procedure which does least squares fitting.

L_2Boosting Algorithm

Step 1 (initialization). As in Step 1 of generic functional gradient descent.

Step 2. Compute residuals $U_i = Y_i - \hat{F}_m(X_i)$ $(i = 1, \ldots, n)$ and fit the real-valued base procedure to the current residuals (typically by (penalized) least squares) as in Step 2 of the generic functional gradient descent; the fit is denoted by $\hat{g}_{m+1}(\cdot)$. Update

$$\hat{F}_{m+1}(\cdot) = \hat{F}_m(\cdot) + \hat{g}_{m+1}(\cdot).$$

We remark here that, assuming the base procedure does some (potentially penalized) least squares fitting of the residuals, the line search in Step 3 of the generic algorithm becomes trivial with $\hat{s}_{m+1} = 1$.

Step 3 (iteration). Increase iteration index m by one and repeat Step 2 until a stopping iteration M is achieved.

The estimate $\hat{F}_M(\cdot)$ is an estimator of the regression function $\mathbb{E}[Y|X = \cdot]$. L_2Boosting is nothing else than repeated least squares fitting of residuals (cf. Bühlmann and Yu 2003; Friedman 2001). With $m = 2$ (one boosting step), it has already been proposed by Tukey (Tukey 1977) under the name "twicing". In the non-stochastic context, the L_2Boosting algorithm is known as "Matching Pursuit" (Mallat and Zhang 1993) which is popular in signal processing for fitting overcomplete dictionaries.

LogitBoost

Boosting using the log-likelihood loss for binary classification (and more generally for multi-class problems) is known as LogitBoost (Friedman et al. 2000).

LogitBoost uses some Newton-stepping with the Hessian, rather than the line search in Step 3 of the generic boosting algorithm:

LogitBoost Algorithm

Step 1 (initialization). Start with conditional probability estimates $\hat{p}_1(X_i) = 1/2$ $(i = 1, \ldots, n)$ (for $\mathbb{P}[Y = 1|X = X_i]$). Set $m = 1$.

Step 2. Compute the pseudo-response (negative gradient)

$$U_i = \frac{Y_i - \hat{p}_m(X_i)}{\hat{p}_m(X_i)(1 - \hat{p}_m(X_i))},$$

and the weights

$$w_i = \hat{p}_m(X_i)(1 - \hat{p}_m(X_i)).$$

Fit the real-valued base procedure to the current pseudo-response U_i $(i = 1, \ldots, n)$ by weighted least squares, using the current weights w_i $(i = 1, \ldots n)$; the fit is denoted by $\hat{g}_{m+1}(\cdot)$. Update

$$\hat{F}_{m+1}(\cdot) = \hat{F}_m(\cdot) + 0.5 \cdot \hat{g}_{m+1}(\cdot)$$

and

$$\hat{p}_{m+1}(X_i) = \frac{\exp(\hat{F}_{m+1}(X_i))}{\exp(\hat{F}_{m+1}(X_i)) + \exp(-\hat{F}_{m+1}(X_i))}.$$

Step 3 (iteration). Increase iteration index m by one and repeat Step 2 until a stopping iteration M is achieved.

The estimate $\hat{F}_M(\cdot)$ is an estimator for half of the log-odds ratio $0.5 \cdot \text{logit}(\mathbb{P}[Y = 1|X = \cdot]$ (see Table 33.1). Thus, a classifier (under equal misclassification loss for the labels $Y = 0$ and $Y = 1$) is

$$\text{sign}(\hat{F}_M(\cdot)),$$

and an estimate for the conditional probability $\mathbb{P}[Y = 1|X = \cdot]$ is

$$\hat{p}_M(\cdot) = \frac{\exp(\hat{F}_M(\cdot))}{\exp(\hat{F}_M(\cdot)) + \exp(-\hat{F}_M(\cdot))}.$$

A requirement for LogitBoost is that the base procedure has the option to be fitted by *weighted* least squares.

Multi-Class Problems

The LogitBoost algorithm described above can be modified for multi-class problems where the response variable takes values in a finite set $\{0, 1, \ldots, J-1\}$ with $J > 2$ by using the multinomial log-likelihood loss (Friedman et al. 2000). But sometimes it can be advantageous to run instead a binary classifier (e.g. with boosting) for many binary problems. The most common approach is to code for J binary problems where the jth problem assigns the response

$$Y^{(j)} = \begin{cases} 1, & \text{if } Y = j, \\ 0, & \text{if } Y \neq j. \end{cases}$$

i.e. the so-called "one versus all" approach. For example, if single class-label can be distinguished well from all others, the "one versus all" approach seems adequate: empirically, this has been reported for classifying tumor types based on microarray gene expressions when using a LogitBoost algorithm (Dettling and Bühlmann 2003).

Other codings of a multi-class into into multiple binary problems are discussed in Allwein et al. (2001).

33.4.3 Poisson Regression

For count data with $Y \in \{0, 1, 2, \ldots\}$, we can use Poisson regression: we assume that $Y|X = x$ has a Poisson$(\lambda(x))$ distribution and the goal is to estimate the function $g(x) = \log(\lambda(x))$. The negative log-likelihood yields then the loss function

$$\rho(y, g) = -yg + \exp(g), \quad g = \log(\lambda),$$

which can be used in the functional gradient descent algorithm in Sect. 33.4.2.

33.4.4 Small Step Size

It is often better to use small step sizes instead of using the full line search step-length $\hat{s}_{m+1}$ from Step 3 in the generic boosting algorithm (or $\hat{s}_{m+1} \equiv 1$ for L_2Boost or $\hat{s}_{m+1} \equiv 0.5$ for LogitBoost). We advocate here to use the step-size

$$\nu \hat{s}_{m+1}, \ 0 < \nu \leq 1,$$

where ν is constant during boosting iterations and small, e.g. $\nu = 0.1$. The parameter ν can be seen as a simple shrinkage parameter, where we use the shrunken $\nu \hat{g}_{m+1}(\cdot)$ instead of the unshrunken $\hat{g}_{m+1}(\cdot)$. Small step-sizes (or shrinkage) make the boosting algorithm slower and require a larger number M of iterations. However, the computational slow-down often turns out to be advantageous for better out-of-sample prediction performance, cf. Friedman (2001), Bühlmann and Yu (2003). There are also some theoretical reasons to use boosting with ν (infinitesimally) small (Efron et al. 2004).

33.4.5 The Bias-Variance Trade-Off for L_2Boosting

We discuss here the behavior of boosting in terms of model-complexity and estimation error when the number of iterations increase. This is best understood in the framework of squared error loss and L_2Boosting.

We represent the base procedure as an operator

$$\mathcal{S} : \mathbb{R}^n \to \mathbb{R}^n, \ (U_1, \ldots, U_n)^T \mapsto (\hat{U}_1, \ldots, \hat{U}_n)^T$$

which maps a (pseudo-)response vector $(U_1, \ldots, U_n)^T$ to its fitted values; the predictor variables X are absorbed here into the operator notation. That is,

$$\mathcal{S}(U_1, \ldots, U_n)^T = (\hat{g}(X_1), \ldots, \hat{g}(X_n))^T,$$

where $\hat{g}(\cdot) = \hat{g}_{X,U}(\cdot)$ is the estimate from the base procedure based on data (X_i, U_i), $i = 1, \ldots, n$. Then, the boosting operator in iteration m equals

$$\mathcal{B}_m = I - (I - \mathcal{S})^m$$

and the fitted values of boosting after m iterations are

$$\mathcal{B}_m Y = \mathbf{Y} - (I - \mathcal{S})^m \mathbf{Y}, \ \mathbf{Y} = (Y_1, \ldots, Y_n)^T.$$

Heuristically, if the base procedure satisfies $\|I - \mathcal{S}\| < 1$ for a suitable norm, i.e. has a "learning capacity" such that the residual vector is shorter than the input-response vector, we see that $\mathcal{B}_m$ converges to the identity I as $m \to \infty$, and $\mathcal{B}_m \mathbf{Y}$ converges to the fully saturated model $\mathbf{Y}$ as $m \to \infty$, interpolating the response data exactly. Thus, we have to stop the boosting algorithm at some suitable iteration number $m = M$, and we see that a bias-variance trade-off is involved when varying the iteration number M.

33.4.6 L_2Boosting with Smoothing Spline Base Procedure for One-Dimensional Curve Estimation

The case where the base procedure is a smoothing spline for a one-dimensional predictor $X \in \mathbb{R}^1$ is instructive, although being only a toy example within the range of potential applications of boosting algorithms.

In our notation from above, $\mathcal{S}$ denotes a smoothing spline operator which is the solution ($\mathcal{S}\mathbf{Y} = g(X_1), \ldots, f(X_n)$) of the following optimization problem (cf. Wahba 1990)

$$\text{argmin}_g n^{-1} \sum_{i=1}^{n} (Y_i - g(X_i))^2 + \lambda \int g''(x)^2 dx.$$

The smoothing parameter λ controls the bias-variance trade-off, and tuning the smoothing spline estimator usually boils down to estimating a good value of λ. Alternatively, the L_2Boosting approach for curve-estimation with a smoothing spline base procedure is as follows.

Choosing the Base Procedure

Within the class of smoothing spline base procedures, we choose a spline by fixing a smoothing parameter λ. This should be done such that the base procedure has low variance but potentially high bias: for example, we may choose λ such that the degrees of freedom $df = \text{trace}(\mathcal{S})$ is low, e.g. $df = 2.5$. Although the base procedure has typically high bias, we will reduce it by pursuing suitably many boosting iterations. Choosing the df is not really a tuning parameter: we only have to make sure that df is small enough, so that the initial estimate (or first few boosting estimates) are not already overfitting. This is easy to achieve in practice and a theoretical characterization is described in Bühlmann and Yu (2003)).

Related aspects of choosing the base procedure are described in Sects. 33.4.7 and 33.4.9. The general "principle" is to choose a base procedure which has low variance and having the property that when taking linear combinations thereof, we obtain a model-class which is rich enough for the application at hand.

MSE Trace and Stopping

As boosting iterations proceed, the bias of the estimator will go down and the variance will increase. However, this bias-variance exhibits a very different behavior as when classically varying the smoothing parameter (the parameter λ).

It can be shown that the variance increases with exponentially small increments of the order $\exp(-Cm)$, $C > 0$, while the bias decays quickly: the optimal

mean squared error for the best boosting iteration m is (essentially) the same as for the optimally selected tuning parameter λ (Bühlmann and Yu 2003), but the trace of the mean squared error is very different, see Fig. 33.4. The L_2Boosting method is much less sensitive to overfitting and hence often easier to tune. The mentioned insensitivity about overfitting also applies to higher-dimensional problems, implying potential advantages about tuning.

Asymptotic Optimality

Such L_2Boosting with smoothing splines achieves the asymptotically optimal minimax MSE rates, and the method can even adapt to higher order smoothness of the true underlying function, without knowledge of the true degree of smoothness (Bühlmann and Yu 2003).

L_2Boosting Using Kernel Estimators

As pointed out above, L_2Boosting of smoothing splines can achieve faster mean squared error convergence rates than the classical $O(n^{-4/5})$, assuming that the true underlying function is sufficiently smooth. We illustrate here a related phenomenon with kernel estimators.

We consider fixed, univariate design points $x_i = i/n$ $(i = 1, \ldots, n)$ and the Nadaraya-Watson kernel estimator for the nonparametric regression function $\mathbb{E}[Y|X = x]$:

$$\hat{g}(x; h) = (nh)^{-1} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) Y_i = n^{-1} \sum_{i=1}^{n} K_h(x - x_i) Y_i,$$

where $h > 0$ is the bandwidth, $K(\cdot)$ a kernel in the form of a probability density which is symmetric around zero and $K_h(x) = h^{-1} K(x/h)$. It is straightforward to derive the form of L_2Boosting using $m = 2$ iterations (with $\hat{f}^{[0]} \equiv 0$ and $\nu = 1$), i.e., twicing Tukey (1977), with the Nadaraya-Watson kernel estimator:

$$\hat{f}^{[2]}(x) = (nh)^{-1} \sum_{i=1}^{n} K_h^{\mathrm{tw}}(x - x_i) Y_i, \; K_h^{\mathrm{tw}}(u) = 2K_h(u) - K_h * K_h(u),$$

where $K_h * K_h(u) = n^{-1} \sum_{r=1}^{n} K_h(u - x_r) K_h(x_r)$. For fixed design points $x_i = i/n$, the kernel $K_h^{\mathrm{tw}}(\cdot)$ is asymptotically equivalent to a higher-order kernel (which can take negative values) yielding a squared bias term of order $O(h^8)$, assuming that the true regression function is four times continuously differentiable. Thus, twicing or L_2Boosting with $m = 2$ iterations amounts to be a Nadaraya-Watson kernel estimator with a higher-order kernel. This explains from another angle why boosting

is able to improve the mean squared error rate of the base procedure. More details including also non-equispaced designs are given in DiMarzio and Taylor (2008).

33.4.7 L_2Boosting for Additive and Interaction Regression Models

In Sect. 33.4.5, we already pointed out that L_2Boosting yields another way of regularization by seeking for a compromise between bias and variance. This regularization turns out to be particularly powerful in the context with many predictor variables.

Additive Modeling

Consider the component-wise smoothing spline which is defined as a smoothing spline with *one selected* predictor variable $X^{(\hat{\iota})}$ ($\hat{\iota} \in \{1, \ldots, d\}$), where

$$\hat{\iota} = \text{argmin}_\iota \sum_{i=1}^{n} (Y_i - \hat{g}_\iota(X_i^{(\iota)}))^2,$$

and $\hat{g}_\iota$ are smoothing splines with single predictors $X^{(j)}$, all having the same low degrees of freedom df, e.g. $df = 2.5$.

L_2Boost with component-wise smoothing splines yields an additive model, since in every boosting iteration, a function of one selected predictor variable is linearly added to the current fit and hence, we can always rearrange the summands to represent the boosting estimator as an additive function in the original variables, $\sum_{j=1}^{d} \hat{m}_j(x_j)$, $x \in \mathbb{R}^d$. The estimated functions $\hat{m}_j(\cdot)$ are fitted in a stage-wise fashion and they are different from the backfitting estimates in additive models (cf. Hastie and Tibshirani 1990). Boosting has much greater flexibility to add complexity, in a stage-wise fashion: in particular, boosting does variable selection, since some of the predictors will never be chosen, and it assigns variable amount of degrees of freedom to the selected components (or function estimates); the degrees of freedom are defined below. An illustration of this interesting way to fit additive regression models with high-dimensional predictors is given in Figs. 33.5 and 33.6 (actually, a penalized version of L_2Boosting, as described below, is shown).

When using regression stumps (decision trees having two terminal nodes) as the base procedure, we also get an additive model fit (by the same argument as with component-wise smoothing splines). If the additive terms $m_j(\cdot)$ are smooth functions of the predictor variables, the component-wise smoothing spline is often a better base procedure than stumps (Bühlmann and Yu 2003). For the purpose of classification, e.g. with LogitBoost, stumps often seem to do a decent job; also, if

the predictor variables are non-continuous, component-wise smoothing splines are often inadequate.

Finally, if the number d of predictors is "reasonable" in relation to sample size n, boosting techniques are not necessarily better than more classical estimation methods (Bühlmann and Yu 2003). It seems that boosting has most potential when the predictor dimension is very high (Bühlmann and Yu 2003). Presumably, more classical methods become then very difficult to tune while boosting seems to produce a set of solutions (for every boosting iteration another solution) whose best member, chosen e.g. via cross-validation, has often very good predictive performance. A reason for the efficiency of the trace of boosting solutions is given in Sect. 33.4.10.

Degrees of Freedom and AIC_c-Stopping Estimates

For component-wise base procedures, which pick one or also a pair of variables at the time, all the component-wise fitting operators are involved: for simplicity, we focus on additive modeling with component-wise fitting operators $\mathcal{S}_j,\ j = 1, \ldots, d$, e.g. the component-wise smoothing spline.

The boosting operator, when using the step size $0 < \nu \leq 1$, is then of the form

$$\mathcal{B}_m = I - (I - \nu \mathcal{S}_{\hat{\imath}_1})(I - \nu \mathcal{S}_{\hat{\imath}_2}) \ldots (I - \nu \mathcal{S}_{\hat{\imath}_m}),$$

where $\hat{\imath}_i \in \{1, \ldots, d\}$ denotes the component which is picked in the component-wise smoothing spline in the ith boosting iteration.

If the $\mathcal{S}_j$'s are all linear operators, and ignoring the effect of selecting the components, it is reasonable to define the degrees of boosting as

$$df(\mathcal{B}_m) = \text{trace}(\mathcal{B}_m).$$

We can represent

$$\mathcal{B}_m = \sum_{j=1}^{d} M_j,$$

where $M_j = M_{j,m}$ is the linear operator which yields the fitted values for the jth additive term, e.g. $M_j \mathbf{Y} = (\hat{m}_j(X_1), \ldots, \hat{m}_j(X_n))^T$. Note that the M_j's can be easily computed in an iterative way by up-dating in the ith boosting iteration as follows:

$$M_{\hat{\imath}_i,new} \leftarrow M_{\hat{\imath}_i,old} + \nu \mathcal{S}_{\hat{\imath}_i}(I - \mathcal{B}_{i-1})$$

and all other $M_j,\ j \neq \hat{\imath}_i$ do not change. Thus, we have a decomposition of the total degrees of freedom into the d additive terms:

$$df(\mathcal{B}_m) = \sum_{j=1}^{d} df_{j,m},$$
$$df_{j,m} = \text{trace}(M_j).$$

The individual degrees of freedom $df_{j,m}$ are a useful measure to quantify the complexity of the jth additive function estimate $\hat{m}_j(\cdot)$ in boosting iteration m. Note that $df_{j,m}$ will increase very sub-linearly as a function of boosting iterations m, see also Fig. 33.4.

Having some degrees of freedom at hand, we can now use the AIC, or some corrected version thereof, to define a stopping rule of boosting without doing some sort of cross-validation: the corrected AIC statistic (Hurvich et al. 1998) for boosting in the mth iteration is

$$AIC_c = \log(\hat{\sigma}^2) + \frac{1 + \text{trace}(\mathcal{B}_m)/n}{1 - (\text{trace}(\mathcal{B}_m) + 2)/n}, \tag{33.15}$$

$$\hat{\sigma}^2 = n^{-1} \sum_{i=1}^{n} (Y_i - (\mathcal{B}_m \mathbf{Y})_i)^2. \tag{33.16}$$

Alternatively, we could use generalized cross-validation (cf. Hastie et al. 2001), which involves degrees of freedom. This would exhibit the same computational advantage, as AIC_c, over cross-validation: instead of running boosting multiple times, AIC_c and generalized cross-validation need only one run of boosting (over a suitable number of iterations).

Penalized L_2Boosting

When viewing the AIC_c criterion in (33.15) as a reasonable estimate of the true underlying mean squared error (ignoring uninteresting constants), we may attempt to construct a boosting algorithm which reduces in every step the AIC_c statistic (an estimate of the out-sample MSE) most, instead of maximally reducing the in-sample residual sum of squares.

We describe here penalized boosting for additive model fitting using individual smoothing splines:

Penalized L_2Boost with Additive Smoothing Splines

Step 1 (initialization). As in Step 1 of L_2Boost by fitting a component-wise smoothing spline.
Step 2. Compute residuals $U_i = Y_i - \hat{F}_m(X_i)$ $(i = 1, \ldots, n)$. Choose the individual smoothing spline which reduces AIC_c most: denote the selected component by $\hat{\imath}_{m+1}$ and the fitted function, using the selected component $\hat{\imath}_{m+1}$ by $\hat{g}_{m+1}(\cdot)$.

Update

$$\hat{F}_{m+1}(\cdot) = \hat{F}_m(\cdot) + \nu \hat{g}_{m+1}(\cdot).$$

for some step size $0 < \nu \leq 1$.

Step 3 (iteration). Increase iteration index m by one and repeat Step 2 until the AIC_c criterion in (33.15) cannot be improved anymore.

This algorithm cannot be written in terms of fitting a base procedure multiple times since selecting the component $\hat{t}$ in Step 2 not only depends on the residuals $U_1, \ldots, U_n$, but also on the degrees of boosting, i.e. trace($\mathcal{B}_{m+1}$); the latter is a complicated, although linear function, of the boosting iterations $m' \in \{1, 2, \ldots, m\}$. Penalized L_2Boost yields more sparse solutions than the corresponding L_2Boost (with component-wise smoothing splines as corresponding base procedure). The reason is that $df_{j,m}$ increases only little in iteration $m + 1$, if the jth selected predictor variables has already been selected many times in previous iterations; this is directly connected to the slow increase in variance and overfitting as exemplified in Fig. 33.4.

An illustration of penalized L_2Boosting with individual smoothing splines is shown in Figs. 33.5 and 33.6, based on simulated data. The simulation model is

$$\begin{aligned} &X_1, \ldots, X_n \text{ i.i.d. } \sim \text{ Unif.}[0,1]^{100}, \\ &Y_i = \sum_{j=1}^{10} m_j(X^{(j)}) + \varepsilon_i \ (i = 1, \ldots, n), \\ &\varepsilon_1, \ldots, \varepsilon_n \text{ i.i.d. } \sim \mathcal{N}(0, 0.5), \end{aligned} \tag{33.17}$$

where the m_j's are smooth curves having varying curve complexities, as illustrated in Fig. 33.6. Sample size is $n = 200$ which is small in comparison to $d = 100$ (but the effective number of predictors is only 10).

In terms of prediction performance, penalized L_2Boosting is not always better than L_2Boosting; Fig. 33.7 illustrates an advantage of penalized L_2Boosting. But penalized L_2Boosting is always sparser (or at least not less sparse) than the corresponding L_2Boosting.

Obviously, penalized L_2Boosting can be used for other than additive smoothing spline model fitting. The modifications are straightforward as long as the individual base procedures are linear operators.

Interaction Modeling

L_2Boosting for additive modeling can be easily extended to interaction modeling (having low degree of interaction). Among the most prominent case is the second order interaction model $\sum_{j,k=1}^{d} \hat{m}_{j,k}(x_j, x_k)$, where $\hat{m}_{j,k} : \mathbb{R}^2 \to \mathbb{R}$.

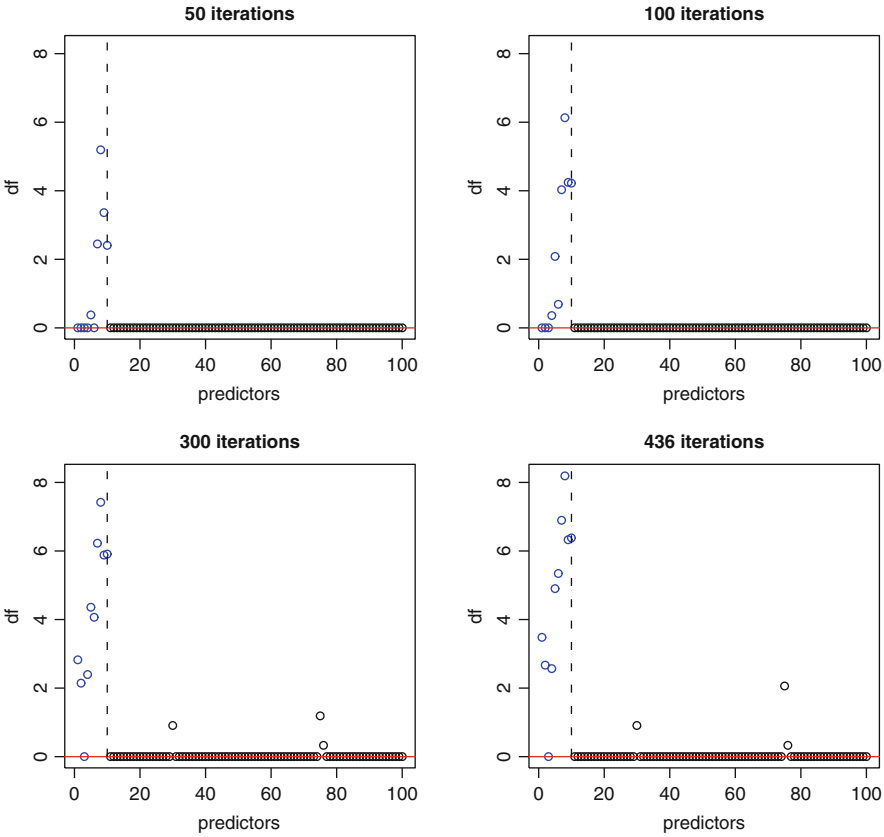


Fig. 33.5 Degrees of freedom (df) in additive model fitting for all 100 predictor variables (*from model* (33.17)) during the process of penalized L_2Boosting with individual smoothing splines (having $df = \text{trace}(\mathcal{S}_j) = 2.5$ for each spline). The first ten predictor variables (*separated by the dashed line*) are effective. The result is based on one realization from model (33.17) with sample size $n = 200$. The plot on the lower right corresponds to the estimated optimal number of boosting iterations using the AIC_c criterion in (33.15). Only three non-effective predictors have been selected (and assigned small amount of df), and one effective predictor has not been selected (but whose true underlying function is close to the zero-line, see Fig. 33.6)

Boosting with a pairwise thin plate spline, which selects the best pair of predictor variables yielding lowest residual sum of squares (when having the same degrees of freedom for every thin plate spline), yields a second-order interaction model. We demonstrate in Fig. 33.7 the effectiveness of this procedure in comparison with the second-order MARS fit (Friedman 1991). The underlying model is the Friedman #1 model:

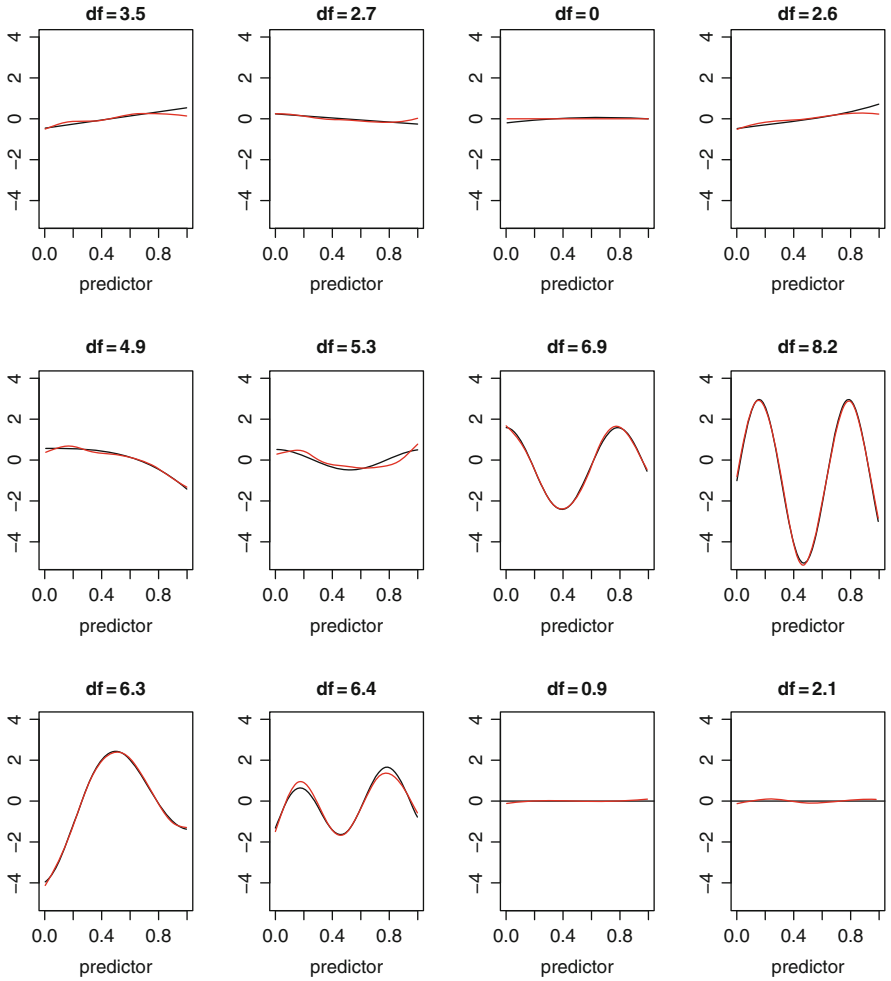


Fig. 33.6 True underlying additive regression curves (*black*) and estimates (*red*) from penalized L_2Boosting as described in Fig. 33.5 (using *436 iterations*, estimated from (33.15)). The last two plots correspond to non-effective predictors (the true functions are the *zero-line*), where L_2Boosting assigned most df among non-effective predictors

$$\begin{aligned} &X_1, \ldots, X_n \text{ i.i.d. } \sim \text{ Unif.}([0,1]^d), \ d \in \{10, 20\}, \\ &Y_i = 10 \sin(\pi X^{(1)} X^{(2)}) + 20(X^{(3)} - 0.5)^2 + 10X^{(4)} + 5X^{(5)} \\ &\quad + \varepsilon_i \ (i = 1, \ldots, n), \\ &\varepsilon_1, \ldots, \varepsilon_n \text{ i.i.d } \sim \mathcal{N}(0, 1). \end{aligned} \tag{33.18}$$

The sample size is chosen as $n = 50$ which is small in comparison to $d = 20$.

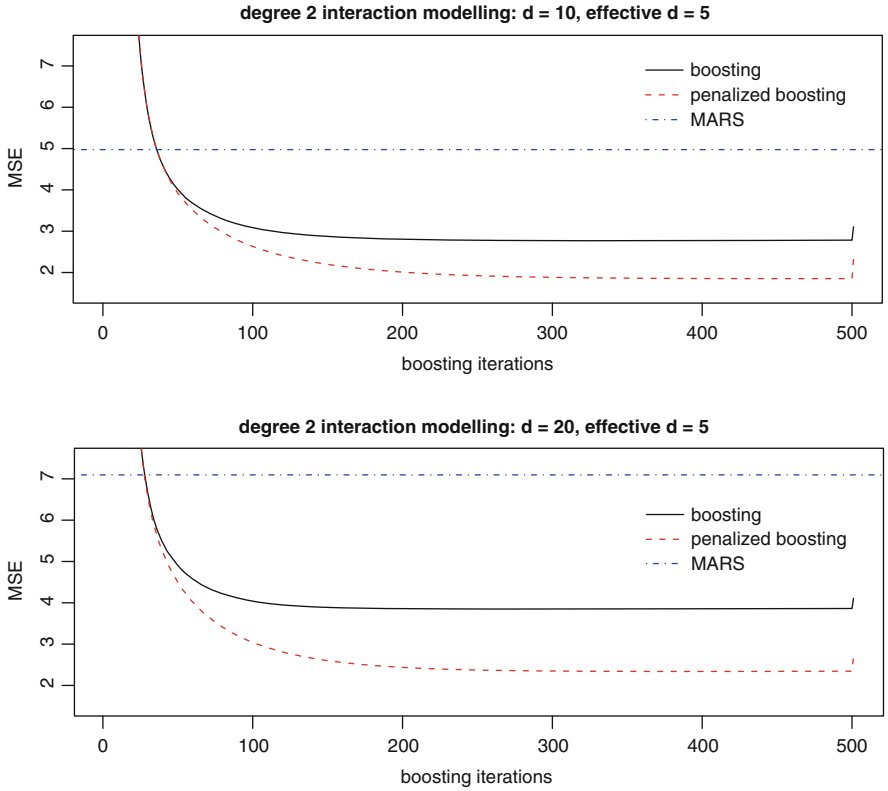


Fig. 33.7 Mean squared errors for L_2Boost with pairwise thin-plate splines (of *two predictor variables*, having $df = \text{trace}(\mathcal{S}_{j,k}) = 2.5$) (*black*), its penalized version (*red*) and MARS restricted to the (*correct*) second order interactions (*blue*). The point with abscissa x=501 for the boosting methods corresponds to the performance when estimating the number of iterations using (33.15). Based on simulated data from model (33.18) with $n = 50$

In high-dimensional settings, it seems that such interaction L_2Boosting is clearly better than the more classical MARS fit, while both of them share the same superb simplicity of interpretation.

33.4.8 Linear Modeling

L_2Boosting turns out to be also very useful for linear models, in particular when there are many predictor variables:

$$\mathbf{Y} = \mathbf{X}\beta + \varepsilon$$

where we use the well-known matrix-based notation. An attractive base procedure is component-wise linear least squares regression, using the one selected predictor variables which reduces residual sum of squares most.

This method does variable selection, since some of the predictors will never be picked during boosting iterations; and it assigns variable amount of degrees of freedom (or shrinkage), as discussed for additive models above. Recent theory shows that this method is consistent for very high-dimensional problems where the number of predictors $d = d_n$ is allowed to grow like $\exp(Cn)$ $(C > 0)$, but the true underlying regression coefficients are sparse in terms of their ℓ_1-norm, i.e. $\sup_n \|\beta\|_1 = \sup_n \sum_{j=1}^{d_n} |\beta_j| < \infty$, where β is the vector of regression coefficients (Bühlmann 2006).

33.4.9 Boosting Trees

The most popular base procedures for boosting, at least in the machine learning community, are trees. This may be adequate for classification, but when it comes to regression, or also estimation of conditional probabilities $\mathbf{P}[Y = 1|X = x]$ in classification, smoother base procedures often perform better if the underlying regression or probability curve is a smooth function of continuous predictor variables (Bühlmann and Yu 2003).

Even when using trees, the question remains about the size of the tree. A guiding principle is as follows: take the smallest trees, i.e. trees with the smallest number k of terminal nodes, such that the class of linear combinations of k-node trees is sufficiently rich for the phenomenon to be modeled; of course, there is also here a trade-off between sample size and the complexity of the function class.

For example, when taking stumps with $k = 2$, the set of linear combinations of stumps is dense in (or "yields" the) set of additive functions (Breiman 2004). In Friedman et al. (2000), this is demonstrated from a more practical point of view. When taking trees with three terminal nodes ($k = 3$), the set of linear combinations of 3-node trees yields all second-order interaction functions. Thus, when aiming for consistent estimation of the full regression (or conditional class-probability) function, we should choose trees with $k = d + 1$ terminal nodes (in practice only if the sample size is "sufficiently large" in relation to d), (cf. Breiman 2004).

Consistency of the AdaBoost algorithm is proved in Jiang (2004), for example when using trees having $d + 1$ terminal nodes. More refined results are given in Mannor et al. (2002), Zhang and Yu (2005) for modified boosting procedures with more general loss functions.

Interpretation

The main disadvantage from a statistical perspective is the lack of interpretation when boosting trees. This is in sharp contrast to boosting for linear, additive or interaction modeling. An approach to enhance interpretation is described in Friedman (2001).

33.4.10 Boosting and ℓ_1-Penalized Methods (Lasso)

Another method which does variable selection and variable amount of shrinkage is basis pursuit (Chen et al. 1999) or Lasso (Tibshirani 1996) which employs an ℓ_1-penalty for the coefficients in the log-likelihood.

There is an intriguing connection between L_2Boosting with componentwise linear least squares and the Lasso, as pointed out in Hastie et al. (2001). The connection has been rigorously established in Efron et al. (2004): they consider a version of L_2Boosting, called forward stagewise linear regression (FSLR), and they show that FSLR with infinitesimally small step-sizes (i.e., the value ν in Sect. 33.4.4) produces a set of solutions which is equivalent (as step-sizes tend to zero) to the set of Lasso solutions when varying the regularization parameter λ in the Lasso

$$\hat{\beta}(\lambda) = \mathrm{argmin}_\beta \Big(\|\mathbf{Y} - \mathbf{X}\beta\|_2^2/n + \lambda \|\beta\|_1 \Big).$$

The equivalence only holds though if the design matrix $\mathbf{X}$ satisfies a very restrictive "positive cone condition" (Efron et al. 2004).

Despite the fact that L_2Boosting and Lasso are not equivalent methods in general, it may be useful to interpret boosting as being "related" to ℓ^1-penalty based methods. This is particularly interesting when looking at the problem of high-dimensional variable selection. For the Lasso, sufficient and necessary conditions on the design $\mathbf{X}$ have been derived for consistent variable selection (Meinshausen and Bühlmann 2006; Zhao and Yu 2006). In view of these rather restrictive design conditions, the adaptive Lasso has been proposed (Zou 2006). Related to the adaptive Lasso, Twin boosting (Bühlmann and Hothorn 2010) is a very general method, like the generic boosting algorithm in Sect. 33.4.2 which has better variable selection properties than boosting. Similarly, when looking at estimation error in terms of $\|\hat{\beta} - \beta\|_1$ or $\|\hat{\beta} - \beta\|_2$, many refined results have been worked out for the Lasso (cf. Bickel et al. 2009).

33.4.11 Aggregation

In the machine learning community, there has been a substantial focus on consistent estimation in the convex hull of function classes (cf. Bartlett 2003; Bartlett et al. 2006; Lugosi and Vayatis 2004) which is a special case of aggregation (cf. Tsybakov 2004). For example, one may want to estimate a regression or probability function which can be written as

$$\sum_{k=1}^{\infty} w_k g_k(\cdot), \ w_k \geq 0, \ \sum_{k=1}^{\infty} w_k = 1,$$

where the $g_k(\cdot)$'s belong to a function class such as stumps or trees with a fixed number of terminal nodes. The quantity above is a convex combination of individual functions, in contrast to boosting which pursues linear combination of individual functions. By scaling, which is necessary in practice and theory (cf. Lugosi and Vayatis 2004), one can actually look at this as a linear combination of functions whose coefficients satisfy $\sum_k w_k = \lambda$. This then represents an ℓ_1-constraint as in Lasso, a relation which we have already outlined above.

33.4.12 Other References

Boosting, or functional gradient descent, has also been proposed for other settings than regression or classification, including survival analysis (Benner 2002), ordinal response problems (Tutz and Hechenbichler 2005), generalized monotonic regression (Leitenstorfer and Tutz 2007), and high-multivariate financial time series (Audrino and Barone-Adesi 2005; Audrino and Bühlmann 2003). More references are provided in Bühlmann and Hothorn (2007).

Random Forests (Breiman 2001) is another, powerful ensemble method which exhibits excellent predictive performance over a wide range of problems. In addition, it assigns variable importance which is of tremendous use for feature/variable selection and ranking features/variables (cf. Strobl et al. 2008). Some theoretical properties are derived in Li and Jeon (2006) and Biau et al. (2008).

Support vector machines (cf. Hastie et al. 2001; Schölkopf and Smola 2002; Vapnik 1998) have become very popular in classification due to their good performance in a variety of data sets, similarly as boosting methods for classification. A connection between boosting and support vector machines has been made in Rosset et al. (2004), suggesting also a modification of support vector machines to more sparse solutions (Zhu et al. 2004).

Acknowledgments: I would like to thank Marcel Dettling for some constructive comments.

References

Allwein, E., Schapire, R., Singer, Y.: Reducing multiclass to binary: a unifying approach for margin classifiers. J. Mach. Learn. Res. **1**, 113–141 (2001)

Amit, Y., Geman, D.: Shape quantization and recognition with randomized trees. Neural Comput. **9**, 1545–1588 (1997)

Audrino F., Barone-Adesi G.: A multivariate FGD technique to improve VaR computation in equity markets. Comput. Manag. Sci. **2**, 87–106 (2005)

Audrino, F., Bühlmann, P.: Volatility estimation with functional gradient descent for very high-dimensional financial time series. J. Comput. Fin, **6**(3), 65–89 (2003)

Bartlett, P.L.: Prediction algorithms: complexity, concentration and convexity. In: Proceedings of the 13th IFAC Symposium on System Identification, pp. 1507–1517 (2003)

Bartlett, P.L., Jordan, M.I., McAuliffe, J.D.: Convexity, classification, and risk bounds. J. Am. Stat. Assoc. **101**, 138–156 (2006)
Bauer, E., Kohavi, R.: An empirical comparison of voting classification algorithms: bagging, boosting and variants. Mach. Learn. **36**, 1545–1588 (1999)
Biau, G., Devroye, L. Lugosi, G.: Consistency of Random Forests and other averaging classifiers. J. Mach. Learn. Res. **9**, 2015–2033 (2008)
Benner, A.: Application of "aggregated classifiers" in survival time studies. In: Härdle, W., Rönz, B. (eds.) In: COMPSTAT 2002 – Proceedings in Computational Statistics – 15th Symposium held in Physika, Heidelberg, Berlin (2002)
Bickel, P., Ritov, Y., Tsybakov, A.: Simultaneous analysis of lasso and dantzig selector. Ann. Stat. **37**, 1705–1732 (2009)
Borra, S., Di Ciaccio, A.: Improving nonparametric regression methods by bagging and boosting. Comput. Stat. Data Anal. **38**, 407–420 (2002)
Breiman, L.: Bagging predictors. Mach. Learn. **24**, 123–140 (1996a)
Breiman, L.: Out-of-bag estimation. Technical Report (1996b); Available from ftp://ftp.stat.berkeley.edu/pub/users/breiman/
Breiman, L.: Arcing classifiers. Ann. Stat. **26**, 801–824 (1998)
Breiman, L.: Prediction games & arcing algorithms. Neu. Comput. **11**, 1493–1517 (1999)
Breiman, L.: Random Forests. Mach. Learn. **45**, 5–32 (2001)
Breiman, L.: Population theory for boosting ensembles. Ann. Stat. **32**, 1–11 (2004)
Bühlmann, P.: Bagging, subagging and bragging for improving some prediction algorithms. In: Akritas, M.G., Politis, D.N. (eds.) In: Recent Advances and Trends in Nonparametric Statistics, Elsevier, Amsterdam (2003)
Bühlmann, P.: Boosting for high-dimensional linear models. Ann. Stat. **34**, 559–583 (2006)
Bühlmann, P., Hothorn, T.: Boosting algorithms: regularization, prediction and model fitting (with discussion). Stat. Sci. **22**, 477–505 (2007)
Bühlmann, P., Hothorn, T.: Twin Boosting: improved feature selection and prediction. Stat. Comput. **20**, 119–138 (2010)
Bühlmann, P., Yu, B: Discussion on Additive logistic regression: a statistical view of boosting (Auths. Friedman, J., Hastie, T., Tibshirani, R.) Ann. Stat. **28**, 377–386 (2000)
Bühlmann, P., Yu, B.: Analyzing bagging. Ann. Stat. **30**, 927–961 (2002)
Bühlmann, P., Yu, B.: Boosting with the L_2loss: regression and classification. J. Am. Stat. Assoc. **98**, 324–339 (2003)
Buja, A., Stuetzle, W.: Observations on bagging. Statistica Sinica **16**, 323–351 (2006)
Bylander, T.: Estimating generalization error on two-class datasets using out-of-bag estimates. Mach. Learn. **48**, 287–297 (2002)
Chen, S.S., Donoho, D.L., Saunders, M.A.: Atomic decomposition by basis pursuit. SIAM J. Sci. Comput. **20**(1), 33–61 (1999)
Chen, S.X., Hall, P.: Effects of bagging and bias correction on estimators defined by estimating equations. Statistica Sinica **13**, 97–109 (2003)
DiMarzio, M., Taylor, C.: On boosting kernel regression. J. Stat. Plann. Infer. **138**, 2483–2498 (2008)
Dettling, M.: BagBoosting for tumor classification with gene expression data. Bioinformatics 20 (18), 3583–3593 (2004).
Dettling, M., Bühlmann, P.: Boosting for tumor classification with gene expression data. Bioinformatics **19**(9), 1061–1069 (2003)
Dudoit, S., Fridlyand, J.: Bagging to improve the accuracy of a clustering procedure. Bioinformatics **19**(9), 1090–1099 (2003)
Efron, B., Tibshirani, R.: The problem of regions. Ann. Stat. **26**, 1687–1718 (1998)
Efron, B., Hastie, T., Johnstone, I., Tibshirani, R.: Least angle regression (with discussion). Ann. Stat. **32**, 407–451 (2004)
Freund, Y.: Boosting a weak learning algorithm by majority. Inform. Comput. **121**, 256–285 (1995)

Freund, Y., Schapire, R.E.: Experiments with a new boosting algorithm. In Machine Learning: Proceedings of 13th International Conference, pp. 148–156. Morgan Kauffman, San Francisco (1996)

Friedman, J.H.: Multivariate adaptive regression splines. Ann. Stat. **19**, 1–141 (1991)

Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Ann. Stat. **29**, 1189–1232 (2001)

Friedman, J.H., Hastie, T., Tibshirani, R.: Additive logistic regression: a statistical view of boosting. Ann. Stat. **28**, 337–407 (2000)

Hastie, T.J., Tibshirani, R.J.: Generalized Additive Models. Chapman & Hall, London (1990)

Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning. Data Mining, Inference and Prediction. Springer, New York (2001)

Hothorn, T., Bühlmann, P., Kneib, T., Schmid M., Hofner, B.: Model-based boosting 2.0. Journal of Machine Learning Research 11, 2109–2113 (2010).

Hurvich, C.M., Simonoff, J.S., Tsai, C.-L.: Smoothing parameter selection in nonparametric regression using an improved Akaike information criterion. J. Roy. Stat. Soc. B **60**, 271–293 (1998)

Jiang, W.: Process consistency for AdaBoost (with discussion). Ann. Stat. **32**, 13–29, (disc. pp. 85–134) (2004)

Leitenstorfer, F., Tutz, G.: Generalized monotonic regression based on B-splines with an application to air pollution data. Biostatistics **8**, 654–673 (2007)

Li, Y., Jeon, Y.: Random Forests and adaptive nearest neighbors. J. Am. Stat. Assoc. **101**, 578–590 (2006)

Lugosi, G., Vayatis, N.: On the Bayes-risk consistency of regularized boosting methods. Ann. Stat. **32**, 30–55 (disc. pp. 85–134) (2004)

Mallat, S., Zhang, Z.: Matching pursuits with time-frequency dictionaries. IEEE Trans. Signal Process. **41**, 3397–3415 (1993)

Mannor, S., Meir, R., Zhang, T.: The consistency of greedy algorithms for classification. Proceedings COLT02, Vol. 2375 of LNAI, pp. 319–333. Springer, Sydney (2002)

Mason, L., Baxter, J., Bartlett, P., Frean, M.: Functional gradient techniques for combining hypotheses. In: Smola, A.J., Bartlett, P.J., Schölkopf, B., Schuurmans, D. (eds.) In: Advances in Large Margin Classifiers MIT Press, Cambridge, MA (2000)

Meinshausen, N., Bühlmann, P.: High-dimensional graphs and variable selection with the Lasso. Ann. Stat. **34**, 1436–1462 (2006)

Meinshausen, N., Bühlmann, P.: Stability selection (with discussion). Journal of the Royal Statistical Society: Series B, 72, 417–473 (2010).

Meinshausen, N., Meier, L., Bühlmann, P.: p-values for high-dimensional regression. J. Am. Stat. Assoc. **104**, 1671–1681 (2009)

Politis, D.N., Romano, J.P., Wolf, M.: Subsampling. Springer, New York (1999)

Ridgeway, G.: Looking for lumps: Boosting and bagging for density estimation. Comput. Stat. Data Anal. **38**(4), 379–392 (2002)

Rosset, S., Zhu, J., Hastie, T.: Boosting as a regularized path to a maximum margin classifier. J. Mach. Learn. Res. **5**, 941–973 (2004)

Schapire, R.E.: The strength of weak learnability. Mach. Learn. **5**, 197–227 (1990)

Schapire, R.E.: The boosting approach to machine learning: an overview. In: Denison, D.D., Hansen, M.H., Holmes, C.C., Mallick, B., Yu, B. (eds.) In: MSRI Workshop on Nonlinear Estimation and Classification. Springer, New York (2002)

Schölkopf, B., Smola, A.J.: Learning with Kernels. MIT Press, Cambridge (2002)

Strobl, C., Boulesteix, A.-L., Kneib, T., Augustin, T., Zeileis, A.: Conditional variable importance for random forests. BMC Bioinformatics **9**(307), 1–11 (2008)

Tibshirani, R.: Regression shrinkage and selection via the lasso. J. Roy. Stat. Soc. B **58**, 267–288 (1996)

Tsybakov, A.: Optimal aggregation of classifiers in statistical learning. Ann. Stat. **32**, 135–166 (2004)

Tukey, J.W.: Exploratory data analysis. Addison-Wesley, Reading, MA (1977)

Tutz, G., Hechenbichler, K.: Aggregating classifiers with ordinal response structure. J. Stat. Comput. Simul. **75**, 391–408 (2005)
Vapnik, V.N.: Statistical Learning Theory. Wiley, New York (1998)
Wahba, G.: Spline Models for Observational Data. Society for Industrial and Applied Mathematics (1990)
Zhang, T., Yu, B.: Boosting with early stopping: convergence and consistency. Ann. Stat. **33**, 1538–1579 (2005)
Zhao, P., Yu, B.: On model selection consistency of Lasso. J. Mac. Learn. Res. **7**, 2541–2563 (2006)
Zhu, J., Rosset, S., Hastie, T., Tibshirani, R.: 1-norm support vector machines. Advances in Neural Information Processing Systems 16: Proceedings of the 2003 Conference, 49–56 (2004)
Zou, H.: The adaptive Lasso and its oracle properties. J. Am. Stat. Assoc. **101**, 1418–1429 (2006)

Part IV
Selected Applications

Chapter 34
Heavy-Tailed Distributions in VaR Calculations

Adam Misiorek and Rafał Weron

34.1 Introduction

Market risks are the prospect of financial losses – or gains – due to unexpected changes in market prices and rates. Evaluating the exposure to such risks is nowadays of primary concern to risk managers in financial and non-financial institutions alike. Since the early 1990s a commonly used market risk estimation methodology has been the Value at Risk (VaR). A VaR measure is the highest possible loss L incurred from holding the current portfolio over a certain period of time at a given confidence level (Alexander 2008; Jorion 2006):

$$\mathbb{P}\left(L > \text{VaR}\right) \leq 1 - c, \tag{34.1}$$

where c is the confidence level, typically $c \geq 95\%$. By convention, $L = -\Delta X(\tau)$, where $\Delta X(\tau)$ is the relative change (return) in portfolio value over the time horizon τ. Hence, large values of L correspond to large losses (or large negative returns).

The VaR figure has two important characteristics: (1) it provides a common consistent measure of risk across different positions and risk factors and (2) it takes into account the correlations or dependencies between different risk factors. Because of its intuitive appeal and simplicity, it is no surprise that within a few years VaR has become the standard risk measure used around the world. However, it has a number deficiencies, among them the non-subadditivity – a sum of VaR's of two

A. Misiorek
Santander Consumer Bank S.A., Wrocław, Poland
e-mail: adam.misiorek@santanderconsumer.pl

R. Weron (✉)
Institute of Organization and Management, Wrocław University of Technology, Wrocław, Poland
e-mail: rafal.weron@gmail.com

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_34,

portfolios can be smaller than the VaR of the combined portfolio. To cope with these shortcomings, Artzner et al. (1999) proposed an alternative measure that satisfies the assumptions of a coherent, i.e. an adequate, risk measure. The Expected Shortfall (ES), also called Expected Tail Loss or Conditional VaR, is the expected value of the losses in excess of VaR, i.e. ES $= \mathbb{E}(L|L > \text{VaR})$. It is interesting to note, that the notion of Expected Shortfall has been familiar to insurance practitioners for decades. It is very similar to the mean excess function which is used to characterize claim size distributions (see Chap. 9 in Cizek et al. 2011).

The essence of the VaR and ES computations is estimation of low quantiles in the portfolio return distributions. Hence, the performance of market risk measurement methods depends on the quality of distributional assumptions on the underlying risk factors. Many of the concepts in theoretical and empirical finance developed over the past decades – including the classical portfolio theory, the Black-Scholes-Merton option pricing model and even the RiskMetrics variance-covariance approach to VaR – rest upon the assumption that asset returns follow a normal distribution. But is this assumption justified by empirical data?

No, it is not! It has been long known that asset returns are not normally distributed. Rather, the empirical observations exhibit excess kurtosis (fat tails). In Fig. 34.1 we plot a ten-year history (January 3, 2000 – December 31, 2009) of the Deutsche Aktienindex (DAX) index, its returns (or log-returns) and the distribution of the returns. The contrast with the Gaussian law is striking. This heavy tailed or leptokurtic character of the distribution of price changes has been repeatedly observed in various markets and may be quantitatively measured by the kurtosis in excess of 3, a value obtained for the normal distribution (Guillaume et al. 1997; Rachev and Mittnik 2000).

Interestingly, the problem of the underestimation of risk by the Gaussian distribution has been dealt with by the regulators in an ad hoc way. The Basle Committee on Banking Supervision (1995) suggested that for the purpose of determining minimum capital reserves financial institutions use a ten day VaR at the $c = 99\%$ confidence level multiplied by a safety factor $s \in [3, 4]$, with the exact value of s depending on the past performance of the model. It has been argued by Stahl (1997) and Danielsson et al. (1998) that the range of the safety factor comes from the heavy-tailed nature of the returns distribution. Indeed, if we assume that the asset returns distribution is symmetric and has finite variance σ^2 then from Chebyshev's inequality we obtain $\mathbb{P}(L \geq \epsilon) \leq \sigma^2/2\epsilon^2$, where L represents the random loss over the specified time horizon. So if we want to calculate the upper bound for a 99% VaR, setting $\sigma^2/2\epsilon^2 = 1\%$ yields $\epsilon = 7.07\sigma$, which in turn implies that $\text{VaR}_{99\%} \leq 7.07\sigma$. However, if we assumed a Gaussian distribution of returns then we would have $\text{VaR}_{99\%} \leq 2.33\sigma$, which is roughly three times lower than the bound obtained for a heavy-tailed, finite variance distribution.

Having said this much about the inadequacy of the Gaussian distribution for financial modeling and risk management we have no other choice but offer some heavy-tailed alternatives. We have to mention, though, that all distributional classes described in this chapter present computational challenge. Large parts of the text are thus devoted to numerical issues. In Sect. 34.2 we deal with the historically earliest

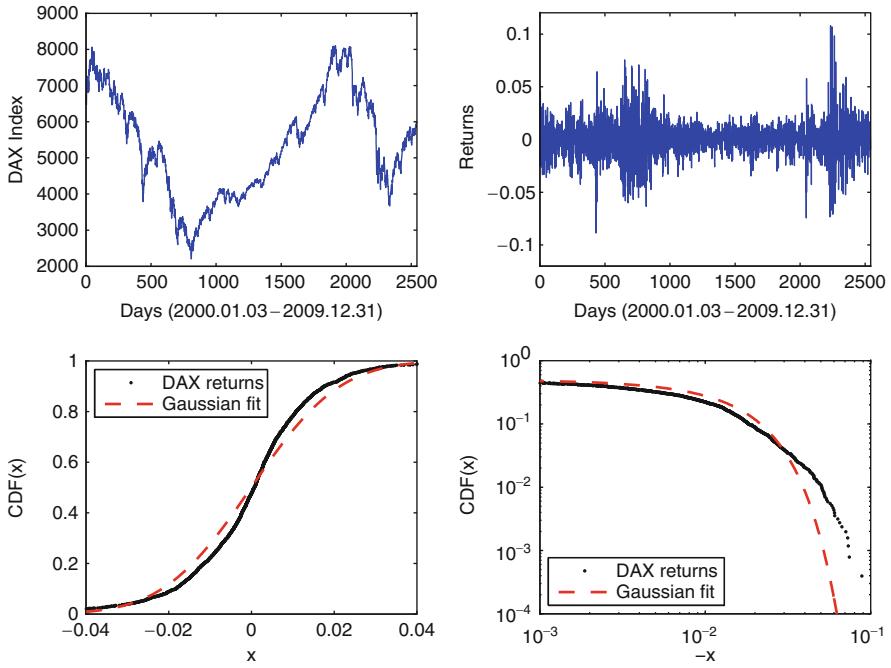


Fig. 34.1 *Top panels*: DAX daily closing values X_t and daily returns $\log(X_{t+1}/X_t)$ from the period January 3, 2000 – December 31, 2009. *Bottom panels*: Gaussian fit to the DAX daily returns empirical cumulative distribution function (CDF). For better exposition of the fit in the central part of the distribution the range is limited to $\pm 4\%$. The right panel is a magnification of the left tail fit on a double logarithmic scale clearly showing the discrepancy between the data and the normal distribution

alternative – the stable laws – and briefly characterize their recent generalizations – the so-called truncated and tempered stable distributions. Further, in Sect. 34.3 we study the class of generalized hyperbolic laws. Finally, in Sect. 34.4 we introduce the notion of copulas and discuss the relation between VaR, asset portfolios and heavy tails.

34.2 Stable Distributions

34.2.1 Definitions and Basic Properties

Since the pioneering work of Louis Bachelier in 1900, financial asset returns have been modeled by the Gaussian distribution. The theoretical rationale for this comes from the Central Limit Theorem (CLT), which states that the sum of a large number

of independent, identically distributed (i.i.d.) variables – say, decisions of investors – from a finite-variance distribution will be (asymptotically) normally distributed. However, empirical evidence indicates that financial asset returns tend to have heavier tails than Gaussian. Possible reasons for the failure of the CLT are infinite-variance distributions of the variables, non-identical distributions of the variables, dependences between the variables or any combination of the three.

The dependence issue is hard to tackle analytically, however, if only the finite variance assumption is dropped we have a readily usable solution. Namely, the generalized version of the CLT states that the limiting distribution of sums of such variables is stable (Nolan 2012). This, together with the fact that stable distributions are leptokurtic and can accommodate fat tails and asymmetry, provides us with a theoretically justified modeling tool. Indeed, as early as in the 1960s stable laws were proposed as an alternative model for asset returns (Mandelbrot 1963).

Stable laws – also called α-stable, stable Paretian or Lévy stable – were introduced by Lévy (1925) during his investigations of the behavior of sums of independent random variables. The name "stable" reflects the fact that a sum of two independent random variables having a stable distribution with the same index α is again stable with index α. Recall, that this invariance property holds also for Gaussian variables. In fact, the Gaussian distribution is stable with $\alpha = 2$.

The stable distribution requires four parameters for complete description. The index of stability $\alpha \in (0, 2]$, also called the tail index, tail exponent or characteristic exponent, determines the rate at which the tails of the distribution taper off, see the left panel in Fig. 34.2. The skewness parameter $\beta \in [-1, 1]$ defines the asymmetry. When $\beta > 0$, the distribution is skewed to the right, i.e. the right tail is thicker, see the right panel in Fig. 34.2. When it is negative, it is skewed to the left. When $\beta = 0$, the distribution is symmetric about the mode (the peak) of the distribution. As α approaches 2, β loses its effect and the distribution approaches the Gaussian

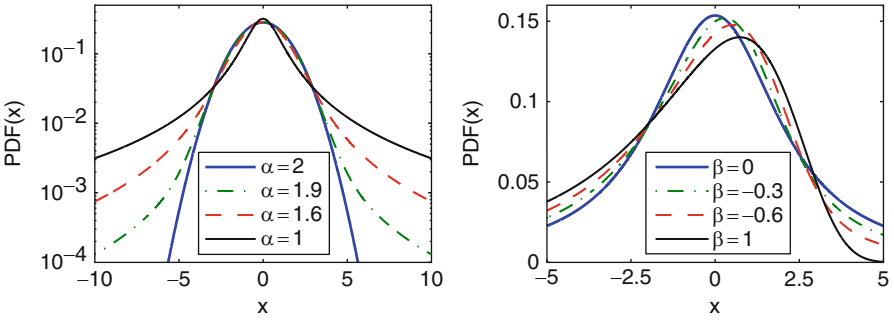


Fig. 34.2 *Left panel*: A semilog plot of symmetric ($\beta = \mu = 0$) stable probability density functions (PDFs) for four different values of α showing the dependence on the tail exponent. The Gaussian ($\alpha = 2$) PDF forms a parabola and is the only stable density with exponential tails. *Right panel*: A plot of stable PDFs for $\alpha = 1.1$ and four different values of β showing the dependence on the skewness parameter

distribution regardless of β. The last two parameters, $\sigma > 0$ and $\mu \in \mathbb{R}$, are the usual scale and location parameters, respectively.

From a practitioner's point of view the crucial drawback of the stable distribution is that, with the exception of three special cases, its probability density function (PDF) and cumulative distribution function (CDF) do not have closed form expressions. These exceptions include the well known Gaussian ($\alpha = 2$) law, whose density is given by:

$$f_G(x) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}, \tag{34.2}$$

and the lesser known Cauchy ($\alpha = 1, \beta = 0$) and Lévy ($\alpha = 0.5, \beta = 1$) laws.

Hence, the stable distribution can be most conveniently described by its characteristic function (CF) – the inverse Fourier transform of the PDF. The most popular parameterization of the characteristic function $\phi(t)$ of $X \sim S_\alpha(\sigma, \beta, \mu)$, i.e. a stable random variable with parameters α, σ, β and μ, is given by (Samorodnitsky and Taqqu 1994; Weron 1996):

$$\log \phi(t) = \begin{cases} -\sigma^\alpha |t|^\alpha \{1 - i\beta \text{sign}(t) \tan\frac{\pi\alpha}{2}\} + i\mu t, & \alpha \neq 1, \\ -\sigma|t|\{1 + i\beta \text{sign}(t)\frac{2}{\pi}\log|t|\} + i\mu t, & \alpha = 1. \end{cases} \tag{34.3}$$

Note, that the traditional scale parameter σ of the Gaussian distribution is not the same as σ in the above representation. A comparison of formulas (34.2) and (34.3) yields the relation: $\sigma_{\text{Gaussian}} = \sqrt{2}\sigma$.

For numerical purposes, it is often useful to use Nolan's (1997) parameterization:

$$\log \phi_0(t) = \begin{cases} -\sigma^\alpha |t|^\alpha \{1 + i\beta \text{sign}(t) \tan\frac{\pi\alpha}{2}[(\sigma|t|)^{1-\alpha} - 1]\} + i\mu_0 t, & \alpha \neq 1, \\ -\sigma|t|\{1 + i\beta \text{sign}(t)\frac{2}{\pi}\log(\sigma|t|)\} + i\mu_0 t, & \alpha = 1, \end{cases} \tag{34.4}$$

which yields a CF (and hence the PDF and CDF) jointly continuous in all four parameters. The location parameters of the two representations (S and S^0) are related by $\mu = \mu_0 - \beta\sigma\tan\frac{\pi\alpha}{2}$ for $\alpha \neq 1$ and $\mu = \mu_0 - \beta\sigma\frac{2}{\pi}\log\sigma$ for $\alpha = 1$. Moreover, when $\alpha > 1$ the mean of the distribution exists and is equal to μ.

The latter is a result of a more general property: the pth moment of a stable random variable is finite if and only if $p < \alpha$. Hence, when $\alpha < 2$ the variance is infinite and the tails exhibit a power-law behavior (i.e. they are asymptotically equivalent to a Pareto law). More precisely, using a CLT type argument it can be shown that (Janicki and Weron 1994a; Samorodnitsky and Taqqu 1994):

$$\begin{cases} \lim_{x\to\infty} x^{\alpha}\mathbb{P}(X > x) = C_{\alpha}(1+\beta)\sigma^{\alpha}, \\ \lim_{x\to\infty} x^{\alpha}\mathbb{P}(X < -x) = C_{\alpha}(1+\beta)\sigma^{\alpha}, \end{cases} \tag{34.5}$$

where $C_{\alpha} = \left(2\int_0^{\infty} x^{-\alpha}\sin(x)dx\right)^{-1} = \frac{1}{\pi}\Gamma(\alpha)\sin\frac{\pi\alpha}{2}$. The convergence to the power-law tail varies for different α's and is slower for larger values of the tail index. Moreover, the tails of stable CDFs exhibit a crossover from an approximate power decay with exponent $\alpha > 2$ to the true tail with exponent α. This phenomenon is more visible for large α's (Weron 2001).

34.2.2 Truncating or Tempering the Tails

Mandelbrot's (1963) seminal work on applying stable distributions in finance gained support in the first few years after its publication, but subsequent works have questioned the stable distribution hypothesis, in particular, the stability under summation (for a review see Rachev and Mittnik 2000). Over the next few years, the stable law temporarily lost favor and alternative processes were suggested as mechanisms generating stock returns. In the mid 1990s the stable distribution hypothesis has made a dramatic comeback, at first in the econophysics literature. Several authors have found a very good agreement of high-frequency returns with a stable distribution up to six standard deviations away from the mean (Cont et al. 1997). For more extreme observations, however, the distribution they found fell off approximately exponentially. To cope with such observations the so called truncated Lévy distributions (TLD) were introduced by Mantegna and Stanley (1994). The original definition postulated a sharp truncation of the stable PDF at some arbitrary point. Later, however, exponential smoothing was proposed by Koponen (1995) leading to the following characteristic function:

$$\log\phi(t) = -\frac{\sigma^{\alpha}}{\cos\frac{\pi\alpha}{2}}\left[(t^2+\lambda^2)^{\alpha/2}\cos\left\{\alpha\arctan\frac{|t|}{\lambda}\right\} - \lambda^{\alpha}\right], \tag{34.6}$$

where $\alpha \neq 1$ is the tail exponent, σ is the scale parameter and λ is the truncation coefficient (for simplicity β and μ are set to zero here). Clearly the symmetric TLD reduces to the symmetric stable distribution ($\beta = \mu = 0$) when $\lambda = 0$. For small and intermediate returns the TLD behaves like a stable distribution, but for extreme returns the truncation causes the distribution to converge to a Gaussian (hence, all moments are finite), see Fig. 34.3. Thus the observation that the asset returns distribution is a TLD explains both the short-term stable behavior and the long run convergence to the normal distribution (for interesting insights on the CLT-type behavior of the TLD see a recent paper of Grabchak and Samorodnitsky 2010).

The (exponentially smoothed) TLD was not recognized in finance until the introduction of the KoBoL (Boyarchenko and Levendorskii 2000) and CGMY models (Carr et al. 2002). Around this time Rosinski coined the term under which

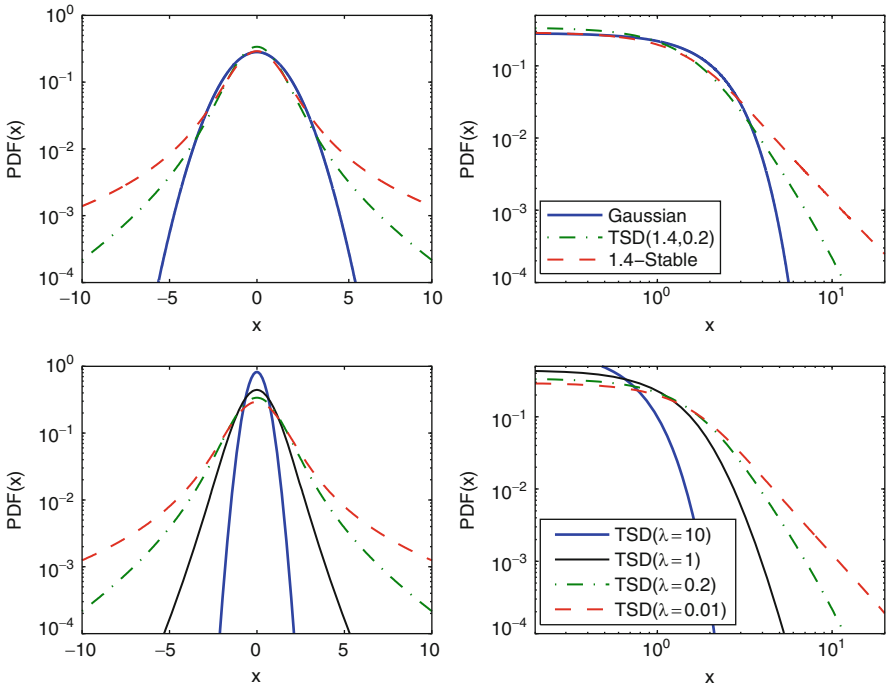


Fig. 34.3 *Top panels*: Semilog and loglog plots of symmetric 1.4-stable, symmetric tempered stable with $\alpha = 1.4$ and $\lambda = 0.2$, and Gaussian PDFs. *Bottom panels*: Semilog and loglog plots of symmetric tempered stable PDFs with $\alpha = 1.4$ and four different truncation coefficients: $\lambda = 10, 1, 0.2, 0.01$. Note, that for large λ's the distribution approaches the Gaussian (though with a different scale) and for small λ's the stable law with the same shape parameter α

the TLD is known today in the mathematics literature – tempered stable distribution (TSD; see Rosinski 2007).

Despite the interesting statistical properties, the TSDs (TLDs) have not been applied extensively to date. The most probable reason for this being the complicated definition of the TSD. Like for stable distributions, only the characteristic function is known. No closed form formulas exist for the density or the distribution functions. No integral formulas, like Zolotarev's (1986) for the stable laws (see Sect. 34.2.3), have been discovered to date. Hence, statistical inference is, in general, limited to ML utilizing the FFT technique for approximating the PDF (Bianchi et al. 2010; Grabchak 2008). Moreover, compared to the stable distribution, the TSD introduces one more parameter making the estimation procedure even more complicated. Other parameter fitting techniques proposed so far comprise a combination of ad hoc approaches and moment matching (Boyarchenko and Levendorskii 2000; Matacz 2000). Apart from a few special cases, also the simulation of TSD variables is cumbersome and numerically demanding (Bianchi et al. 2010; Kawai and Masuda 2011; Poirot and Tankov 2006).

34.2.3 Computation of Stable Density and Distribution Functions

The lack of closed form formulas for most stable densities and distribution functions has negative consequences. Numerical approximation or direct numerical integration have to be used, leading to a drastic increase in computational time and loss of accuracy. Of all the attempts to be found in the literature a few are worth mentioning. DuMouchel (1971) developed a procedure for approximating the stable CDF using Bergström's series expansion. Depending on the particular range of α and β, Holt and Crow (1973) combined four alternative approximations to compute the stable PDF. Both algorithms are computationally intensive and time consuming, making maximum likelihood estimation a nontrivial task, even for modern computers. In the late 1990s, two more efficient techniques have been proposed.

Mittnik et al. (1999) exploited the PDF–CF relationship and applied the fast Fourier transform (FFT). However, for data points falling between the equally spaced FFT grid nodes an interpolation technique has to be used. The authors suggested that linear interpolation suffices in most practical applications, see also Rachev and Mittnik (2000). Taking a larger number of grid points increases accuracy, however, at the expense of higher computational burden. Setting the number of grid points to $N = 2^{13}$ and the grid spacing to $h = 0.01$ allows to achieve comparable accuracy to the direct integration method (see below), at least for a range of α's typically found for financial data ($1.6 < \alpha < 1.9$).

As for the computational speed, the FFT based approach is faster for large samples, whereas the direct integration method favors small data sets since it can be computed at any arbitrarily chosen point. Mittnik et al. (1999) report that for $N = 2^{13}$ the FFT based method is faster for samples exceeding 100 observations and slower for smaller data sets. We must stress, however, that the FFT based approach is not as universal as the direct integration method – it is efficient only for large alpha's and only as far as the PDF calculations are concerned. When computing the CDF the former method must numerically integrate the density, whereas the latter takes the same amount of time in both cases.

The direct integration method, proposed by Nolan (1997, 1999) consists of a numerical integration of Zolotarev's (1986) formulas for the density or the distribution function. To save space we state only the formulas for the PDF. Complete formulas can be also found in Cizek et al. (2011), Chap. 1.

Set $\zeta = -\beta \tan \frac{\pi\alpha}{2}$. Then the density $f(x;\alpha,\beta)$ of a standard stable random variable in representation S^0, i.e. $X \sim S^0_\alpha(1,\beta,0)$, can be expressed as (note, that Zolotarev (1986, Sect. 2.2) used another parametrization):

- When $\alpha \neq 1$ and $x \neq \zeta$:

$$f(x;\alpha,\beta) = \frac{\alpha(x-\zeta)^{\frac{1}{\alpha-1}}}{\pi \mid \alpha - 1 \mid} \int_{-\theta_0}^{\frac{\pi}{2}} V(\boldsymbol{\theta};\alpha,\beta) \exp\left\{-(x-\zeta)^{\frac{\alpha}{\alpha-1}} V(\boldsymbol{\theta};\alpha,\beta)\right\} d\boldsymbol{\theta},$$

  for $x > \zeta$ and $f(x;\alpha,\beta) = f(-x;\alpha,-\beta)$ for $x < \zeta$,

- When $\alpha \neq 1$ and $x = \zeta$:

$$f(x;\alpha,\beta) = \frac{\Gamma(1+\frac{1}{\alpha})\cos(\xi)}{\pi(1+\zeta^2)^{\frac{1}{2\alpha}}},$$

- When $\alpha = 1$:

$$f(x;1,\beta) = \begin{cases} \frac{1}{2|\beta|} e^{\frac{\pi x}{2\beta}} \int_{-\frac{\pi}{2}}^{\frac{\pi}{2}} V(\boldsymbol{\theta};1,\beta) \exp\left\{-e^{\frac{\pi x}{2\beta}} V(\boldsymbol{\theta};1,\beta)\right\} d\boldsymbol{\theta}, & \beta \neq 0, \\ \frac{1}{\pi(1+x^2)}, & \beta = 0, \end{cases}$$

where

$$\xi = \begin{cases} \frac{1}{\alpha}\arctan(-\zeta), & \alpha \neq 1, \\ \frac{\pi}{2}, & \alpha = 1, \end{cases} \tag{34.7}$$

and

$$V(\boldsymbol{\theta};\alpha,\beta) = \begin{cases} (\cos\alpha\xi)^{\frac{1}{\alpha-1}} \left(\frac{\cos\boldsymbol{\theta}}{\sin\alpha(\xi+\boldsymbol{\theta})}\right)^{\frac{\alpha}{\alpha-1}} \frac{\cos\{\alpha\xi+(\alpha-1)\boldsymbol{\theta}\}}{\cos\boldsymbol{\theta}}, & \alpha \neq 1, \\ \frac{2}{\pi}\left(\frac{\frac{\pi}{2}+\beta\boldsymbol{\theta}}{\cos\boldsymbol{\theta}}\right) \exp\left\{\frac{1}{\beta}(\frac{\pi}{2}+\beta\boldsymbol{\theta})\tan\boldsymbol{\theta}\right\}, & \alpha = 1, \beta \neq 0. \end{cases}$$

To our best knowledge, currently no statistical computing environment offers the computation of stable density and distribution functions in its standard release. Users have to rely on third-party libraries or commercial products. A few are worth mentioning. The standalone program STABLE (downloadable from John Nolan's web page: http://academic2.american.edu/~jpnolan/stable/stable.html) is probably the most efficient. It was written in Fortran and calls several external IMSL routines, see Nolan (1997) for details. Apart from speed, the STABLE program also exhibits high relative accuracy (ca. 10^{-13}; for default tolerance settings) for extreme tail events and 10^{-10} for values used in typical financial applications (like approximating asset return distributions). The STABLE program is also available in library form through Robust Analysis Inc. (www.robustanalysis.com). This library provides interfaces to Matlab, S-plus/R and Mathematica.

In the late 1990s Diethelm Würtz has initiated the development of Rmetrics, an open source collection of S-plus/R software packages for computational finance (www.rmetrics.org). In the *fBasics* package stable PDF and CDF calculations are performed using the direct integration method, with the integrals being computed by R's function *integrate*.

The FFT based approach is utilized in Cognity, a commercial risk management platform that offers derivatives pricing and portfolio optimization based on the assumption of stably distributed returns (www.finanalytica.com). The FFT implementation is also available in Matlab (*stablepdf_fft.m*) from the Statistical Software Components repository (http://ideas.repec.org/c/boc/bocode/m429004.html).

34.2.4 Simulation of Stable Variables

Simulating sequences of stable random variables is not straightforward, since there are no analytic expressions for the inverse $F^{-1}(x)$ nor the CDF $F(x)$ itself. All standard approaches like the rejection or the inversion methods would require tedious computations. See Chap. II.3 for a review of non-uniform random number generation techniques.

A much more elegant and efficient solution was proposed by Chambers et al. (1976). They noticed that a certain integral formula derived by Zolotarev (1964) yielded the following algorithm:

- Generate a random variable U uniformly distributed on $(-\frac{\pi}{2}, \frac{\pi}{2})$ and an independent exponential random variable W with mean 1;
- For $\alpha \neq 1$ compute:

$$X = (1+\zeta^2)^{\frac{1}{2\alpha}} \frac{\sin\{\alpha(U+\xi)\}}{\{\cos(U)\}^{1/\alpha}} \left[\frac{\cos\{U-\alpha(U+\xi)\}}{W}\right]^{\frac{1-\alpha}{\alpha}}, \tag{34.8}$$

- For $\alpha = 1$ compute:

$$X = \frac{1}{\xi}\left\{\left(\frac{\pi}{2}+\beta U\right)\tan U - \beta \log\left(\frac{\frac{\pi}{2}W\cos U}{\frac{\pi}{2}+\beta U}\right)\right\}, \tag{34.9}$$

where ξ is given by eqn. (34.7). This algorithm yields a random variable $X \sim S_\alpha(1,\beta,0)$, in representation (34.3). For a detailed proof see Weron (1996).

Given the formulas for simulation of a standard stable random variable, we can easily simulate a stable random variable for all admissible values of the parameters α, σ, β and μ using the following property. If $X \sim S_\alpha(1,\beta,0)$ then

$$Y = \begin{cases} \sigma X + \mu, & \alpha \neq 1, \\ \sigma X + \frac{2}{\pi}\beta\sigma\log\sigma + \mu, & \alpha = 1, \end{cases} \tag{34.10}$$

is $S_\alpha(\sigma,\beta,\mu)$. It is interesting to note that for $\alpha = 2$ (and $\beta = 0$) the Chambers-Mallows-Stuck (CMS) method reduces to the well known Box-Muller algorithm for generating Gaussian random variables (Janicki and Weron 1994b) .

Many other approaches have been proposed in the literature, including application of Bergström and LePage series expansions, see Mantegna (1994) and Janicki and Kokoszka (1992), respectively. However, the CMS method is regarded as the fastest and the most accurate. On a PC equipped with a Core 2 Duo 2.66 GHz CPU one million variables are generated in about 1.03 s (using *stablernd.m* from the SSC repository: http://ideas.repec.org/c/boc/bocode/m429003.html), compared to about 0.058 s for one million standard normal random variables obtained via the Box-Muller algorithm (*randn.m* in Matlab). Because of its unquestioned superiority and relative simplicity, the CMS method is implemented in some statistical computing

environments (e.g. the *rstable* function in S-plus/R) even if no other routines related to stable distributions are provided.

34.2.5 Estimation of Parameters

The lack of known closed-form density functions for all but a few members of the stable family results in a considerable numerical complexity of statistical inference for these distributions. For instance, maximum likelihood (ML) estimates have to be based on numerical approximations or direct numerical integration of the formulas presented in Sect. 34.2.3. Consequently, ML estimation is difficult to implement and time consuming for samples encountered in modern finance. However, there are also other numerical methods that have been found useful in practice and are discussed in this section.

Given a sample $x_1, \ldots, x_n$ of i.i.d. $S_\alpha(\sigma, \beta, \mu)$ observations, in what follows, we provide estimates $\hat{\alpha}$, $\hat{\sigma}$, $\hat{\beta}$ and $\hat{\mu}$ of all four stable law parameters. We start the discussion with the simplest, fastest and ... least accurate quantile methods, then develop the slower, yet much more accurate sample CF methods and, finally, conclude with the slowest but most accurate ML approach.

All presented methods work quite well assuming that the sample under consideration is indeed stable. However, testing for stability is not an easy task. Despite some more or less successful attempts (Brcich et al. 2005; Cizek et al. 2011; Matsui and Takemura 2008; Paolella 2001), there are no standard, widely-accepted tests for assessing stability. A possible remedy may be to use one of the computationally less demanding tail exponent estimators (Fan 2006; Mittnik and Paolella 1999) or simply "visual inspection" to see whether the empirical densities resemble those of stable laws (Nolan 2001; Weron 2001).

Sample Quantile Methods

The origins of sample quantile methods for stable laws go back to Fama and Roll (1971), who provided very simple estimates for parameters of symmetric ($\beta = 0, \mu = 0$) stable laws with $\alpha > 1$. A decade later McCulloch (1986) generalized their method and provided consistent estimators of all four stable parameters (with the restriction $\alpha \geq 0.6$). After McCulloch define:

$$v_\alpha = \frac{x_{0.95} - x_{0.05}}{x_{0.75} - x_{0.25}} \quad \text{and} \quad v_\beta = \frac{x_{0.95} + x_{0.05} - 2x_{0.50}}{x_{0.95} - x_{0.05}}, \tag{34.11}$$

where x_f denotes the f-th population quantile, so that $S_\alpha(\sigma, \beta, \mu)(x_f) = f$. Statistics v_α and v_β are functions of α and β only, i.e. they are independent of both σ and μ. This relationship may be inverted and the parameters α and β may be viewed as functions of v_α and v_β. Substituting v_α and v_β by their sample values and applying linear interpolation between values found in tables given in McCulloch

(1986) yields estimators $\hat{\alpha}$ and $\hat{\beta}$. Scale and location parameters, σ and μ, can be estimated in a similar way. However, due to the discontinuity of the CF for $\alpha = 1$ and $\beta \neq 0$ in representation (34.3), this procedure is much more complicated.

In a recent paper, Dominicy and Veredas (2010) further extended the quantile approach by introducing the method of simulated quantiles. It is a promising approach which can also handle multidimensional cases as, for instance, the joint estimation of N univariate stable distributions (but with the constraint of a common tail index).

Sample Characteristic Function Methods

Given an i.i.d. random sample $x_1, \ldots, x_n$ of size n, define the sample CF by: $\hat{\phi}(t) = \frac{1}{n}\sum_{j=1}^{n} \exp(itx_j)$. Since $|\hat{\phi}(t)|$ is bounded by unity all moments of $\hat{\phi}(t)$ are finite and, for any fixed t, it is the sample average of i.i.d. random variables $\exp(itx_j)$. Hence, by the law of large numbers, $\hat{\phi}(t)$ is a consistent estimator of the CF $\phi(t)$.

To our best knowledge, Press (1972) was the first to use the sample CF in the context of statistical inference for stable laws. He proposed a simple estimation method for all four parameters, called the method of moments, based on transformations of the CF. However, the convergence of this method to the population values depends on the choice of four estimation points, whose selection is problematic.

Koutrouvelis (1980) presented a much more accurate regression-type method which starts with an initial estimate of the parameters and proceeds iteratively until some prespecified convergence criterion is satisfied. Each iteration consists of two weighted regression runs. The number of points to be used in these regressions depends on the sample size and starting values of α. Typically no more than two or three iterations are needed. The speed of the convergence, however, depends on the initial estimates and the convergence criterion.

The regression method is based on the following observations concerning the CF $\phi(t)$. First, from (34.3) we can easily derive:

$$\log(-\log|\phi(t)|^2) = \log(2\sigma^\alpha) + \alpha \log|t|. \tag{34.12}$$

The real and imaginary parts of $\phi(t)$ are for $\alpha \neq 1$ given by:

$$\Re\{\phi(t)\} = \exp(-|\sigma t|^\alpha)\cos\left[\mu t + |\sigma t|^\alpha \beta \operatorname{sign}(t)\tan\frac{\pi\alpha}{2}\right], \tag{34.13}$$

$$\Im\{\phi(t)\} = \exp(-|\sigma t|^\alpha)\sin\left[\mu t + |\sigma t|^\alpha \beta \operatorname{sign}(t)\tan\frac{\pi\alpha}{2}\right]. \tag{34.14}$$

Apart from considerations of principal values, (34.13–34.14) lead to:

$$\arctan\left(\frac{\Im\{\phi(t)\}}{\Re\{\phi(t)\}}\right) = \mu t + \beta\sigma^\alpha \tan\frac{\pi\alpha}{2}\operatorname{sign}(t)|t|^\alpha. \tag{34.15}$$

Equation (34.12) depends only on α and σ and suggests that we can estimate these two parameters by regressing $y = \log(-\log|\phi_n(t)|^2)$ on $w = \log|t|$ in the model: $y_k = m + \alpha w_k + \epsilon_k$, where t_k is an appropriate set of real numbers, $m = \log(2\sigma^\alpha)$, and ϵ_k denotes an error term. Koutrouvelis (1980) proposed to use $t_k = \frac{\pi k}{25}, k = 1, 2, \ldots, K$; with K ranging between 9 and 134 for different values of α and sample sizes.

Once $\hat{\alpha}$ and $\hat{\sigma}$ have been obtained and α and σ have been fixed at these values, estimates of β and μ can be obtained using (34.15). Next, the regressions are repeated with $\hat{\alpha}$, $\hat{\sigma}$, $\hat{\beta}$ and $\hat{\mu}$ as the initial parameters. The iterations continue until a prespecified convergence criterion is satisfied. Koutrouvelis proposed to use Fama and Roll's (1971) formula and the 25% truncated mean for initial estimates of σ and μ, respectively.

Kogon and Williams (1998) eliminated this iteration procedure and simplified the regression method. For initial estimation they applied McCulloch's method, worked with the continuous representation (34.4) of the CF instead of the classical one (34.3) and used a fixed set of only 10 equally spaced frequency points t_k. In terms of computational speed their method compares favorably to the original method of Koutrouvelis, see Table 34.1. It has a significantly better performance near $\alpha = 1$ and $\beta \neq 0$ due to the elimination of discontinuity of the CF. However, it returns slightly worse results for other values of α. Matlab implementations of McCulloch's quantile technique (*stabcull.m*) and the regression approach of Koutrouvelis (*stabreg.m*) are distributed with the MFE Toolbox accompanying the monograph of Weron (2006) and can be downloaded from www.ioz.pwr.wroc.pl/pracownicy/weron/MFE.htm.

A typical performance of the described estimators is summarized in Table 34.1. McCulloch's quantile technique, the regression approach of Koutrouvelis and the method of Kogon and Williams were applied to 1,000 simulated samples of two thousand $S_{1.7}(0.005, 0.1, 0.001)$ random numbers each. McCulloch's method yielded the worst, but acceptable estimates and computational time significantly lower than the regression approaches. On the other hand, both the Koutrouvelis and the Kogon-Williams implementations yielded good estimators with the latter

Table 34.1 Comparison of McCulloch's quantile technique, the regression approach of Koutrouvelis and the method of Kogon and Williams for 1,000 simulated samples of two thousand $S_{1.7}(0.005, 0.1, 0.001)$ random numbers each. Parameter estimates are mean values over 1,000 samples. Values of the Mean Absolute Percentage Error ($\text{MAPE}_{\boldsymbol{\theta}} = \frac{1}{n}\sum_{i=1}^{n}|\hat{\boldsymbol{\theta}} - \boldsymbol{\theta}|/\boldsymbol{\theta}$) are given in parentheses. In the last column CPU time factors (average computational times relative to the method of Kogon and Williams) for one sample of 2,000 random variables are provided

Method	$\hat{\alpha}$	$\hat{\sigma}$	$\hat{\beta}$	$\hat{\mu}$	CPU time factor
McCulloch	1.7070	0.0050	0.1108	0.0013	0.33×
	(2.72%)	(2.14%)	(108.97%)	(29.90%)	
Koutrouvelis	1.7021	0.0050	0.0933	0.0012	5.62×
	(1.66%)	(1.63%)	(91.99%)	(27.76%)	
Kogon-Williams	1.7030	0.0050	0.0934	0.0010	1.00×
	(1.91%)	(1.71%)	(99.64%)	(16.72%)	

performing considerably faster, but slightly less accurate. We have to say, though, that all methods had problems with estimating β. Like it or not, our search for the optimal estimation technique is not over yet. We have no other choice but turn to the last resort – the ML method.

Maximum Likelihood Method

The maximum likelihood (ML) estimation scheme for stable distributions does not differ from that for other laws, at least as far as the theory is concerned. For a vector of observations $x = (x_1, \ldots, x_n)$, the ML estimate of the parameter vector $\boldsymbol{\theta} = (\alpha, \sigma, \beta, \mu)$ is obtained by maximizing the log-likelihood function:

$$L_{\boldsymbol{\theta}}(x) = \sum_{i=1}^{n} \log \tilde{f}(x_i; \boldsymbol{\theta}), \tag{34.16}$$

where $\tilde{f}(\cdot; \boldsymbol{\theta})$ is the stable density function. The tilde denotes the fact that, in general, we do not know the explicit form of the stable PDF and have to approximate it numerically. The ML methods proposed in the literature differ in the choice of the approximating algorithm. However, all of them have an appealing common feature – under certain regularity conditions the ML estimator is asymptotically normal with the variance specified by the Fischer information matrix (DuMouchel 1973). The latter can be approximated either by using the Hessian matrix arising in maximization or, as in Nolan (2001), by numerical integration.

Because of computational complexity there are only a few documented attempts of estimating stable law parameters via maximum likelihood worth mentioning. DuMouchel (1971) developed an approximate ML method, which was based on grouping the data set into bins and using a combination of means to compute the density (FFT for the central values of x and series expansions for the tails) to compute an approximate log-likelihood function. This function was then numerically maximized.

Much better, in terms of accuracy and computational time, are more recent ML estimation techniques. Mittnik et al. (1999) utilized the FFT approach for approximating the stable density function, whereas Nolan (2001) used the direct integration method. Both approaches are comparable in terms of efficiency. The differences in performance are the result of different approximation algorithms, see Sect. 34.2.3. Matsui and Takemura (2006) further improved Nolan's method for the boundary cases, i.e. in the tail and mode of the densities and in the neighborhood of the Cauchy and the Gaussian distributions, but only in the symmetric stable case.

As Ojeda (2001) observes, the ML estimates are almost always the most accurate, closely followed by the regression-type estimates and McCulloch's quantile method. However, ML estimation techniques are certainly the slowest of all the discussed methods. For instance, ML estimation for a sample of 2,000 observations using

a gradient search routine which utilizes the direct integration method is over 11 thousand (!) times slower than the Kogon-Williams algorithm (calculations performed on a PC running STABLE ver. 3.13; see Sect. 34.2.3 where the program was briefly described). Clearly, the higher accuracy does not justify the application of ML estimation in many real life problems, especially when calculations are to be performed on-line. For this reason the program STABLE offers an alternative – a fast quasi ML technique. It quickly approximates stable densities using a 3-dimensional spline interpolation based on pre-computed values of the standardized stable density on a grid of (x, α, β) values. At the cost of a large array of coefficients, the interpolation is highly accurate over most values of the parameter space and relatively fast – only ca. 13 times slower than the Kogon-Williams algorithm for a sample of 2,000 observations.

Alternative Methods

Besides the popular methods discussed so far other estimation algorithms have been proposed in the literature. A Bayesian Markov chain Monte Carlo (MCMC) approach was initiated by Buckle (1995). It was later modified by Lombardi (2007) who used an approximated version of the likelihood, instead of the twice slower Gibbs sampler, and by Peters et al. (2011) who proposed likelihood-free Bayesian inference for stable models.

In a recent paper Garcia et al. (2010) estimate the stable law parameters with (constrained) indirect inference, a method particularly suited to situations where the model of interest is difficult to estimate but relatively easy to simulate. They use the skewed-t distribution as an auxiliary model, since it has the same number of parameters as the stable with each parameter playing a similar role.

34.3 Generalized Hyperbolic Distributions

34.3.1 Definitions and Basic Properties

The hyperbolic law saw its appearance in finance in the mid-1990s, when a number of authors reported that it provides a very good model for the distributions of daily stock returns from a number of leading German enterprises (Eberlein and Keller 1995; Kuchler et al. 1999). Since then it has become a popular tool in stock price modeling and market risk measurement (Bibby and Sorensen 2003; Chen et al. 2008; Eberlein et al. 1998; McNeil et al. 2005).

The origin of the distribution dates back to the 1940s and the empirical observation by Ralph Bagnold that the log-histogram of the size of sand particles tends to form a hyperbola. A formal mathematical description was developed by Barndorff-Nielsen (1977). The hyperbolic distribution provides the possibility of modeling heavier tails than the Gaussian, since its log-density forms a hyperbola

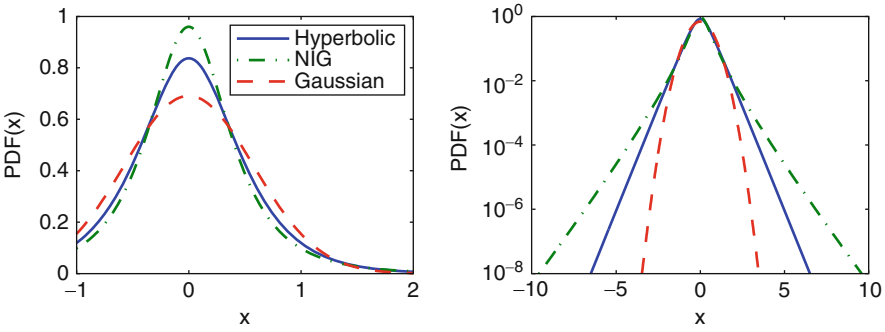


Fig. 34.4 Densities and log-densities of symmetric hyperbolic, NIG and Gaussian distributions having the same variance, see (34.31). The name of the hyperbolic distribution is derived from the fact that its log-density forms a hyperbola, which is clearly visible in the right panel

while that of the Gaussian is a parabola, see Fig. 34.4. As we will see later in this Section, the hyperbolic law is a member of a larger, versatile class of generalized hyperbolic (GH) distributions, which also includes the normal-inverse Gaussian (NIG) and variance-gamma (VG) distributions as special cases. For a concise review of special and limiting cases of the GH distribution see Paolella (2007), Chap. 9.

The Hyperbolic Distribution

The hyperbolic distribution is defined as a normal variance-mean mixture where the mixing distribution is the generalized inverse Gaussian (GIG) law with parameter $\lambda = 1$, i.e. it is conditionally Gaussian, see Barndorff-Nielsen (1977) and Barndorff-Nielsen and Blaesild (1981). More precisely, a random variable Z has the hyperbolic distribution if:

$$(Z|Y) \sim \mathrm{N}(\mu + \beta Y, Y), \tag{34.17}$$

where Y is a generalized inverse Gaussian $\mathrm{GIG}(\lambda = 1, \chi, \psi)$ random variable and $\mathrm{N}(m, s^2)$ denotes the Gaussian distribution with mean m and variance s^2. The GIG law is a very versatile positive domain distribution with the PDF given by:

$$f_{\mathrm{GIG}}(x) = \frac{(\psi/\chi)^{\lambda/2}}{2\mathrm{K}_\lambda(\sqrt{\chi\psi})} x^{\lambda-1} e^{-\frac{1}{2}(\chi x^{-1} + \psi x)}, \quad x > 0, \tag{34.18}$$

where the three parameters take values in one of the ranges: (1) $\chi > 0, \psi \geq 0$ if $\lambda < 0$, (2) $\chi > 0, \psi > 0$ if $\lambda = 0$ or (3) $\chi \geq 0, \psi = 0$ if $\lambda > 0$. The generalized inverse Gaussian law has a number of interesting properties that we will use later in this section. The distribution of the inverse of a GIG variable is again GIG but with a different λ, namely if:

$$Y \sim \mathrm{GIG}(\lambda, \chi, \psi) \quad \text{then} \quad Y^{-1} \sim \mathrm{GIG}(-\lambda, \chi, \psi). \tag{34.19}$$

A GIG variable can be also reparameterized by setting $a = \sqrt{\chi/\psi}$ and $b = \sqrt{\chi\psi}$, and defining $Y = a\tilde{Y}$, where:

$$\tilde{Y} \sim \text{GIG}(\lambda, b, b). \tag{34.20}$$

The normalizing constant $\text{K}_\lambda(t)$ in formula (34.18) is the modified Bessel function of the third kind with index λ, also known as the MacDonald function. It is defined as:

$$\text{K}_\lambda(t) = \frac{1}{2}\int_0^\infty x^{\lambda-1} e^{-\frac{1}{2}t\left(x+x^{-1}\right)} dx, \qquad t > 0. \tag{34.21}$$

In the context of hyperbolic distributions, the Bessel functions are thoroughly discussed in Barndorff-Nielsen and Blaesild (1981). Here we recall only two properties that will be used later. Namely, (1) $\text{K}_\lambda(t)$ is symmetric with respect to λ, i.e. $\text{K}_\lambda(t) = \text{K}_{-\lambda}(t)$, and (2) for $\lambda = \pm\frac{1}{2}$ it can be written in a simpler form:

$$\text{K}_{\pm\frac{1}{2}}(t) = \sqrt{\frac{\pi}{2}} t^{-\frac{1}{2}} e^{-t}. \tag{34.22}$$

For other values of λ numerical approximations of the integral in eqn. (34.21) have to be used, see e.g. Press et al. (1992).

Relation (34.17) implies that a hyperbolic random variable $Z \sim \text{H}(\psi, \beta, \chi, \mu)$ can be represented in the form: $Z \sim \mu + \beta Y + \sqrt{Y}\text{N}(0, 1)$, with the CF:

$$\phi_Z(u) = e^{iu\mu} \int_0^\infty e^{i\beta z u - \frac{1}{2}zu^2} dF_Y(z). \tag{34.23}$$

Here $F_Y(z)$ denotes the distribution function of a GIG random variable Y with parameter $\lambda = 1$, see (34.18). Hence, the hyperbolic PDF is given by:

$$f_{\text{H}}(x; \psi, \beta, \chi, \mu) = \frac{\sqrt{\psi/\chi}}{2\sqrt{\psi+\beta^2}K_1(\sqrt{\psi\chi})} e^{-\sqrt{\{\psi+\beta^2\}\{\chi+(x-\mu)^2\}}+\beta(x-\mu)}, \tag{34.24}$$

or in an alternative parameterization (with $\delta = \sqrt{\chi}$ and $\alpha = \sqrt{\psi+\beta^2}$) by:

$$f_{\text{H}}(x; \alpha, \beta, \delta, \mu) = \frac{\sqrt{\alpha^2-\beta^2}}{2\alpha\delta K_1(\delta\sqrt{\alpha^2-\beta^2})} e^{-\alpha\sqrt{\delta^2+(x-\mu)^2}+\beta(x-\mu)}. \tag{34.25}$$

The latter is more popular and has the advantage of $\delta > 0$ being the traditional scale parameter. Out of the remaining three parameters, α and β determine the shape, with α being responsible for the steepness and $0 \leq |\beta| < \alpha$ for the skewness, and $\mu \in R$ is the location parameter.

Finally, note that if we only have an efficient algorithm to compute K_1, the calculation of the PDF is straightforward. However, the CDF has to be numerically integrated from (34.24) or (34.25).

The General Class

The generalized hyperbolic (GH) law can be represented as a normal variance-mean mixture where the mixing distribution is the generalized inverse Gaussian law with any $\lambda \in \mathbb{R}$. Hence, the GH distribution is described by five parameters $\boldsymbol{\theta} = (\lambda, \alpha, \beta, \delta, \mu)$, using parameterization (34.25), and its PDF is given by:

$$f_{\mathrm{GH}}(x;\boldsymbol{\theta}) = \kappa \left\{\delta^2 + (x-\mu)^2\right\}^{\frac{1}{2}(\lambda-\frac{1}{2})} \mathrm{K}_{\lambda-\frac{1}{2}}\left(\alpha\sqrt{\delta^2+(x-\mu)^2}\right) e^{\beta(x-\mu)}, \tag{34.26}$$

where:

$$\kappa = \frac{(\alpha^2-\beta^2)^{\frac{\lambda}{2}}}{\sqrt{2\pi}\alpha^{\lambda-\frac{1}{2}}\delta^{\lambda}\mathrm{K}_{\lambda}(\delta\sqrt{\alpha^2-\beta^2})}. \tag{34.27}$$

The tail behavior of the GH density is "semi-heavy", i.e. the tails are lighter than those of non-Gaussian stable laws, but much heavier than Gaussian. Formally they are characterized by the following asymptotic relation (Barndorff-Nielsen and Blaesild 1981):

$$f_{\mathrm{GH}}(x) \approx |x|^{\lambda-1} e^{(\mp\alpha+\beta)x} \quad \text{for} \quad x \to \pm\infty, \tag{34.28}$$

which can be interpreted as exponential "tempering" of the power-law tails (compare with the TSD described in Sect. 34.2.2).

For $|\beta + z| < \alpha$ the moment generating function of the GH law takes the form:

$$M(z) = e^{\mu z}\left\{\frac{\alpha^2-\beta^2}{\alpha^2-(\beta+z)^2}\right\}^{\frac{\lambda}{2}} \frac{\mathrm{K}_{\lambda}\left(\delta\sqrt{\alpha^2-(\beta+z)^2}\right)}{\mathrm{K}_{\lambda}\left(\delta\sqrt{\alpha^2-\beta^2}\right)}. \tag{34.29}$$

Note, that $M(z)$ is smooth, i.e. infinitely many times differentiable, near 0 and hence every moment exists. If we set $\zeta = \delta\sqrt{\alpha^2-\beta^2} = \sqrt{\psi\chi}$ then the first two moments lead to the following formulas for the mean and variance of a GH random variable:

$$\mathbb{E}X = \mu + \frac{\beta\delta^2}{\zeta}\frac{\mathrm{K}_{\lambda+1}(\zeta)}{\mathrm{K}_{\lambda}(\zeta)}, \tag{34.30}$$

$$\mathrm{Var}X = \delta^2\left[\frac{\mathrm{K}_{\lambda+1}(\zeta)}{\zeta\mathrm{K}_{\lambda}(\zeta)} + \frac{\beta^2\delta^2}{\zeta^2}\left\{\frac{\mathrm{K}_{\lambda+2}(\zeta)}{\mathrm{K}_{\lambda}(\zeta)} - \left(\frac{\mathrm{K}_{\lambda+1}(\zeta)}{\zeta\mathrm{K}_{\lambda}(\zeta)}\right)^2\right\}\right]. \tag{34.31}$$

The Normal-Inverse Gaussian Distribution

The normal-inverse Gaussian (NIG) laws were introduced by Barndorff-Nielsen (1995) as a subclass of the generalized hyperbolic laws obtained for $\lambda = -\frac{1}{2}$. The density of the NIG distribution is given by:

$$f_{\text{NIG}}(x) = \frac{\alpha\delta}{\pi} e^{\delta\sqrt{\alpha^2-\beta^2}+\beta(x-\mu)} \frac{\text{K}_1(\alpha\sqrt{\delta^2+(x-\mu)^2})}{\sqrt{\delta^2+(x-\mu)^2}}. \tag{34.32}$$

Like for the hyperbolic distribution the calculation of the PDF is straightforward, but the CDF has to be numerically integrated from eqn. (34.32).

At the expense of four parameters, the NIG distribution is able to model asymmetric distributions with "semi-heavy" tails. However, if we let $\alpha \to 0$ the NIG distribution converges to the Cauchy distribution (with location parameter μ and scale parameter δ), which exhibits extremely heavy tails. Obviously, the NIG distribution may not be adequate to deal with cases of extremely heavy tails such as those of Pareto or non-Gaussian stable laws. However, empirical experience suggests excellent fits of the NIG law to financial data (Karlis 2002; Karlis and Lillestöl 2004; Venter and de Jongh 2002).

Moreover, the class of normal-inverse Gaussian distributions possesses an appealing feature that the class of hyperbolic laws does not have. Namely, it is closed under convolution, i.e. a sum of two independent NIG random variables is again NIG (Barndorff-Nielsen 1995). In particular, if X_1 and X_2 are independent NIG random variables with common parameters α and β but having different scale and location parameters $\delta_{1,2}$ and $\mu_{1,2}$, respectively, then $X = X_1 + X_2$ is $\text{NIG}(\alpha, \beta, \delta_1 + \delta_1, \mu_1 + \mu_2)$. This feature is especially useful in time scaling of risks, e.g. in deriving 10-day risks from daily risks. Only two subclasses of the generalized hyperbolic distributions are closed under convolution. The other class with this important property is the class of variance-gamma (VG) distributions, which is obtained when δ is equal to 0. This is only possible for $\lambda > 0$ and $\alpha > |\beta|$. The VG distributions (with $\beta = 0$) were introduced to finance by Madan and Seneta (1990), long before the popularity of GH and NIG laws.

34.3.2 Simulation of Generalized Hyperbolic Variables

The most natural way of simulating GH variables stems from the fact that they can be represented as normal variance-mean mixtures. Since the mixing distribution is the GIG law, the resulting algorithm reads as follows:

1. Simulate a random variable $Y \sim \text{GIG}(\lambda, \chi, \psi) = \text{GIG}(\lambda, \delta^2, \alpha^2 - \beta^2)$;
2. Simulate a standard normal random variable N;
3. Return $X = \mu + \beta Y + \sqrt{Y}N$.

The algorithm is fast and efficient if we have a handy way of simulating GIG variates. For $\lambda = -\frac{1}{2}$, i.e. when sampling from the so-called inverse Gaussian (IG) distribution, there exists an efficient procedure that utilizes a transformation yielding two roots. It starts with the observation that if we let $\vartheta = \sqrt{\chi/\psi}$ then the IG(χ, ψ) density (= GIG$(-\frac{1}{2}, \chi, \psi)$; see eqn. (34.18)) of Y can be written as:

$$f_{\mathrm{IG}}(x) = \sqrt{\frac{\chi}{2\pi x^3}} \exp\left\{\frac{-\chi(x-\vartheta)^2}{2x\vartheta^2}\right\}. \tag{34.33}$$

Now, following Shuster (1968) we may write:

$$V = \frac{\chi(Y-\vartheta)^2}{Y\vartheta^2} \sim \chi^2_{(1)}, \tag{34.34}$$

i.e. V is distributed as a chi-square random variable with one degree of freedom. As such it can be simply generated by taking a square of a standard normal random number. Unfortunately, the value of Y is not uniquely determined by eqn. (34.34). Solving this equation for Y yields two roots:

$$y_1 = \vartheta + \frac{\vartheta}{2\chi}\left(\vartheta V - \sqrt{4\vartheta\chi V + \vartheta^2 V^2}\right) \quad \text{and} \quad y_2 = \frac{\vartheta^2}{y_1}.$$

The difficulty in generating observations with the desired distribution now lies in choosing between the two roots. Michael et al. (1976) showed that Y can be simulated by choosing y_1 with probability $\vartheta/(\vartheta + y_1)$. So for each random observation V from a $\chi^2_{(1)}$ distribution the smaller root y_1 has to be calculated. Then an auxiliary Bernoulli trial is performed with probability $p = \vartheta/(\vartheta + y_1)$. If the trial results in a "success", y_1 is chosen; otherwise, the larger root y_2 is selected. The *rnig* function of the Rmetrics collection of software packages for S-plus/R (see also Sect. 34.2.3 where Rmetrics was briefly described), utilize this routine.

In the general case, the GIG distribution – as well as the (generalized) hyperbolic law – can be simulated via the rejection algorithm. An adaptive version of this technique is used to obtain hyperbolic random numbers in the *rhyp* function of Rmetrics. Rejection is also implemented in the HyperbolicDist package for S-plus/R developed by David Scott, see the R-project home page http://cran.r-project.org. The package utilizes a version of the algorithm proposed by Atkinson (1982), i.e. rejection coupled either with a two ("GIG algorithm" for any admissible value of λ) or a three part envelope ("GIGLT1 algorithm" for $0 \leq \lambda < 1$). Envelopes, also called hat or majorizing functions, provide an upper limit for the PDF of the sampled distribution. The proper choice of such functions can substantially increase the speed of computations, see Chap. II.3. As Atkinson (1982) shows, once the parameter values for these envelopes have been determined, the algorithm efficiency is reasonable for most values of the parameter space. However, finding the appropriate parameters requires optimization and makes the technique burdensome.

This difficulty led to a search for a short algorithm which would give comparable efficiencies but without the drawback of extensive numerical optimizations. A solution, based on the "ratio-of-uniforms" method, was provided by Dagpunar (1989). First, recalling properties (34.19) and (34.20), observe that we only need to find a method to simulate $\tilde{Y} \sim \mathrm{GIG}(\lambda, b, b)$ variables and only for $\lambda \geq 0$. Next, define the relocated variable $\tilde{Y}_m = \tilde{Y} - m$, where $m = \frac{1}{b}(\lambda - 1 + \sqrt{(\lambda-1)^2 + b^2})$ is the mode of the density of $\tilde{Y}$. Then the relocated variable can be generated by taking $\tilde{Y}_m = V/U$, where the pair (U, V) is uniformly distributed over the region $\{(u, v) : 0 \leq u \leq \sqrt{h(v/u)}\}$ with:

$$h(t) = (t+m)^{\lambda-1} \exp\left(-\frac{b}{2}\frac{t+m+1}{t+m}\right), \quad \text{for} \quad t \geq -m.$$

Since this region is irregularly shaped, it is more convenient to generate the pair (U, V) uniformly over a minimal enclosing rectangle $\{(u, v) : 0 \leq u \leq u_+, v_- \leq v \leq v_+\}$. Finally, the variate (V/U) is accepted if $U^2 \leq h(V/U)$. The efficiency of the algorithm depends on the method of deriving and the actual choice of u_+ and $v_\pm$. Further, for $\lambda \leq 1$ and $b \leq 1$ there is no need for the shift at mode m. Such a version of the algorithm is implemented in UNU.RAN, a library of C functions for non-uniform random number generation developed at the Vienna University of Economics, see http://statistik.wu-wien.ac.at/unuran.

34.3.3 Estimation of Parameters

Maximum Likelihood Method

The parameter estimation of GH distributions can be performed by the ML method, since there exist closed-form formulas (although, involving special functions) for the densities of these laws. The computational burden is not as heavy as for stable laws, but it still is considerable.

In general, the ML estimation algorithm is as follows. For a vector of observations $x = (x_1, \ldots, x_n)$, the ML estimate of the parameter vector $\boldsymbol{\theta} = (\lambda, \alpha, \beta, \delta, \mu)$ is obtained by maximizing the log-likelihood function:

$$L(x; \boldsymbol{\theta}) = \log \kappa + \frac{\lambda - \frac{1}{2}}{2} \sum_{i=1}^{n} \log(\delta^2 + (x_i - \mu)^2) + \\ + \sum_{i=1}^{n} \log \mathrm{K}_{\lambda-\frac{1}{2}}(\alpha\sqrt{\delta^2 + (x_i-\mu)^2}) + \sum_{i=1}^{n} \beta(x_i - \mu), \quad (34.35)$$

where κ is defined by (34.27). Obviously, for hyperbolic ($\lambda = 1$) distributions the algorithm uses simpler expressions of the log-likelihood function due to relation (34.22).

The routines proposed in the literature differ in the choice of the optimization scheme. The first software product that allowed statistical inference with hyperbolic distributions – the HYP program – used a gradient search technique, see Blaesild and Sorensen (1992). In a large simulation study Prause (1999) utilized the bracketing method. Matlab functions *hypest.m* and *nigest.m* distributed with the MFE Toolbox (Weron 2006) use yet another technique – the downhill simplex method, with slight modifications due to parameter restrictions.

The main factor for the speed of the estimation is the number of modified Bessel functions to compute. Note, that for $\lambda = 1$ (i.e. the hyperbolic distribution) this function appears only in the constant κ. For a data set with n independent observations we need to evaluate n and $n + 1$ Bessel functions for NIG and generalized hyperbolic distributions, respectively, whereas only one for the hyperbolic. This leads to a considerable reduction in the time necessary to calculate the likelihood function in the hyperbolic case. Prause (1999) reported a reduction of ca. 33%, however, the efficiency results are highly sample and implementation dependent.

We also have to say that the optimization is challenging. Some of the parameters are hard to separate since a flat-tailed GH distribution with a large scale parameter is hard to distinguish from a fat-tailed distribution with a small scale parameter, see Barndorff-Nielsen and Blaesild (1981) who observed such a behavior already for the hyperbolic law. The likelihood function with respect to these parameters then becomes very flat, and may have local mimima. In the case of NIG distributions Venter and de Jongh (2002) proposed simple estimates of α and β that can be used as staring values for the ML scheme. Starting from relation (34.28) for the tails of the NIG density (i.e. with $\lambda = -1/2$) they derived the following approximation:

$$\alpha - \beta \sim \frac{1}{2} \frac{x_{1-f} + \mathbb{E}(X|X > x_{1-f})}{\mathbb{E}(X^2|X > x_{1-f}) - x_{1-f}\mathbb{E}(X|X > x_{1-f})},$$

$$\alpha + \beta \sim -\frac{1}{2} \frac{x_f + \mathbb{E}(X|X < x_f)}{\mathbb{E}(X^2|X < x_f) - x_f\mathbb{E}(X|X < x_f)},$$

where x_f is the f-th population quantile, see Sect. 34.2.5. After the choice of a suitable value for f, Venter and de Jongh used $f = 5\%$, the "tail estimates" of α and β are obtained by replacing the quantiles and expectations by their sample values in the above relations.

Another method of providing the starting values for the ML scheme was suggested by Prause (1999). He estimated the parameters of a symmetric ($\beta = \mu = 0$) GH law with a reasonable kurtosis (i.e. with $\delta\alpha \approx 1.04$) that had the variance equal to that of the empirical distribution.

Other Methods

Besides the ML approach other estimation methods have been proposed in the literature. Prause (1999) tested different estimation techniques by replacing the log-

likelihood function with other score functions, like the Anderson-Darling and or L^p-norms. But the results were disappointing. Karlis and Lillestöl (2004) made use of the MCMC technique (see Chap. II.4), however, again the results obtained were not impressive. Karlis (2002) described an Expectation-Maximization (EM) type algorithm (see Chap. II.6) for ML estimation of the NIG distribution. The algorithm can be programmed in any statistical package supporting Bessel functions and it has all the properties of the standard EM algorithm, like sure, but slow, convergence, parameters in the admissible range, etc. Recently Fragiadakis et al. (2009) used this approach to construct goodness-of-fit tests for symmetric NIG distributions. The tests are based on a weighted integral incorporating the empirical CF of suitably standardized data. The EM scheme can be also generalized to multivariate GH distributions (but with fixed λ, see Protassov 2004).

34.4 Value at Risk, Portfolios and Heavy Tails

34.4.1 Copulas

The facts presented in Sect. 34.1 clearly show that we not only can, but must use heavy tailed alternatives to the Gaussian law in order to obtain acceptable estimates of market losses. But can we substitute the Gaussian distribution with other distributions in Value at Risk (Expected Shortfall) calculations for whole portfolios of assets? Recall, that the definition of VaR utilizes the quantiles of the portfolio returns distribution and not the returns distribution of individual assets in the portfolio. If all asset return distributions are assumed to be Gaussian and linearly dependent then the portfolio distribution is multivariate normal and well known statistical tools can be applied (Franke et al. 2008). However, when asset returns are distributed according to a different law (or different laws!) then the multivariate distribution may be hard to tackle. In particular, linear correlation may no longer be a meaningful measure of dependence.

It turns out that in this context the concept of copulas is very helpful (Joe 1997; Nelsen 1999). In rough terms, a copula is a multivariate distribution function defined on the unit cube $[0,1]^n$, i.e. $C : [0,1]^n \to [0,1]$. What is important for VaR calculations is that a copula enables us to construct a multivariate distribution function from the marginal (possibly different) distribution functions of n individual asset returns in a way that takes their dependence structure into account. This dependence structure may be no longer measured by correlation, but by other adequate functions like rank correlation, comonotonicity or tail dependence (McNeil et al. 2005). Moreover, it can be shown that for every multivariate distribution function there exists a copula which contains all information on dependence (this is the essence of "Sklar's theorem"). For example, if the random variables are independent, then the independence (or product) copula is just the product of n variables: $C(u_1, \ldots, u_n) = u_1 \cdot \ldots \cdot u_n$. If the random variables have a multivariate

normal distribution with a given covariance matrix then the Gaussian copula is obtained.

Copula functions do not impose any restrictions on the model, so in order to reach a model that is to be useful in a given risk management problem, a particular specification of the copula must be chosen. From the wide variety of copulas that exist probably the elliptical and Archimedean copulas are the ones most often used in applications. Elliptical copulas are simply the copulas of elliptically contoured (or elliptical) distributions, e.g. (multivariate) normal, t, symmetric stable and symmetric generalized hyperbolic (Fang et al. 1987). Rank correlation and tail dependence coefficients can be easily calculated for elliptical copulas. There are, however, drawbacks – elliptical copulas do not have closed form expressions, are restricted to have radial symmetry and have all marginal distributions of the same type. These restrictions may disqualify elliptical copulas from being used in some risk management problems. In particular, there is usually a stronger dependence between big losses (e.g. market crashes) than between big gains. Clearly, such asymmetries cannot be modeled with elliptical copulas. In contrast to elliptical copulas, all commonly encountered Archimedean copulas have closed form expressions. Their popularity also stems from the fact that they allow for a great variety of different dependence structures. Many interesting parametric families of copulas are Archimedean, including the well known Clayton:

$$C_C(u_1, u_2; \boldsymbol{\theta}) = \left(u_1^{-\boldsymbol{\theta}} + u_2^{-\boldsymbol{\theta}} - 1\right)^{-1/\boldsymbol{\theta}}, \tag{34.36}$$

and Frank:

$$C_F(u_1, u_2; \boldsymbol{\theta}) = -\frac{1}{\boldsymbol{\theta}} \log\left(1 + \frac{(e^{-\boldsymbol{\theta}u_1} - 1)(e^{-\boldsymbol{\theta}u_2} - 1)}{e^{-\boldsymbol{\theta}} - 1}\right), \tag{34.37}$$

copulas. In Fig. 34.5 their densities are plotted for sample values of the dependence parameter $\boldsymbol{\theta}$ corresponding to the estimates obtained for the portfolio considered in Sect. 34.4.2.

Calibration

Calibration of multivariate copula-based models can be performed in a number of ways. Simultaneous estimation of all parameters (of the copula and marginal distributions) using the full (or exact) maximum likelihood (FML, EML) approach is the most direct estimation method. However, this comes at the price of solving a complicated, multivariate and typically nonlinear optimization problem.

A feasible alternative which exploits the attractive feature of copulas for which the dependence structure is independent of the marginal distributions is known as the Inference Functions for Margins (IFM, see Fusai and Roncoroni 2008; Joe 1997) or the sequential two-step maximum likelihood (TSML, see Trivedi and Zimmer 2005) method. In this approach, first the marginals are estimated on univariate data, then

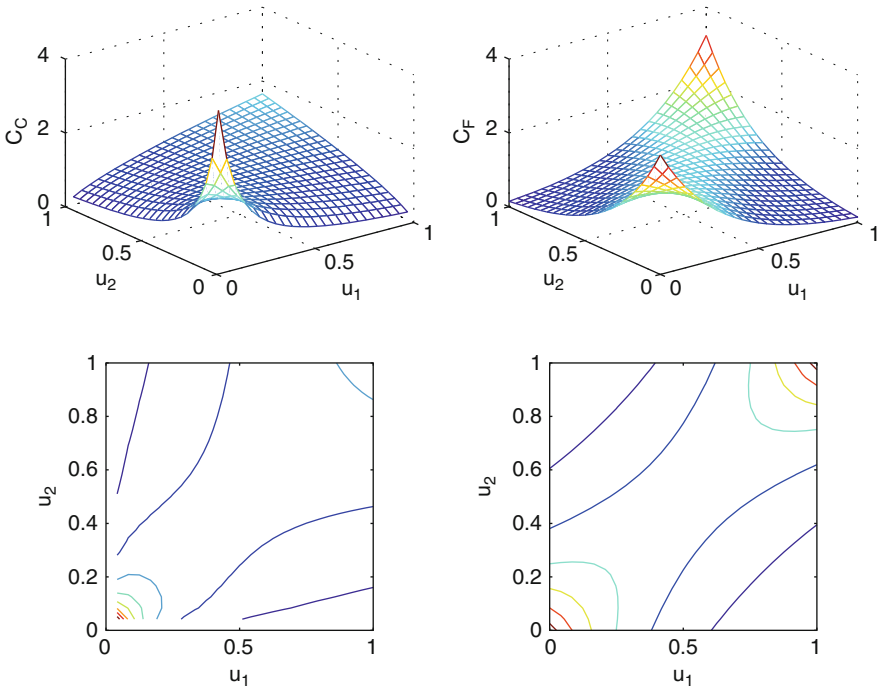


Fig. 34.5 Densities of a two-dimensional Clayton copula C_C with $\boldsymbol{\theta}_C = 0.6515$ (*top left*) and a two-dimensional Frank copula C_F with $\boldsymbol{\theta}_F = 3.0884$ (*top right*). *Bottom panels*: Contour plots of the densities in the upper panels. Note, that with this choice of the dependence parameters $(\boldsymbol{\theta}_C, \boldsymbol{\theta}_F)$ the Clayton copula models strong dependence between big losses $(u_1, u_2 \to 0)$, while the Frank copula focuses on both, big losses and big gains

the dependence parameter $\boldsymbol{\theta}$ is calibrated after the estimated marginals have been substituted into the copula. The IFM method has additional variants depending upon whether the first step is implemented parametrically or nonparametrically (using kernel density estimates). For a bivariate copula the procedure could be summarized as follows:

1. Using ML estimate the univariate marginal densities $\hat{f}_j(\mathbf{x}_j)$, $j = 1, 2$, for two random i.i.d. samples $\mathbf{x}_j = \{x_{j,1}, \ldots, x_{j,N}\}$. In the nonparametric variant a kernel density estimator is used to find the $\hat{f}_j$'s.
2. Set $\mathbf{u}_j = \hat{F}_j(\mathbf{x}_j)$, where F_j is the CDF corresponding to f_j. The vectors $\mathbf{u}_j$ may be treated as realizations of uniform random variables.
3. Given $\mathbf{u}_j$, $j = 1, 2$, and a copula C with density c, the dependence parameter $\boldsymbol{\theta}$ can be estimated as follows:

$$\hat{\boldsymbol{\theta}} = \arg\max_{\boldsymbol{\theta}} \sum_{i=1}^{N} \log c(\hat{u}_{1,i}, \hat{u}_{2,i}; \boldsymbol{\theta}). \tag{34.38}$$

Simulating Random Variables from Copulas

For risk management purposes, we are interested in the Value at Risk of a portfolio of assets. While analytical methods for the computation of VaR exist for the multivariate normal distribution (i.e. for the Gaussian copula), in most other cases we have to use Monte Carlo simulations. A general technique for random variate generation from copulas is the conditional distributions method (Nelsen 1999), also called conditional sampling (Fusai and Roncoroni 2008). A random vector $(u_1, \ldots, u_n)^T$ having a joint distribution function C can be generated by the following algorithm:

1. Simulate $u_1 \sim U(0, 1)$,
2. For $k = 2, \ldots, n$ simulate $u_k \sim F_{U_k|U_1 \ldots U_{k-1}}(\cdot|u_1, \ldots, u_{k-1})$.

The above function is the conditional distribution of the variable U_k given the values of $U_1, \ldots, U_{k-1}$, i.e.:

$$F_{U_k|U_1 \ldots U_{k-1}}(u_k|u_1, \ldots, u_{k-1}) \stackrel{\text{def}}{=} \mathbb{P}(U_k \leq u_k | U_1 = u_1, \ldots, U_{k-1} = u_{k-1}) = \frac{\partial^{k-1} C_k(u_1, \ldots, u_k)}{\partial u_1 \ldots \partial u_{k-1}} \Big/ \frac{\partial^{k-1} C_{k-1}(u_1, \ldots, u_{k-1})}{\partial u_1 \ldots \partial u_{k-1}},$$

where C_k's are k-dimensional margins of the n-dimensional copula C, i.e.

$$C_k(u_1, \ldots, u_k) = C(u_1, \ldots, u_k, 1, \ldots, 1).$$

The main drawback of this method is the fact that it involves a differentiation step for each dimension of the problem. Also simulation of $u_k \sim F_{U_k|U_1 \ldots U_{k-1}}(\cdot|u_1, \ldots, u_{k-1})$ may be non-trivial. It requires drawing v from a uniform distribution and setting $u_k \sim F^{-1}_{U_k|U_1 \ldots U_{k-1}}(v|u_1, \ldots, u_{k-1})$, with often having to compute F^{-1} numerically. Hence, the conditional distributions technique is typically not practical in higher dimensions. For this reason, alternative methods have been developed for specific types of copulas. For instance, random variables distributed according to Archimedean copula functions can be generated by the mixture of powers method of Marshall and Olkin (1988), which utilizes Laplace transforms. Also conditional sampling can be simplified for Archimedean copulas with the conditional distribution $F_{U_k|U_1 \ldots U_{k-1}}$ rewritten in terms of the copula generator function. A comprehensive list of algorithms can be found in Alexander (2008), Fusai and Roncoroni (2008), and McNeil et al. (2005).

34.4.2 Empirical Evidence

In this section we apply the techniques discussed so far to financial data. We want to build a VaR model for a hypothetical portfolio consisting (in equal parts) of the

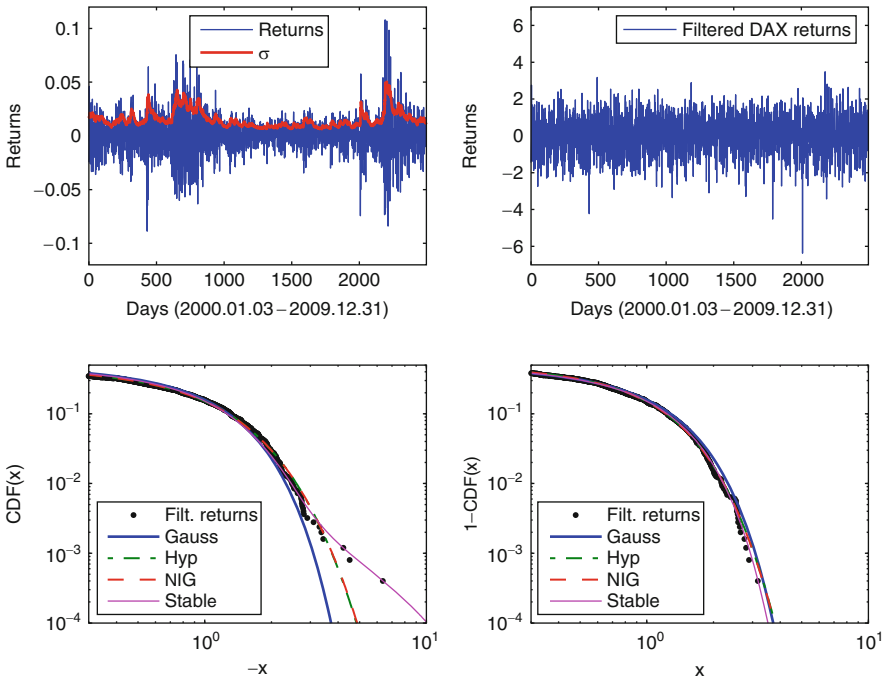


Fig. 34.6 *Top left*: DAX index (log-)returns and the GARCH(1,1)-based daily volatility estimate σ_t. *Top right*: σ_t-filtered DAX returns, see eqn. (34.40). *Bottom panels*: The left and right tails of the CDF of filtered returns and of four fitted distributions: Gaussian, hyperbolic, NIG, and stable (for parameter estimates see Table 34.2). Note, that the left tail is significantly heavier than the right tail. The latter is pretty well modeled by the Gaussian law

Deutsche Aktienindex (DAX) and the Polish WIG20 index. Both are blue chip stock market indexes. DAX consists of the 30 major German companies trading on the Frankfurt Stock Exchange and WIG20 of the 20 major Polish companies trading on the Warsaw Stock Exchange. We use daily closing index values from the period January 3, 2000 – December 31, 2009. Eliminating missing values (mostly German and Polish holidays) we end up with 2,494 (log-)returns for each index, see the top left panels in Figs. 34.6 and 34.7.

Building a VaR Model

Like most financial time series, also these index returns contain volatility clusters which imply that the probability of a specific incurred loss is not the same on each day. During days of higher volatility we should expect larger than usual losses and during calmer days – smaller than usual. To remove volatility clusters it is necessary to model the process that generates them. Following Barone-Adesi et al. (1999) and

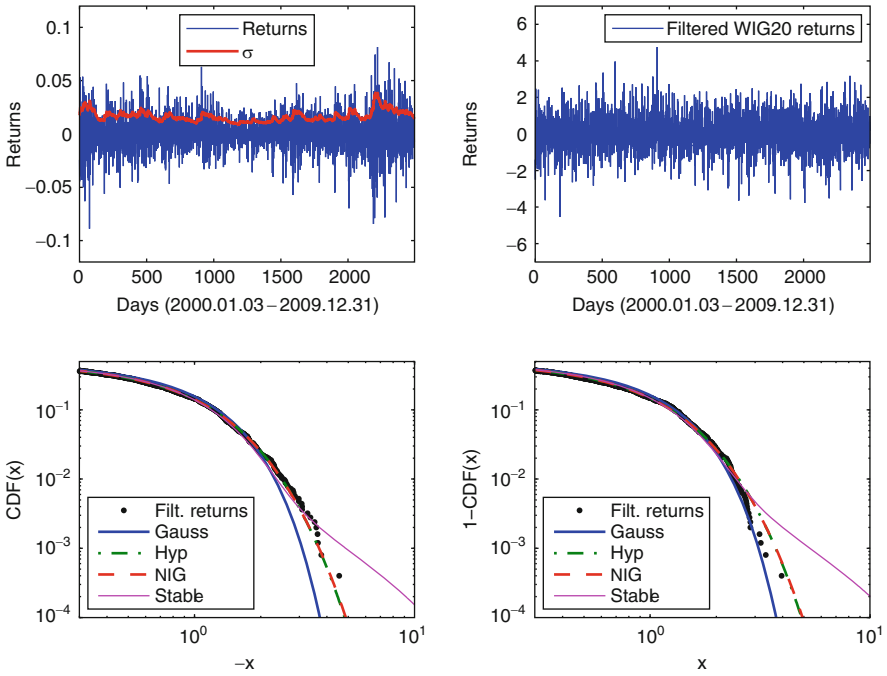


Fig. 34.7 *Top left*: WIG20 index (log-)returns and the GARCH(1,1)-based daily volatility estimate σ_t. *Top right*: σ_t-filtered WIG20 returns, see eqn. (34.40). *Bottom panels*: The left and right tails of the CDF of filtered returns and of four fitted distributions: Gaussian, hyperbolic, NIG, and stable (for parameter estimates see Table 34.3). Unlike for the DAX returns, the left tail is not significantly heavier than the right tail

Kuester et al. (2006) we eliminate volatility clusters by filtering the returns r_t with a GARCH(1,1) process:

$$r_t = \sigma_t \epsilon_t, \quad \text{with} \quad \sigma_t = c_0 + c_1 r_{t-1}^2 + d_1 \sigma_{t-1}^2, \tag{34.39}$$

where c_0, c_1 and d_1 are constants and

$$\epsilon_t = \frac{r_t}{\sigma_t}, \tag{34.40}$$

are the filtered returns, see the top right panels in Figs. 34.6 and 34.7. We could also insert a moving average term in the conditional mean to remove any serial dependency if needed.

To find the right model class for each dataset we fit four distributions: Gaussian, hyperbolic, NIG, and stable to the DAX and WIG20 filtered returns. The results are presented in Tables 34.2 and 34.3. We also compare both fits using Kolmogorov (K) and Anderson-Darling (AD) test statistics (D'Agostino and Stephens 1986). The latter may be treated as a weighted Kolmogorov statistics which puts more weight

Table 34.2 Gaussian, hyperbolic, NIG and stable fits to 2,494 filtered returns of the Deutsche Aktienindex (DAX) from the period January 3, 2000 – December 31, 2009. The values of the Kolmogorov (K) and Anderson-Darling (AD) goodness-of-fit statistics suggest the hyperbolic distribution as the best model, with the NIG law following closely by

Distribution	Parameters				Statistics	
	α	σ (δ)	β	μ	K	AD
Gaussian		1.000		−0.007	1.858	3.554
Hyperbolic	2.769	2.009	–0.542	0.508	**0.779**	**0.623**
NIG	2.536	2.350	–0.555	0.520	0.796	0.643
Stable	1.957	0.682	–1.000	−0.019	1.389	1.697

Table 34.3 Gaussian, hyperbolic, NIG and stable fits to 2,494 filtered returns of the Polish WIG20 index from the period January 3, 2000 – December 31, 2009. Like for the filtered DAX returns, the values of the Kolmogorov (K) and Anderson-Darling (AD) goodness-of-fit statistics suggest the hyperbolic distribution as the best model, with the NIG law following closely by

Distribution	Parameters				Statistics	
	α	σ (δ)	β	μ	K	AD
Gaussian		1.003		0.008	1.584	3.500
Hyperbolic	2.048	1.284	−0.002	0.010	**0.681**	**0.452**
NIG	1.732	1.747	0.030	−0.022	0.682	0.463
Stable	1.890	0.654	0.207	0.020	0.881	1.095

to the differences in the tails of the distributions. Although no asymptotic results are known for stable or generalized hyperbolic laws, approximate critical values for these goodness-of-fit tests can be obtained via the bootstrap technique (Cizek et al. 2011; Stute et al. 1993). In this chapter, though, we will not perform hypothesis testing and just compare the test values. Naturally, the lower the values the better the fit. For both datasets, both statistics suggest the hyperbolic distribution as the best model, with the NIG law following closely by. Note, that for the DAX filtered returns the left tail is significantly heavier than the right tail, with the latter being pretty well modeled by the Gaussian law, see the bottom panels in Figs. 34.6 and 34.7. This is also confirmed by very negative skewness parameters (β). In contrast, the WIG20 filtered returns are roughly symmetric.

Computing VaR and Backtesting the Models

Based on the goodness-of-fit results of Sect. 34.4.2 we use two distributional classes for the marginals – hyperbolic or NIG (for simplicity the same distributional class is used for both marginals). The dependence structure is modeled either by the Clayton or the Frank copula. This leaves us with four models: Clayton(Hyp,Hyp), Clayton(NIG,NIG), Frank(Hyp,Hyp), and Frank(NIG,NIG). As a benchmark model we use the Filtered Historical Simulation (FHS) introduced by Barone-Adesi et al. (1999), which has been documented to perform pretty well under different circumstances (Kuester et al. 2006).

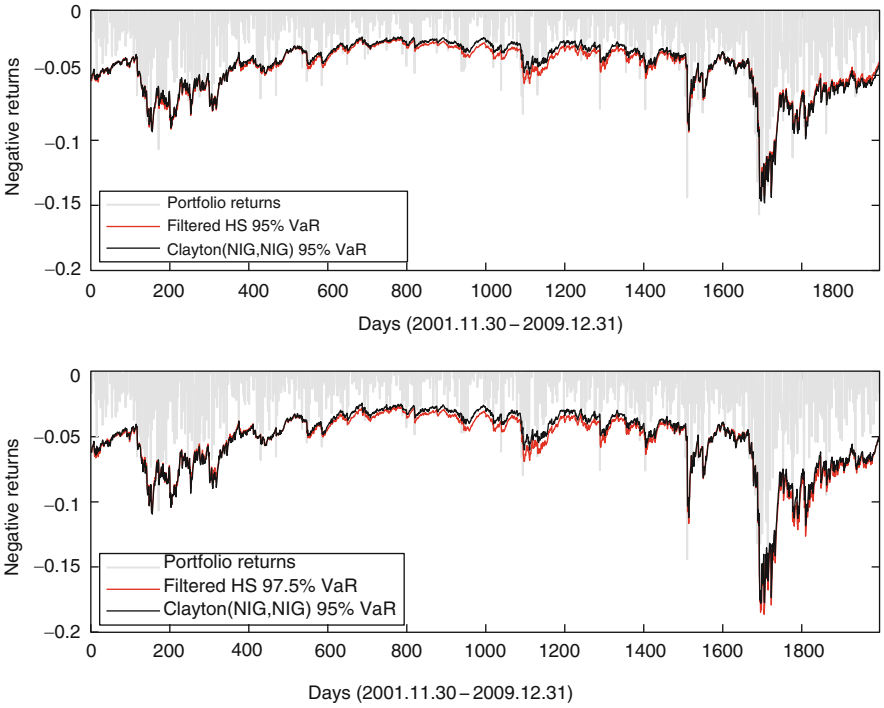


Fig. 34.8 The DAX-WIG20 portfolio returns and one day ahead $\text{VaR}_{95\%}$ (*top*) and $\text{VaR}_{97.5\%}$ (*bottom*) estimates for the FHS and Clayton(NIG,NIG) models. The latter yields slighly lower (in absolute terms) VaR numbers and a better conditional coverage, especially at the 97.5% level (see Table 34.4)

The dynamic simulation scenario consists of computing the one day VaR at four different levels (90%, 95%, 97.5%, and 99%) for a rolling window of 500 observations (roughly two years of data). This leaves us with $2{,}494 - 500 = 1{,}994$ daily VaR estimates for each of the five methods, see Fig. 34.8. These forecasts are then compared with the actual returns of the portfolio in the backtesting procedure. Each day is marked as 0 if VaR (at a given level) is not exceeded and 1 otherwise. The resulting sequence is Bernoulli distributed with parameter $(1 - c)$, see eqn. (34.1).

Several statistical tests have been proposed for backtesting purposes. In this study, we use Christoffersen's (1998) approach to test the conditional coverage. This model independent approach is designed to overcome the clustering effect. The tests are carried out in the likelihood ratio (LR) framework. Three LR statistics are calculated: for the unconditional coverage, independence and conditional coverage. The former two are distributed asymptotically as $\chi^2(1)$ and the latter as $\chi^2(2)$. If we condition on the first observation, then the conditional coverage LR test statistics is the sum of the other two.

Table 34.4 The p-values of Christoffersen's (1998) conditional coverage test for the five considered VaR models at four different levels (90%, 95%, 97.5%, and 99%). The Clayton(NIG,NIG) model yields the best coverage over all VaR levels, with the Clayton(Hyp,Hyp) model and FHS following closely by. The models based on the Frank copula fail at the high VaR levels

Model	$\text{VaR}_{90\%}$	$\text{VaR}_{95\%}$	$\text{VaR}_{97.5\%}$	$\text{VaR}_{99\%}$
FHS	0.360	**0.719**	0.441	0.778
Clayton(Hyp, Hyp)	0.462	0.715	0.953	0.685
Frank(Hyp, Hyp)	0.401	0.103	0.028	0.003
Clayton(NIG,NIG)	**0.505**	**0.719**	**0.971**	**0.814**
Frank(NIG,NIG)	0.401	0.148	0.021	0.001

The p-values of Christoffersen's (1998) conditional coverage test for the five considered VaR models are displayed in Table 34.4. The models based on the Frank copula produce disappointing results and fail at the high VaR levels. The Clayton(NIG,NIG) model yields the best coverage over all VaR levels. The Clayton(Hyp,Hyp) model is a little worse, despite a slightly better fit of the hyperbolic distribution to the filtered returns of the portfolio components, see Tables 34.2 and 34.3. The FHS model follows closely by and underperforms only at the 97.5% level. Perhaps, as Pritsker (2006) argues, two years of daily data (roughly 500 daily returns) may not contain enough extreme outliers to accurately compute 2.5% VaR.

Copulas allow us to construct models which go beyond the standard notions of correlation and multivariate Gaussian distributions. As such, in conjunction with alternative asset returns distributions discussed earlier in this chapter, they yield an ideal tool to model a wide variety of financial portfolios and products. No wonder they are gradually becoming an element of good risk management practice.

References

Alexander, C.: Market Risk Analysis, Wiley, Chichester (2008)

Artzner, P., Delbaen, F., Eber, J.-M., Heath, D.: Coherent measures of risk, Math. Fin. **9**, 203–228 (1999)

Atkinson, A.C.: The simulation of generalized inverse Gaussian and hyperbolic random variables. SIAM J. Sci. Stat. Comput. **3**, 502–515 (1982)

Barndorff-Nielsen, O.E.: Exponentially decreasing distributions for the logarithm of particle size, Proc. Roy. Soc. Lond. A **353**, 401–419 (1977)

Barndorff-Nielsen, O.E.: Normal\\Inverse Gaussian Processes and the Modelling of Stock Returns, Research Report 300, Department of Theoretical Statistics, University of Aarhus (1995)

Barndorff-Nielsen, O.E., Blaesild, P.: Hyperbolic distributions and ramifications: Contributions to theory and applications. In: Taillie, C., Patil, G., Baldessari, B. (eds.) Statistical Distributions in Scientific Work, vol. 4, pp.19–44. Reidel, Dordrecht (1981)

Barone-Adesi, G., Giannopoulos, K., Vosper, L.: VaR without correlations for portfolios of derivative securities. J. Futures Market. **19**(5), 583–602 (1999)

Basle Committee on Banking Supervision: An internal model-based approach to market risk capital requirements. http://www.bis.org. (1995)

Bianchi, M.L., Rachev, S.T., Kim, Y.S., Fabozzi, F.J.: Tempered stable distributions and processes in finance: Numerical analysis. In: Corazza, M., Claudio, P.P. (eds.) Mathematical and Statistical Methods for Actuarial Sciences and Finance, Springer, New York (2010)
Bibby, B.M., Sørensen, M.: Hyperbolic processes in finance, In: Rachev, S.T. (eds.) Handbook of Heavy-tailed Distributions in Finance, North Holland, NY, USA (2003)
Blaesild, P., Sorensen, M.: HYP – a Computer Program for Analyzing Data by Means of the Hyperbolic Distribution, Research Report 248, Department of Theoretical Statistics, Aarhus University (1992)
Boyarchenko, S.I., Levendorskii, S.Z.: Option pricing for truncated Lévy processes. Int. J. Theor. Appl. Fin. **3**, 549–552 (2000)
Brcich, R.F., Iskander, D.R., Zoubir, A.M.: The stability test for symmetric alpha stable distributions. IEEE Trans. Signal Process. **53**, 977–986 (2005)
Buckle, D.J.: Bayesian inference for stable distributions. J. Am. Stat. Assoc. **90**, 605–613 (1995)
Carr, P., Geman, H., Madan, D.B., Yor, M.: The fine structure of asset returns: An empirical investigation, J. Bus. **75**, 305–332 (2002)
Chambers, J.M., Mallows, C.L., Stuck, B.W.: A Method for Simulating Stable Random Variables. J. Am. Stat. Assoc. **71**, 340–344 (1976)
Chen, Y., Härdle, W., Jeong, S.-O.: Nonparametric risk management with generalized hyperbolic distributions. J. Am. Stat. Assoc. **103**, 910–923 (2008)
Christoffersen, P.: Evaluating interval forecasts. Int. Econ. Rev. **39**(4), 841–862 (1998)
Cizek, P., Härdle, W., Weron, R.: Statistical Tools for Finance and Insurance, 2nd edition, Springer, Berlin (2011)
Cont, R., Potters, M., Bouchaud, J.-P.: Scaling in stock market data: Stable laws and beyond, In: Dubrulle, B., Graner, F., Sornette, D. (eds.) Scale Invariance and Beyond, Proceedings of the CNRS Workshop on Scale Invariance, Springer, Berlin (1997)
D'Agostino, R.B., Stephens, M.A.: Goodness-of-Fit Techniques, Marcel Dekker, New York (1986)
Dagpunar, J.S.: An easily implemented generalized inverse gaussian generator. Comm. Stat. Simul. **18**, 703–710 (1989)
Danielsson, J., Hartmann, P., De Vries, C.G.: The cost of conservatism: Extreme returns, value at risk and the Basle multiplication factor. Risk **11**, 101–103 (1998)
Dominicy, Y., Veredas, D.: The Method of Simulated Quantiles, ECARES working paper, pp. 2010–008 (2010)
DuMouchel, W.H.: Stable Distributions in Statistical Inference, Ph.D. Thesis, Department of Statistics, Yale University (1971)
DuMouchel, W.H.: On the asymptotic normality of the maximum–likelihood estimate when sampling from a stable distribution. Ann. Stat. **1**(5), 948–957 (1973)
Eberlein, E., Keller, U.: Hyperbolic distributions in finance. Bernoulli **1**, 281–299 (1995)
Eberlein, E., Keller, U., Prause, K.: New insights into the smile, mispricing and Value at Risk: The hyperbolic model, J. Bus. **71**, 371–406 (1998)
Fama, E.F., Roll, R.: Parameter estimates for symmetric stable distributions. J. Am. Stat. Assoc. **66**, 331–338 (1971)
Fan, Z.: arameter estimation of stable distributions. Comm. Stat. Theor. Meth. **35**(2), 245–255 (2006)
Fang, K.-T., Kotz, S., Ng, K.-W.: Symmetric Multivariate and Related Distributions, Chapman & Hall, London (1987)
Franke, J., Härdle, W., Hafner, Ch.: Statistics of Financial Markets, 2nd ed., Springer, Berlin (2008)
Fragiadakis, K., Karlis, D., Meintanis, S.G.: Tests of fit for normal inverse Gaussian distributions. Stat. Meth. **6**, 553–564 (2009)
Fusai, G., Roncoroni, A.: Implementing Models in Quantitative Finance: Methods and Cases, Springer, New York (2008)
Garcia, R., Renault, E., Veredas, D.: Estimation of stable distributions by indirect inference, Journal of Econometrics **161**(2), 325–337 (2011)
Grabchak, M.: Maximum likelihood estimation of parametric tempered stable distributions on the real line with applications to finance, Ph.D. thesis, Cornell University (2010)

Grabchak, M., Samorodnitsky, G.: Do financial returns have finite or infinite variance? A paradox and an explanation. Quantitative Finance **10**(8), 883–893 (2010)
Guillaume, D.M., Dacorogna, M.M., Dave, R.R., Müller, U.A., Olsen, R.B., Pictet, O.V.: From the birdï£¡s eye to the microscope: A survey of new stylized facts of the intra-daily foreign exchange markets. Fin. Stoch. **1**, 95–129 (1997)
Holt, D.R., Crow, E.L.: Tables and graphs of the stable probability density functions. J. Res. Natl. Bur. Stand. B **77B**, 143–198 (1973)
Janicki, A., Kokoszka, P.: Computer investigation of the rate of convergence of LePage type series to alpha-stable random variables. Statistica **23**, 365–373 (1992)
Janicki, A., Weron, A.: Can one see α-stable variables and processes, Stat. Sci. **9**, 109–126 (1994a)
Janicki, A., Weron, A.: Simulation and Chaotic Behavior of α-Stable Stochastic Processes, Marcel Dekker (1994b)
Joe, H.: Multivariate Models and Dependence Concepts, Chapman & Hall, London (1997)
Jorion, P.: Value at Risk: The New Benchmark for Managing Financial Risk, (3rd edn.), McGraw-Hill, NY, USA (2006)
Karlis, D.: An EM type algorithm for maximum likelihood estimation for the Normal Inverse Gaussian distribution. Stat. Probab. Lett. **57**, 43–52 (2002)
Karlis, D., Lillestöl, J.: Bayesian estimation of NIG models via Markov chain Monte Carlo methods. Appl. Stoch. Models Bus. Ind. **20**(4), 323–338 (2004)
Kawai, R., Masuda, H.: On simulation of tempered stable random variates, Journal of Computational and Applied Mathematics **235**(8), 2873-2887 (2011)
Kogon, S.M., Williams, D.B.: Characteristic function based estimation of stable parameters, In: Adler, R., Feldman, R., Taqqu, M. (eds.) A Practical Guide to Heavy Tails, pp. 311–335 Birkhauser, Basel (Boston/Stuttgart) (1998)
Koponen, I.: Analytic approach to the problem of convergence of truncated Levy flights towards the Gaussian stochastic process. Phys. Rev. E **52**, 1197–1199 (1995)
Koutrouvelis, I.A.: Regression–Type Estimation of the Parameters of Stable Laws J. Am. Stat. Assoc. **75**, 918–928 (1980)
Kuester, K., Mittnik, S., Paolella, M.S.: Value-at-Risk prediction: A comparison of alternative strategies. J. Fin. Econometrics **4**(1), 53–89 (2006)
Küchler, U., Neumann, K., Sørensen, M., Streller, A.: Stock returns and hyperbolic distributions. Math. Comput. Modell. **29**, 1–15 (1999)
Lévy, P.: Calcul des Probabilites, Gauthier Villars (1925)
Lombardi, M.J.: Bayesian inference for α-stable distributions: A random walk MCMC approach. Comput. Stat. Data Anal. **51**(5), 2688–2700 (2007)
Madan, D.B., Seneta, E.: The variance gamma (V.G.) model for share market returns. J. Bus. **63**, 511–524 (1990)
Mandelbrot, B.B.: The variation of certain speculative prices. J. Bus. **36**, 394–419 (1963)
Mantegna, R.N.: Fast, accurate algorithm for numerical simulation of Levy stable stochastic processes. Phy. Rev. E **49**, 4677–4683 (1994)
Mantegna, R.N., Stanley, H.E.: Stochastic processes with ultraslow convergence to a Gaussian: The truncated Lévy flight. Phys. Rev. Lett. **73**, 2946–2949 (1994)
Marshall, A.W., Olkin, I.: Families of multivariate distributions. J. Am. Stat. Assoc. **83**, 834–841 (1988)
Matacz, A.: Financial modeling and option theory with the truncated lévy process. Int. J. Theor. Appl. Fin. **3**(1), 143–160 (2000)
Matsui, M., Takemura, A.: 'Some improvements in numerical evaluation of symmetric stable density and its derivatives. Comm. Stat. Theor. Meth. **35**(1), 149–172 (2006)
Matsui, M., Takemura, A.: Goodness-of-fit tests for symmetric stable distributions – empirical characteristic function approach. TEST **17**(3), 546–566 (2008)
McCulloch, J.H.: Simple consistent estimators of stable distribution parameters. Comm. Stat. Simul. **15**, 1109–1136 (1986)
McNeil, A.J., Rüdiger, F., Embrechts, P.: Quantitative Risk Management, Princeton University Press, Princeton, NJ (2005)

Michael, J.R., Schucany, W.R., Haas, R.W.: Generating random variates using transformations with multiple roots. Am. Stat. **30**, 88–90 (1976)
Mittnik, S., Doganoglu, T., Chenyao, D.: Computing the Probability Density Function of the Stable Paretian Distribution. Math. Comput. Modell. **29**, 235–240 (1999)
Mittnik, S., Paolella, M.S.: A simple estimator for the characteristic exponent of the stable Paretian distribution. Math. Comput. Modell. **29**, 161–176 (1999)
Mittnik, S., Rachev, S.T., Doganoglu, T., Chenyao, D.: Maximum likelihood estimation of stable paretian models. Math. Comput. Modell. **29**, 275–293 (1999)
Nelsen, R.B.: An Introduction to Copulas, Springer, New York (1999)
Nolan, J.P.: Numerical calculation of stable densities and distribution functions. Comm. Stat. Stoch. Model. **13**, 759–774 (1997)
Nolan, J.P.: An Algorithm for Evaluating Stable Densities in Zolotarev's (M) Parametrization. Math. Comput. Modell. **29**, 229–233 (1999)
Nolan, J.P.: Maximum Likelihood Estimation and Diagnostics for Stable Distributions. In: Barndorff-Nielsen, O.E., Mikosch, T., Resnick, S. (eds.) Lévy Processes, Brikhäuser, Boston (2001)
Nolan, J.P.: Stable Distributions – Models for Heavy Tailed Data, Birkhäuser, Boston (2012); In progress, Chapter 1 online at academic2.american.edu/∼jpnolan.
Ojeda, D.: Comparison of stable estimators, Ph.D. Thesis, Department of Mathematics and Statistics, American University (2001)
Paolella, M.S.: Testing the stable Paretian assumption. Math. Comput. Modell. **34**, 1095–1112 (2001)
Paolella, M.S.: Intermediate Probability: A Computational Approach, Wiley, Chichester (2007)
Peters, G.W., Sisson, S.A., Fan, Y.: Likelihood-free Bayesian inference for α-stable models, Computational Statistics & Data Analysis, forthcoming (2011)
Poirot, J., Tankov, P.: Monte Carlo option pricing for tempered stable (CGMY) processes. Asia. Pac. Financ. Market. **13**(4), 327-344 (2006)
Prause, K.: The Generalized Hyperbolic Model: Estimation, Financial Derivatives, and Risk Measures, Ph.D. Thesis, Freiburg University (1999); http://www.freidok.uni-freiburg.de/volltexte/15.
Press, S.J.: Estimation in univariate and multivariate stable distribution. J. Am. Stat. Assoc. **67**, 842–846 (1972)
Press, W., Teukolsky, S., Vetterling, W., Flannery, B.: Numerical Recipes in C, Cambridge University Press (1992); See also: http://www.nr.com.
Pritsker, M.: The hidden dangers of historical simulation. J. Bank. Finance **30**(2), 561–582 (2006)
Protassov, R.S.: EM-based maximum likelihood parameter estimation for multivariate generalized hyperbolic distributions with fixed λ. Stat. Comput. **14**, 67–77 (2004)
Rachev, S., Mittnik, S.: Stable Paretian Models in Finance, Wiley, New York (2000)
Rosinski, J.: Tempering stable processes. Stoch. Process. Appl. **117**(6), 677–707 (2007)
Samorodnitsky, G., Taqqu, M.S.: Stable Non–Gaussian Random Processes, Chapman & Hall, London (1994)
Shuster, J.: On the inverse gaussian distribution function. J. Am. Stat. Assoc. **63**, 1514–1516 (1968)
Stahl, G.: Three cheers. Risk **10**, 67–69 (1997)
Stute, W., Manteiga, W.G., Quindimil, M.P.: Bootstrap based goodness-of-fit-tests. Metrika **40**, 243–256 (1993)
Trivedi, P.K., Zimmer, D.M.: Copula modeling: An introduction for practitioners. Foundations Trend. Econometrics **1**(1), 1–111 (2005)
Venter, J.H., de Jongh, P.J.: Risk estimation using the Normal Inverse Gaussian distribution. J. Risk **4**, 1–23 (2002)
Weron, R.: On the Chambers-Mallows-Stuck Method for Simulating Skewed Stable Random Variables, Stat. Probab. Lett. **28**, 165–171 (1996); See also R. Weron (1996) Correction to: On the Chambers-Mallows-Stuck Method for Simulating Skewed Stable Random Variables, Working Paper, Available at MPRA: http://mpra.ub.uni-muenchen.de/20761/.

Weron, R.: Levy–Stable distributions revisited: Tail index > 2 does not exclude the Levy–Stable regime. Int. J. Modern Phys. C **12**, 209–223 (2001)
Weron, R.: Modeling and Forecasting Electricity Loads and Prices: A Statistical Approach, Wiley, Chichester (2006)
Zolotarev, V.M.: On representation of stable laws by integrals. Selected Trans. Math. Stat. Probab. **4**, 84–88 (1964)
Zolotarev, V.M.: One–Dimensional Stable Distributions, American Mathematical Society (1986)

Chapter 35
Econometrics

Luc Bauwens and Jeroen V.K. Rombouts

35.1 Introduction

Since the last decade we live in a digitalized world where many actions in human and economic life are monitored. This produces a continuous stream of new, rich and high quality data in the form of panels, repeated cross-sections and long time series. These data resources are available to many researchers at a low cost. This new era is fascinating for econometricians who can address many open economic questions. To do so, new models are developed that call for elaborate estimation techniques. Fast personal computers play an integral part in making it possible to deal with this increased complexity.

This chapter reviews econometric models for which statistical inference requires intensive numerical computations. A common feature of such models is that they incorporate unobserved (or latent) variables, in addition to observed ones. This often implies that the latent variables have to be integrated from the joint distribution of latent and observed variables. The implied integral is typically of high dimension and not available analytically. Simulation methods are almost always required to solve the computational issue, but they bring new problems. A general introduction on simulation based inference can be found in Gouriéroux and Monfort (1997) and Mariano et al. (2000).

The organisation of this chapter is as follows. The second section deals with limited dependent variable models, with a focus on multi-period discrete choice dynamic models. The third section treats the stochastic volatility (SV) model, used

L. Bauwens (✉)
Université Catholique de Louvain, Louvain-la-Neuve, Belgium
e-mail: luc.bauwens@uclouvain.be

J.V.K. Rombouts
HEC Montréal, CIRANO, CIRPEE, CORE, Montreal, Canada
e-mail: jeroen.rombouts@hec.ca

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_35,

in finance and financial econometrics to calibrate the volatility of asset returns, as an alternative to the class of generalized autoregressive conditional heteroskedastic (GARCH) models. It also reviews related dynamic duration models. The fourth section deals with finite mixture models and the last one with change point models. Illustrative applications drawn from the recent literature are used. Programs and data are on the web site www.core.ucl.ac.be/econometrics/Bauwens/HBCS/HBCS.htm.

All the models discussed in this chapter are parametric. Nonparametric and semiparametric models may induce additional computational complexity. We refer to Pagan and Ullah (1999), Horowitz (1998), Li and Racine (2006), Fan and Yao (2006), and Chap. III.10 of this volume for examples on these methods.

35.2 Limited Dependent Variable Models

This section deals with models in which the dependent variable is discrete. Many interesting problems like labour force participation, voting behavior, transport mode choice and brand choice are discrete in nature. In particular, we consider discrete choice models in the case where panel data are available. This allows, for example, to follow individuals with their choices over time, so that richer behavioural models can be constructed. Although the number of parameters in these models does not necessarily increase, the likelihood function, and therefore estimation, becomes more complex. In this section we describe the multinomial multiperiod probit and the mixed multinomial logit model.

We refer to Maddala (1983) for a general introduction to limited dependent and qualitative variables in econometrics and to Franses and Paap (2001) for a basic introduction motivating such models in relation to marketing.

35.2.1 Multinomial Multiperiod Probit

Definition

Denote by U_{ijt} the unobserved utility perceived by individual i who chooses alternative j at time t. This utility may be modelled as follows

$$U_{ijt} = \boldsymbol{X}_{ijt}^T \boldsymbol{\beta} + \epsilon_{ijt} \tag{35.1}$$

where $i = 1, \ldots, I,\ j = 1, \ldots, J,\ t = 1, \ldots, T_i$, $\boldsymbol{X}_{ijt}$ is a k-dimensional vector of explanatory variables, $\boldsymbol{\beta}$ is a k-dimensional parameter vector and ϵ_{ijt} is a random shock known to individual i. This individual chooses alternative j in period t if

$$U_{ijt} > U_{imt} \quad \forall\, j \neq m. \tag{35.2}$$

We observe $\boldsymbol{d}_i = (d_{i1}, \ldots, d_{iT_i})^T$ where $d_{it} = j$ if individual i chooses alternative j at time t. We suppose that there is always only one choice by each individual at each period, i.e. choices are mutually exclusive. The multinomial multiperiod probit model is obtained by assuming

$$\boldsymbol{\epsilon}_i = (\epsilon_{i11}, \ldots, \epsilon_{iJ1}, \ldots, \epsilon_{i1T_i}, \ldots, \epsilon_{iJT_i})^T \sim \text{IIDN}(0, \boldsymbol{\Sigma}) \tag{35.3}$$

Consequently,

$$\begin{aligned} P_i &= \text{P}(\boldsymbol{d}_i) = \text{P}\left(\bigcap_{m \neq d_{it}} \bigcap_{t=1}^{T_i} U_{i,d_{it},t} > U_{imt} \right) \\ &= \text{P}\left(\bigcap_{m \neq d_{it}} \bigcap_{t=1}^{T_i} \epsilon_{i,d_{it},t} - \epsilon_{imt} > (\boldsymbol{X}_{imt} - \boldsymbol{X}_{i,d_{it},t})^T \boldsymbol{\beta} \right) \end{aligned} \tag{35.4}$$

is a $(T_i \times J)$-variate integral. However, since individual choices are based on utility comparisons, it is conventional to work in utility differences relative to alternative J. If we multiply the utilities in (35.1) by a constant, we see that the probability event in (35.4) is invariant, thus a different scaling of the utilities does not alter the choices of the individuals. The rescaled relative utility is then defined as

$$\begin{aligned} \tilde{U}_{ijt} &= (U_{ijt} - U_{iJt})(\sigma_{11} + \sigma_{JJ} - 2\sigma_{1J})^{-1/2} \\ &= \left((\boldsymbol{X}_{ijt} - \boldsymbol{X}_{iJt})^T \boldsymbol{\beta} + \epsilon_{ijt} - \epsilon_{iJt} \right) (\sigma_{11} + \sigma_{JJ} - 2\sigma_{1J})^{-1/2} \\ &= \tilde{\boldsymbol{X}}_{ijt}^T \boldsymbol{\beta} + \tilde{\epsilon}_{ijt}. \end{aligned} \tag{35.5}$$

An individual chooses alternative j in period t if

$$\tilde{U}_{ijt} > \tilde{U}_{imt} \quad \forall\, j \neq m. \tag{35.6}$$

As an identification restriction, one usually imposes a unit variance for the last alternative expressed in utility differences. Define

$$\tilde{\boldsymbol{\epsilon}}_i = (\tilde{\epsilon}_{i11}, \ldots, \tilde{\epsilon}_{i,J-1,1}, \ldots, \tilde{\epsilon}_{i1T_i}, \ldots, \tilde{\epsilon}_{i,J-1,T_i})^T \sim \text{IIDN}(0, \tilde{\boldsymbol{\Sigma}}) \tag{35.7}$$

where $\tilde{\boldsymbol{\Sigma}}$ is the transformed $\boldsymbol{\Sigma}$ with $\tilde{\sigma}_{J-1,J-1} = 1$, so that (35.4) becomes

$$P_i = \text{P}\left(\bigcap_{m \neq d_{it}} \bigcap_{t=1}^{T_i} \tilde{\epsilon}_{i,d_{it},t} - \tilde{\epsilon}_{imt} > (\tilde{X}_{imt} - \tilde{\boldsymbol{X}}_{i,d_{it},t})^T \boldsymbol{\beta} \right), \tag{35.8}$$

which is a $T_i(J-1)$-variate integral. Note that when the $\tilde{\epsilon}_{ijt}$'s are serially uncorrelated, this probability event can be calculated by the product of T_i integrals

of dimension $J-1$, which is easier to compute but this rules out interesting cases, see the applications below.

Estimation

This section briefly explains how the multinomial multiperiod probit model can be estimated in the classical or Bayesian framework. More details can be found in Geweke et al. (1997).

Classical Estimation

Since we assume independent observations on individuals the likelihood is

$$Pr(\boldsymbol{d} \mid \boldsymbol{X}, \boldsymbol{\beta}, \tilde{\boldsymbol{\Sigma}}) = \prod_{i=1}^{I} P_i \tag{35.9}$$

where $\boldsymbol{d} = (\boldsymbol{d}_1, \ldots, \boldsymbol{d}_I)$ and $\boldsymbol{X}$ denotes all the observations on the explanatory variables. Evaluation of this likelihood is infeasible for reasonable values of T_i and J. Classical maximum likelihood estimation methods are usually, except in some trivial cases, based on numerical search algorithms that require many times the evaluation of the likelihood function and are therefore not suitable for this model. For more information on classical estimation, see Hajivassiliou and Ruud (1994), Gouriéroux and Monfort (1997) and Hajivassiliou and Mc Fadden (1998).

Alternative estimation methods are based on simulations of the choice probabilities. The simulated maximum likelihood (SML) method maximizes the simulated likelihood which is obtained by substituting the simulated choice probabilities in (35.9). The method of simulated moments is a simulation based substitute for the generalized method of moments. For further information on these estimation methods we refer to Gouriéroux and Monfort (1997).

Bayesian Inference

The posterior density is

$$\varphi(\boldsymbol{\beta}, \tilde{\boldsymbol{\Sigma}} \mid \boldsymbol{d}, \boldsymbol{X}) \propto Pr(\boldsymbol{d} \mid \boldsymbol{X}, \boldsymbol{\beta}, \tilde{\boldsymbol{\Sigma}})\, \varphi(\boldsymbol{\beta}, \tilde{\boldsymbol{\Sigma}}) \tag{35.10}$$

where $\varphi(\boldsymbol{\beta}, \tilde{\boldsymbol{\Sigma}})$ is the prior density. This does not solve the problem of evaluating a high dimensional integral in the likelihood and it remains hard to compute posterior means for example. Data augmentation, see for example Tanner and Wong (1987), provides a solution because this technique allows to set up a Gibbs sampling scheme using distributions that are easy to draw from. The idea is to augment the parameter

vector with $\tilde{\boldsymbol{U}}$, the latent utilities, so that the posterior density in (35.10) changes to

$$\varphi(\boldsymbol{\beta}, \tilde{\boldsymbol{\Sigma}}, \tilde{\boldsymbol{U}} \mid \boldsymbol{d}, \boldsymbol{X}) \propto Pr(\boldsymbol{d} \mid \boldsymbol{X}, \boldsymbol{\beta}, \tilde{\boldsymbol{\Sigma}}, \tilde{\boldsymbol{U}})\, f(\tilde{\boldsymbol{U}} \mid \boldsymbol{\beta}, \tilde{\boldsymbol{\Sigma}})\, \varphi(\boldsymbol{\beta}, \tilde{\boldsymbol{\Sigma}}) \quad (35.11)$$

implying three blocks in the Gibbs sampler: $\varphi(\boldsymbol{\beta} \mid \tilde{\boldsymbol{\Sigma}}, \tilde{\boldsymbol{U}}, \boldsymbol{d}, \boldsymbol{X})$, $\varphi(\tilde{\boldsymbol{\Sigma}} \mid \boldsymbol{\beta}, \tilde{\boldsymbol{U}}, \boldsymbol{d}, \boldsymbol{X})$ and $\varphi(\tilde{\boldsymbol{U}} \mid \boldsymbol{\beta}, \tilde{\boldsymbol{\Sigma}}, \boldsymbol{d}, \boldsymbol{X})$. For more details on the Gibbs sampler we refer to Chap. II.3 and Chap. III.11. For the first two blocks, the model in (35.5) is the conventional regression model since the utilities, once simulated, are observed. For the last block, remark that $Pr(\boldsymbol{d} \mid \boldsymbol{X}, \boldsymbol{\beta}, \tilde{\boldsymbol{\Sigma}}, \tilde{\boldsymbol{U}})$ is an indicator function since $\tilde{\boldsymbol{U}}$ is consistent with $\boldsymbol{d}$ or not.

Applications

It is possible to extend the model in (35.5) in various ways, such as alternative specific $\boldsymbol{\beta}$'s, individual heterogeneity or a dynamic specification.

Paap and Franses (2000) propose a dynamic specification

$$\begin{aligned}\Delta\tilde{\boldsymbol{U}}_{it} = {} & \Delta\tilde{\boldsymbol{X}}_{it}(\boldsymbol{\alpha} + \boldsymbol{\alpha}_i) + (\boldsymbol{\Pi} - \boldsymbol{I}_{J-1})\left(\tilde{\boldsymbol{U}}_{i,t-1} - \tilde{\boldsymbol{X}}_{i,t-1}(\boldsymbol{\beta} + \boldsymbol{\beta}_i)\right) \\ & + \eta_{it} \end{aligned} \quad (35.12)$$

where $\tilde{\boldsymbol{U}}_{it}$ is the $(J-1)$-dimensional vector of utilities of individual i, $\Delta\tilde{\boldsymbol{U}}_{it} = \tilde{\boldsymbol{U}}_{it} - \tilde{\boldsymbol{U}}_{i,t-1}$, $\tilde{\boldsymbol{X}}_{i,t-1}$ and $\Delta\tilde{\boldsymbol{X}}_{it}$ are matrices of dimension $(J-1) \times k$ for the explanatory variables, $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ are k-dimensional parameter vectors, $\boldsymbol{\Pi}$ is a $(J-1) \times (J-1)$ parameter matrix with eigenvalues inside the unit circle, $\boldsymbol{\eta}_{it} \sim N(0, \tilde{\boldsymbol{\Sigma}})$, and $\boldsymbol{\alpha}_i$ and $\boldsymbol{\beta}_i$ are random individual effects with the same dimension as $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$. These individual heterogeneity effects are assumed to be normally distributed: $\boldsymbol{\alpha}_i \sim N(0, \boldsymbol{\Sigma}_{\boldsymbol{\alpha}})$ and $\boldsymbol{\beta}_i \sim N(0, \boldsymbol{\Sigma}_{\boldsymbol{\beta}})$. The specification in (35.12) is a vector error-correction model where the parameters $\boldsymbol{\alpha} + \boldsymbol{\alpha}_i$ and $\boldsymbol{\beta} + \boldsymbol{\beta}_i$ measure respectively the short-run and long-run effects. The parameters in $\boldsymbol{\Pi}$ determine the speed at which deviations from the long-run relationship are adjusted.

The model parameters are $\boldsymbol{\beta}, \boldsymbol{\alpha}, \tilde{\boldsymbol{\Sigma}}, \boldsymbol{\alpha}_i, \boldsymbol{\beta}_i, \boldsymbol{\Sigma}_{\boldsymbol{\beta}}, \boldsymbol{\Sigma}_{\boldsymbol{\alpha}}$ and $\boldsymbol{\Pi}$ and are augmented by the latent utilities $\tilde{U}_{it}$. Bayesian inference may be done by Gibbs sampling as described in the estimation part above. Table 35.1 describes for each of the nine blocks which posterior distribution is used. For example, $\boldsymbol{\beta}$ has a conditional (on all other parameters) posterior density that is normal.

As an illustration we reproduce the results of Paap and Franses (2000), who provided their Gauss code (which we slightly modified). They use optical scanner data on purchases of four brands of saltine crackers. Chintagunta and Honore (1996) use the same data set to estimate a static multinomial probit model. The data set contains all purchases (choices) of crackers of 136 households over a period of two years, yielding 3,292 observations. Variables such as prices of the brands and whether there was a display and/or newspaper feature of the considered brands

Table 35.1 Summary of conditional posteriors for (35.12)

Parameter	Conditional posterior
$\boldsymbol{\beta}$, $\boldsymbol{\beta}_i$, $\boldsymbol{\alpha}$, $\boldsymbol{\alpha}_i$	Multivariate normal distributions
$\tilde{\boldsymbol{\Sigma}}$, $\boldsymbol{\Sigma}_{\boldsymbol{\alpha}}$, $\boldsymbol{\Sigma}_{\boldsymbol{\beta}}$	Inverted Wishart distributions
$\boldsymbol{\Pi}$	Matrix normal distribution
$\tilde{\boldsymbol{U}}_{it}$	Truncated multivariate normal

Table 35.2 Means of X_{it} variables in (35.12)

	Sunshine	Keebler	Nabisco	Private label
Market share	0.07	0.07	0.54	0.32
Display	0.13	0.11	0.34	0.10
Feature	0.04	0.04	0.09	0.05
Price	0.96	1.13	1.08	0.68

Table 35.3 Posterior moments of $\boldsymbol{\beta}$ and $\boldsymbol{\alpha}$ in (35.12)

	β Parameter		*α Parameter*			*Intercepts*	
	mean	st. dev.	mean	st. dev.		mean	st. dev.
Display	0.307	(0.136)	0.102	(0.076)	Sunshine	−0.071	(0.253)
Feature	0.353	(0.244)	0.234	(0.090)	Keebler	0.512	(0.212)
Price	−1.711	(0.426)	−2.226	(0.344)	Nabisco	1.579	(0.354)

at the time of purchase are also observed and used as the explanatory variables forming $\boldsymbol{X}_{ijt}$ (and then transformed into $\tilde{\boldsymbol{X}}_{ijt}$). Table 35.2 gives the means of these variables. Display and Feature are dummy variables, e.g. Sunshine was displayed 13 per cent and was featured 4 per cent of the purchase occasions. The average market shares reflect the observed individual choices, with e.g. 7 per cent of the choices on Sunshine.

Table 35.3 shows posterior means and standard deviations for the $\boldsymbol{\alpha}$ and $\boldsymbol{\beta}$ parameters. They are computed from 50,000 draws after dropping 20,000 initial draws. The prior on $\tilde{\boldsymbol{\Sigma}}$ is inverted Wishart, denoted by $IW(\boldsymbol{S}, \nu)$, with $\nu = 10$ and $\boldsymbol{S}$ chosen such that $E(\tilde{\boldsymbol{\Sigma}}) = \boldsymbol{I}_3$. Note that Paap and Franses (2000) use a prior such that $E(\tilde{\boldsymbol{\Sigma}}^{-1}) = \boldsymbol{I}_3$. For the other parameters we put uninformative priors. As expected, Display and Feature have positive effects on the choice probabilities and price has a negative effect. This holds both in the short run and the long run. With respect to the private label (which serves as reference category), the posterior means of the intercepts are positive except for the first label whose intercept is imprecisely estimated.

Table 35.4 gives the posterior means and standard deviations of $\tilde{\boldsymbol{\Sigma}}$, $\boldsymbol{\Pi}$, $\tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\beta}}$ and $\tilde{\boldsymbol{\Sigma}}_{\boldsymbol{\alpha}}$. Note that the reported last element of $\tilde{\boldsymbol{\Sigma}}$ is equal to 1 in order to identify the model. This is done, after running the Gibbs sampler with $\tilde{\boldsymbol{\Sigma}}$ unrestricted, by dividing the variance related parameter draws by $\tilde{\sigma}_{J-1,J-1}$. The other parameter draws are divided by the square root of the same quantity. McCulloch et al. (2000) propose an alternative approach where $\tilde{\boldsymbol{\Sigma}}_{J-1,J-1}$ is fixed to 1 by construction, i.e. a

Table 35.4 Posterior means and standard deviations of $\tilde{\boldsymbol{\Sigma}}$, $\boldsymbol{\Pi}$, $\tilde{\boldsymbol{\Sigma}}_\beta$ and $\tilde{\boldsymbol{\Sigma}}_\alpha$ in (35.12)

$\tilde{\boldsymbol{\Sigma}}$

0.563	−0.102	0.433
(0.179)	(0.096)	(0.087)
	0.241	0.293
	(0.119)	(0.069)
		1

$\boldsymbol{\Pi}$

0.474	0.213	0.054
(0.103)	(0.134)	(0.066)
0.440	0.685	−0.196
(0.067)	(0.081)	(0.049)
−0.099	−0.161	0.421
(0.091)	(0.138)	(0.087)

$\tilde{\boldsymbol{\Sigma}}_\beta$

0.431	−0.267	0.335	−0.176	−0.100	0.087
(0.201)	(0.250)	(0.463)	(0.247)	(0.209)	(0.401)
	1.053	0.281	0.412	0.306	0.721
	(0.603)	(0.774)	(0.352)	(0.372)	(0.719)
		5.445	−1.310	−1.010	0.539
		(2.268)	(0.999)	(0.853)	(1.120)
			1.919	1.225	1.950
			(0.672)	(0.560)	(0.664)
				1.496	1.564
				(0.879)	(0.816)
					4.915
					(1.319)

$\tilde{\boldsymbol{\Sigma}}_\alpha$

0.207	−0.023	−0.004
(0.091)	(0.075)	(0.220)
	0.382	0.217
	(0.144)	(0.366)
		6.672
		(2.453)

fully identified parameter approach. They write

$$\tilde{\boldsymbol{\Sigma}} = \begin{pmatrix} \boldsymbol{\Phi} + \boldsymbol{\gamma}\boldsymbol{\gamma}^T & \boldsymbol{\gamma} \\ \boldsymbol{\gamma}^T & 1 \end{pmatrix} \tag{35.13}$$

and show that the conditional posterior of $\boldsymbol{\gamma}$ is normal and that of $\boldsymbol{\Phi}$ is Wishart, so that draws of $\tilde{\boldsymbol{\Sigma}}$ are easily obtained. This approach is of particular interest when a sufficiently informative prior on $\tilde{\boldsymbol{\Sigma}}$ is used. A drawback of this approach is that the Gibbs sampler has higher autocorrelation and that it is more sensitive to initial conditions.

The relatively large posterior means of the diagonal elements of $\boldsymbol{\Pi}$ show that there is persistence in brand choice. The matrices $\tilde{\boldsymbol{\Sigma}}_\beta$ and $\tilde{\Sigma}_\alpha$ measure the unobserved heterogeneity. There seems to be substantial heterogeneity across the individuals, especially for the price of the products (see the third diagonal elements of both matrices). The last three elements in $\tilde{\boldsymbol{\Sigma}}_\beta$ are related to the intercepts.

The multinomial probit model is frequently used for marketing purposes. For example, Allenby and Rossi (1999) use ketchup purchase data to emphasize the importance of a detailed understanding of the distribution of consumer heterogeneity and identification of preferences at the customer level.

A related model is the multivariate probit model which relaxes the assumption that choices are mutually exclusive, as in the multinomial model discussed before. In that case, $\boldsymbol{d}_i$ may contain several 1 s. Chib and Greenberg (1998) discuss classical and Bayesian inference for this model. They also provide examples on voting behavior, on health effects of air pollution and on labour force participation.

35.2.2 Mixed Multinomial Logit

Definition

The multinomial logit model is defined as in (35.1), except that the random shock ϵ_{ijt} is extreme value (or Gumbel) distributed. This gives rise to the independence from irrelevant alternatives (IIA) property which essentially means that $Cov(U_{ijt}, U_{ikt}) = 0 \ \forall j, \ \forall k$. Like the probit model, the mixed multinomial logit (MMNL) model alleviates this restrictive IIA property by treating the $\boldsymbol{\beta}$ parameter as a random vector with density $f_{\boldsymbol{\theta}}(\boldsymbol{\beta})$. The latter density is called the mixing density and is usually assumed to be a normal, lognormal, triangular or uniform distribution. To make clear why this model does not suffer from the IIA property, consider the following example. Suppose that there is only explanatory variable and that $\beta \sim \mathrm{N}(\bar{\beta}, \bar{\sigma}^2)$. We can then write (35.1) as

$$U_{ijt} = X_{ijt}\bar{\beta} + X_{ijt}\bar{\sigma}z + \epsilon_{ijt} \tag{35.14}$$

$$= X_{ijt}\bar{\beta} + \epsilon^*_{ijt} \tag{35.15}$$

where $z \sim N(0, 1)$, implying that the variance of ϵ^*_{ijt} depends on the explanatory variable and that there is nonzero covariance between utilities for different alternatives.

The mixed logit probability is given by

$$P_i = \int \prod_{t=1}^{T_i} \left(\frac{\mathrm{e}^{X_{ijt}^T \boldsymbol{\beta}}}{\sum_{j=1}^{J} \mathrm{e}^{X_{ijt}^T \boldsymbol{\beta}}} \right) f_{\boldsymbol{\theta}}(\boldsymbol{\beta}) \mathrm{d}\boldsymbol{\beta} \tag{35.16}$$

where the term between brackets is the logistic distribution arising from the difference between two extreme value distributions. The model parameter is $\boldsymbol{\theta}$. Note that one may want to keep elements of $\boldsymbol{\beta}$ fixed as in the usual logit model. One usually keeps random the elements of $\boldsymbol{\beta}$ corresponding to the variables that are believed to create correlation between alternatives. The mixed logit model is quite general. McFadden and Train (2000) demonstrate that any random utility model can be approximated to any degree of accuracy by a mixed logit with appropriate choice of variables and mixing distribution.

Estimation

Classical Estimation

Estimation of the MMNL model can be done by SML or the method of simulated moments or simulated scores. To do this, the logit probability in (35.16) is replaced by its simulated counterpart

$$SP_i = \frac{1}{R}\sum_{r=1}^{R}\prod_{t=1}^{T_i}\left(\frac{e^{X_{ijt}^T\boldsymbol{\beta}^r}}{\sum_{j=1}^{J} e^{X_{ijt}^T\boldsymbol{\beta}^r}}\right) \tag{35.17}$$

where the $\{\boldsymbol{\beta}^r\}_{r=1}^R$ are i.i.d. draws of $f_{\boldsymbol{\theta}}(\boldsymbol{\beta})$. The simulated likelihood is the product of all the individual SP_i's. The simulated log-likelihood can be maximized with respect to $\boldsymbol{\theta}$ using numerical optimization techniques like the Newton-Raphson algorithm. To avoid an erratic behaviour of the simulated objective function for different values of $\boldsymbol{\theta}$, the same sequences of basic random numbers is used to generate the sequence $\{\boldsymbol{\beta}^r\}$ used during all the iterations of the optimizer (this is referred to as the technique of "common random numbers").

According to Gouriéroux and Monfort (1997) the SML estimator is asymptotically equivalent to the ML estimator if T (the total number of observations) and R both tend to infinity and $\sqrt{T}/R \to 0$. In practice, it is sufficient to fix R at a moderate value.

The approximation of an integral like in (35.16) by the use of pseudo-random numbers may be questioned. Bhat (2001) implements an alternative quasi-random SML method which uses quasi-random numbers. Like pseudo-random sequences, quasi-random sequences, such as Halton sequences, are deterministic, but they are more uniformly distributed in the domain of integration than pseudo-random ones.

Bayesian Inference

Let us suppose that the mixing distribution is Gaussian, that is, the vector $\boldsymbol{\beta}$ is normally distributed with mean $\boldsymbol{b}$ and variance matrix $\boldsymbol{W}$. The posterior density for I individuals can be written as

$$\varphi(\boldsymbol{b}, \boldsymbol{W} \mid \boldsymbol{d}, \boldsymbol{X}) \propto Pr(\boldsymbol{d} \mid \boldsymbol{X}, \boldsymbol{b}, \boldsymbol{W})\,\varphi(\boldsymbol{b}, \boldsymbol{W}) \tag{35.18}$$

where $Pr(\boldsymbol{d} \mid \boldsymbol{X}, \boldsymbol{b}, \boldsymbol{W}) = \prod_{i=1}^{I} P_i$ and $\varphi(\boldsymbol{b}, \boldsymbol{W})$ is the prior density on $\boldsymbol{b}$ and $\boldsymbol{W}$. Sampling from (35.18) is difficult because P_i is an integral without a closed form as discussed above. We would like to condition on $\boldsymbol{\beta}$ such that the choice probabilities are easy to calculate. For this purpose we augment the model parameter vector with $\boldsymbol{\beta}$. It is convenient to write $\boldsymbol{\beta}_i$ instead of $\boldsymbol{\beta}$ to interpret the random coefficients as representing heterogeneity among individuals. The $\boldsymbol{\beta}_i$'s are independent and

Table 35.5 Summary of conditional posteriors for MMNL model

Parameter	Conditional posterior or sampling method
$\boldsymbol{b}$	Multivariate normal distribution
$\boldsymbol{W}$	Inverted Wishart distribution
$\boldsymbol{\beta}_I$	Metropolis Hastings algortihm

identically distributed with mixing distribution $f(\cdot \mid \boldsymbol{b}, \boldsymbol{W})$. The posterior can then be written as

$$\varphi(\boldsymbol{b}, \boldsymbol{W}, \boldsymbol{\beta}_I \mid \boldsymbol{d}, \boldsymbol{X}) \propto Pr(\boldsymbol{d} \mid \boldsymbol{X}, \boldsymbol{\beta}_I)\, f(\boldsymbol{\beta}_I \mid \boldsymbol{b}, \boldsymbol{W})\, \varphi(\boldsymbol{b}, \boldsymbol{W}) \tag{35.19}$$

where $\boldsymbol{\beta}_I$ collects the $\boldsymbol{\beta}_i$'s for all the I individuals. Draws from this posterior density can be obtained by using the Gibbs sampler. Table 35.5 summarizes the three blocks of the sampler.

For the first two blocks the conditional posterior densities are known and are easy to sample from. The last block is more difficult. To sample from this density, a Metropolis Hastings (MH) algorithm is set up. Note that only one iteration is necessary such that simulation within the Gibbs sampler is avoided. See Train (2003), Chap. 12, for a detailed description of the MH algorithm for the mixed logit model and for guidelines about how to deal with other mixing densities. More general information on the MH algorithm can be found in Chap. II.3.

Bayesian inference in the mixed logit model is called hierarchical Bayes because of the hierarchy of parameters. At the first level, there is the individual parameters $\boldsymbol{\beta}_i$ which are distributed with mean $\boldsymbol{\beta}$ and variance matrix $\boldsymbol{W}$. The latter are called hyper-parameters, on which we have also prior densities. They form the second level of the hierarchy.

Application

We reproduce the results of McFadden and Train (2000) using their Gauss code available on the web site elsa.berkeley.edu/∼train/software.html. They analyse the demand for alternative vehicles. There are 4,654 respondents who choose among six alternatives (two alternatives run on electricity only). There are 21 explanatory variables among which 4 are considered to have a random effect. The mixing distributions for these random coefficients are independent normal distributions. The model is estimated by SML and uses $R = 250$ replications per observation. Table 35.6 includes partly the estimation results of the MMNL model. We report the estimates and standard errors of the parameters of the normal mixing distributions, but we do not report the estimates of the fixed effect parameters corresponding to the 17 other explanatory variables. For example, the luggage space error component induces greater covariance in the stochastic part of utility for pairs of vehicles with

Table 35.6 SML estimates of MMNL random effect parameters

Variable	*Mean*	*Standard deviation*
Electric vehicle (EV) dummy	−1.032 (0.503)	2.466 (0.720)
Compressed natural gass (CNG) dummy	0.626 (0.167)	1.072 (0.411)
Size	1.435 (0.499)	7.457 (2.043)
Luggage space	1.702 (0.586)	5.998 (1.664)

Robust standard errors within parentheses

greater luggage space. We refer to McFadden and Train (2000) or Brownstone and Train (1999) for more interpretations of the results.

Train (2003) provides more information and pedagogical examples on the mixed multinomial model.

35.3 Stochastic Volatility and Duration Models

Stochastic volatility (SV) models may be used as an alternative to generalized autoregressive conditonal heteroskedastic (GARCH) models as a way to model the time-varying volatility of asset returns. Time series of asset returns feature stylized facts, the most important being volatility clustering, which produces a slowly decreasing positive autocorrelation function of the squared returns, starting at a low value (about 0.15). Another stylized fact is excess kurtosis of the distribution (with respect to the Gaussian distribution). See Bollerslev et al. (1994) for a detailed list of the stylized facts and a survey of GARCH models, Shephard (1996) for a comparative survey of GARCH and SV models, and Ghysels et al. (1996) for a survey of SV models focused on their theoretical foundations and their applications in finance. The first four parts of this section deal with SV models while in subsection 35.3.5 we survey similar models for dynamic duration analysis.

35.3.1 Canonical SV Model

The simplest version of a SV model is given by

$$\begin{aligned} y_t &= \exp(h_t/2)\, u_t, \qquad & u_t &\sim N(0,1), \quad t = 1, \ldots, n, \\ h_t &= \omega + \beta h_{t-1} + \sigma v_t, & v_t &\sim N(0,1), \end{aligned} \tag{35.20}$$

where y_t is a return measured at t, h_t is the unobserved log-volatility of y_t, $\{u_t\}$ and $\{v_t\}$ are mutually independent sequences, $(\omega,\ \beta,\ \sigma)$ are parameters to be estimated, jointly denoted $\boldsymbol{\theta}$. The parameter space is $\mathrm{IR} \times (-1, 1) \times \mathrm{IR}_+$. The restriction on β ensures the strict stationarity of y_t. Estimates of β are typically quite close to 1 (in agreement with the first stylized fact), thus β is a "persistence" parameter of the

volatility. The unconditonal mean of h_t is $\mu = \omega/(1-\beta)$ and the second equation may be parametrized using μ by writing $h_t = \mu + \beta(h_{t-1} - \mu) + \sigma v_t$. Another parametrization removes ω from the second equation while writing the first as $y_t = \tau \exp(h_t/2)\, u_t$ where $\tau = \exp(\omega/2)$. These different parametrizations are in one-to-one correspondance. Which one to choose is mainly a matter of convenience and numerical efficiency of estimation algorithms.

For further use, let $\boldsymbol{y}$ and $\boldsymbol{h}$ denote the $n \times 1$ vectors of observed returns and unobserved log-volatilities, respectively.

35.3.2 Estimation

Estimation of the parameters of the canonical SV model may be done by the maximum likelihood (ML) method or by Bayesian inference. Other methods have been used but they are not considered here. We refer to Ghysels et al. (1996), Sect. 5, for a review. ML and, in principle, Bayesian estimation require to compute the likelihood function of an observed sample, which is a difficult task. Indeed, the density of y given $\boldsymbol{\theta}$ and an initial condition h_0 (not explicitly written in the following expressions) requires to compute a multiple integral which has a dimension equal to the sample size:

$$f(\boldsymbol{y}|\boldsymbol{\theta}) = \int f(\boldsymbol{y}, \boldsymbol{h}|\boldsymbol{\theta})\, d\boldsymbol{h} \tag{35.21}$$

$$= \int f(\boldsymbol{y}|\boldsymbol{h}, \boldsymbol{\theta})\, f(\boldsymbol{h}|\boldsymbol{\theta})\, d\boldsymbol{h} \tag{35.22}$$

$$= \int \prod_{t=1}^{n} f(y_t, h_t|\boldsymbol{Y}_{t-1}, \boldsymbol{H}_{t-1}, \boldsymbol{\theta})\, d\boldsymbol{h} \tag{35.23}$$

where $\boldsymbol{Y}_t = \{y_i\}_{i=1}^{t}$ and $\boldsymbol{H}_t = \{h_i\}_{i=0}^{t}$. For model (35.20), this is

$$\int \prod_{t=1}^{n} f_N(y_t|0, e^{h_t})\, f_N(h_t|\omega + \beta h_{t-1}, \sigma^2)\, d\boldsymbol{h}, \tag{35.24}$$

where $f_N(x|\mu, \sigma^2)$ denotes the normal density function of x, with parameters μ and σ^2. An analytical solution to the integration problem is not available. Even a term by term numerical approximation by a quadrature rule is precluded: the integral of $N(0, \exp(h_n)) \times N(\omega + \beta h_{n-1}, \sigma^2)$ with respect to h_n depends on h_{n-1}, and has to be carried over in the previous product, and so on until h_1. This would result in an explosion of the number of function evaluations. Simulation methods are therefore used.

Two methods directly approximate (35.23): efficient importance sampling (EIS), and Monte Carlo maximum likelihood (MCML). Another approach, which can only

be used for Bayesian inference, works with $f(\boldsymbol{y}, \boldsymbol{h}|\boldsymbol{\theta})$ as data density, and produces a posterior joint density for $\boldsymbol{\theta}, \boldsymbol{h}$ given $\boldsymbol{y}$. The posterior density is simulated by a Monte Carlo Markov chain (MCMC) algorithm, which produces simulated values of $\boldsymbol{\theta}$ and $\boldsymbol{h}$. Posterior moments and marginal densities of $\boldsymbol{\theta}$ are then estimated by their simulated sample counterparts. We pursue by describing each method.

EIS (Liesenfeld and Richard 2003)

A look at (35.24) suggests to sample R sequences $\{h_t^r \sim N(\omega + \beta h_{t-1}, \sigma^2)\}_{t=1}^n$, $r = 1 \dots R$, and to approximate it by $(1/R) \sum_{r=1}^R \prod_{t=1}^n N(0, \exp(h_t^r))$. This direct method proves to be inefficient. Intuitively, the sampled sequences of h_t are not linked to the observations y_t. To improve upon this, the integral (35.23), which is the convenient expression to present EIS, is expressed as

$$\int \prod_{t=1}^{n} \frac{f(y_t, h_t|\boldsymbol{Y}_{t-1}, \boldsymbol{H}_{t-1}, \boldsymbol{\theta})}{m(h_t|\boldsymbol{H}_{t-1}, \boldsymbol{\phi}_t)} m(h_t|\boldsymbol{H}_{t-1}, \boldsymbol{\phi}_t)\, d\boldsymbol{h}, \tag{35.25}$$

where $\{m(h_t|\boldsymbol{H}_{t-1}, \boldsymbol{\phi}_t)\}_{t=1}^n$ is a sequence of importance density functions, indexed by parameters $\{\boldsymbol{\phi}_t\}$. These importance functions serve to generate R random draws $\{h_t^1, h_t^2 \dots h_t^R\}_{t=1}^n$, such that the integral is approximated by the sample mean

$$\frac{1}{R} \sum_{r=1}^{R} \prod_{t=1}^{n} \frac{f(y_t, h_t^r|\boldsymbol{Y}_{t-1}, \boldsymbol{H}_{t-1}^r, \boldsymbol{\theta})}{m(h_t^r|\boldsymbol{H}_{t-1}^r, \boldsymbol{\phi}_t)}. \tag{35.26}$$

The essential point is to choose the form of $m()$ and its auxiliary parameters $\boldsymbol{\phi}_t$ so as to secure a good match between the product of $m(h_t|\boldsymbol{H}_{t-1}, \boldsymbol{\phi}_t)$ and the product of $f(y_t, h_t|\boldsymbol{Y}_{t-1}, \boldsymbol{H}_{t-1}, \boldsymbol{\theta})$ *viewed as a function of* $\boldsymbol{h}$. A relevant good match criterion is provided by a choice of $\{\boldsymbol{\phi}_t\}$, for a given family of densities for $m()$, based on the minimization of the Monte Carlo variance of the mean (35.26). The choice of $\{\boldsymbol{\phi}_t\}$ is explained below, after the choice of $m()$.

A convenient choice for $m()$ is the Gaussian family of distributions. A Gaussian approximation to $f()$, as a function of h_t, given y_t and h_{t-1}, turns out to be efficient. It can be expressed as proportional to $\exp(\phi_{1,t} h_t + \phi_{2,t} h_t^2)$, where $(\phi_{1,t}, \phi_{2,t}) = \boldsymbol{\phi}_t$, the auxiliary parameters. It is convenient to multiply it with $\exp[-0.5\sigma^{-2}(-2m_t h_t + h_t^2 + m_t^2)]$, where $m_t = \omega + \beta h_{t-1}$, which comes from the $N(m_t, \sigma^2)$ term included in $f(y_t, h_t|\boldsymbol{Y}_{t-1}, \boldsymbol{H}_{t-1}, \boldsymbol{\theta})$. The product of these two exponential functions can be expressed as a Gaussian density $N(\mu_t, \sigma_t^2)$, where

$$\mu_t = \sigma_t^2(m_t/\sigma^2 + \phi_{1,t}), \quad \sigma_t^2 = \sigma^2/(1 - 2\sigma^2\phi_{2,t}). \tag{35.27}$$

The choice of the auxiliary parameters can be split into n separate problems, one for each t. It amounts to minimize the sum of squared deviations between

$\ln f(y_t|\boldsymbol{Y}_{t-1}, \boldsymbol{H}_t^r, \boldsymbol{\theta})$ plus a correction term, see (35.28), and $\phi_{0,t}+\phi_{1,t}h_t^r+\phi_{2,t}(h_t^r)^2$ where $\phi_{0,t}$ is an auxiliary intercept term. This problem is easily solved by ordinary least squares. See Liesenfeld and Richard (2003) for a detailed explanation.

Let us summarize the core of the EIS algorithm in three steps (for given $\boldsymbol{\theta}$ and $\boldsymbol{y}$):
Step 1: Generate R trajectories $\{h_t^r\}$ using the "natural" samplers $\{N(m_t, \sigma^2)\}$.
Step 2: For each t (starting from $t = n$ and ending at $t = 1$), using the R observations generated in the previous step, estimate by OLS the regression

$$-\frac{1}{2}[h_t^r + y_t^2 e^{-h_t^r} + \left(\frac{\mu_{t+1}^r}{\sigma_{t+1}^r}\right)^2 - \left(\frac{m_{t+1}^r}{\sigma}\right)^2] = \phi_{0,t} + \phi_{1,t}h_t^r + \phi_{2,t}(h_t^r)^2 + \epsilon_t^r \quad (35.28)$$

where ϵ_t^r is an error term. For $t = n$, the dependent variable does not include the last two terms in the square brackets. The superscript r on μ_{t+1}, σ_{t+1} and m_{t+1} indicates that these quantities are evaluated using the r-th trajectory.
Step 3: Generate R trajectories $\{h_t^r\}$ using the efficient samplers $\{N(\mu_t, \sigma_t^2)\}$ and finally compute (35.26).

Steps 1 to 3 should be iterated about five times to improve the efficiency of the approximations. This is done by replacing the natural sampler in step 1 by the importance functions built in the previous iteration. It is also possible to start step 1 of the first iteration with a more efficient sampler than the natural one. This is achieved by multiplying the natural sampler by a normal approximation to $f(y_t|h_t, h_{t-1}, \boldsymbol{\theta}) \propto \exp\{-0.5[y_t^2 \exp(-h_t) + h_t]\}$. The normal approximation is based on a second-order Taylor series expansion of the argument of the exponential in the previous expression around $h_t = 0$. In this way, the initial importance sampler links y_t and h_t. This enables one to reduce to three (instead of five) the number of iterations over the three steps. In practical implementations, R can be fixed to 50. When computing (35.26) for different values of $\boldsymbol{\theta}$, such as in a numerical optimizer, it is important to use common random numbers to generate the set of R trajectories $\{h_t^r\}$ that serve in the computations.

It is also easy to compute by EIS filtered estimates of functions of h_t, such as the conditional standard deviation $\exp(h_t/2)$, conditional on the past returns (but not on the lagged unobserved h_t), given a value of $\boldsymbol{\theta}$ (such as the ML estimate). Diagnostics on the model specification are then obtained as a byproduct: if the model is correctly specified, y_t divided by the filtered estimates of $\exp(h_t/2)$ is a residual that has zero mean, unit variance, and is serially uncorrelated (this also holds for the squared residual).

Richard and Zhang (2007) contains a general presentation of EIS and its properties.

MCML (Durbin and Koopman 1997)

The likelihood to be computed at $\boldsymbol{y}$ (the data) and any given $\boldsymbol{\theta}$ is equal to $f(\boldsymbol{y}|\boldsymbol{\theta})$ and is conveniently expressed as (35.22) for this method. This quantity is approximated by importance sampling with an importance function defined from an approximating

model. The latter is obtained by using the state space representation of the canonical SV model (parametrized with τ):

$$\ln y_t^2 = \ln \tau^2 + h_t + \epsilon_t, \tag{35.29}$$

$$h_t = \beta h_{t-1} + \sigma v_t. \tag{35.30}$$

In the canonical SV model, $\epsilon_t = \ln u_t^2$ is distributed as the logarithm of a $\chi^2(1)$ random variable. However the approximating model replaces this with a Gaussian distribution (defined below), keeping the state equation unchanged. Therefore, the whole machinery of the Kalman filter is applicable to the approximating model, which is a Gaussian linear state space model. If we denote by $g(\boldsymbol{h}|y, \boldsymbol{\theta})$ the importance function that serves to simulate $\boldsymbol{h}$ (see below), we have

$$f(\boldsymbol{y}|\boldsymbol{\theta}) = \int \frac{f(\boldsymbol{y}|\boldsymbol{h}, \boldsymbol{\theta}) f(\boldsymbol{h}|\boldsymbol{\theta})}{g(\boldsymbol{h}|\boldsymbol{y}, \boldsymbol{\theta})} g(\boldsymbol{h}|\boldsymbol{y}, \boldsymbol{\theta})\, d\boldsymbol{h} \tag{35.31}$$

$$= g(\boldsymbol{y}|\boldsymbol{\theta}) \int \frac{f(\boldsymbol{y}|\boldsymbol{h}, \boldsymbol{\theta})}{g(\boldsymbol{y}|\boldsymbol{h}, \boldsymbol{\theta})} g(\boldsymbol{h}|\boldsymbol{y}, \boldsymbol{\theta})\, d\boldsymbol{h}, \tag{35.32}$$

where the second equality results from $g(\boldsymbol{h}|\boldsymbol{y}, \boldsymbol{\theta}) g(\boldsymbol{y}|\boldsymbol{\theta}) = g(\boldsymbol{y}|\boldsymbol{h}, \boldsymbol{\theta}) g(\boldsymbol{h}|\boldsymbol{\theta})$ and $g(\boldsymbol{h}|\boldsymbol{\theta}) = f(\boldsymbol{h}|\boldsymbol{\theta})$. All the densities $g(.)$ and $g(.|.)$ are defined from the approximating Gaussian model. In particular, $g(\boldsymbol{y}|\boldsymbol{\theta})$ is the likelihood function of the Gaussian linear state space model and is easy to compute by the Kalman filter (see the appendix to Sandman and Koopman (1998) for all details). Likewise, $g(\boldsymbol{y}|\boldsymbol{h}, \boldsymbol{\theta})$ obtains from the Gaussian densities $g(\ln y_t^2|h_t, \boldsymbol{\theta})$ resulting from (35.29) with $\epsilon_t \sim N(a_t, s_t^2)$ where a_t and s_t^2 are chosen so that $g(\boldsymbol{y}|\boldsymbol{h}, \boldsymbol{\theta})$ is as close as possible to $f(\boldsymbol{y}|\boldsymbol{h}, \boldsymbol{\theta})$. The parameters a_t and s_t^2 are chosen so that $\ln g(\ln y_t^2|\hat{h}_t, \boldsymbol{\theta})$ and $\ln f(\ln y_t^2|\hat{h}_t, \boldsymbol{\theta})$ have equal first and second derivatives, where $\hat{h}_t$ is the smoothed value of h_t provided by the Kalman filter applied to the approximating model. Remark that this is a different criterion from that used in EIS. Finally, $g(\boldsymbol{h}|\boldsymbol{y}, \boldsymbol{\theta})$ can be simulated with the Gaussian simulation smoother of de Jong and Shephard (1995).

In brief, the likelihood function is approximated by

$$g(y|\boldsymbol{\theta}) \frac{1}{R} \sum_{r=1}^{R} \frac{f(\boldsymbol{y}|\boldsymbol{h}^r, \boldsymbol{\theta})}{g(\boldsymbol{y}|\boldsymbol{h}^r, \boldsymbol{\theta})} \tag{35.33}$$

where $\boldsymbol{h}^r = \{h_t^r\}_{t=1}^T$ is simulated independently R times with the importance sampler and $g(\boldsymbol{y}|\boldsymbol{\theta})$ is computed by the Kalman filter. Equation (35.32) and (35.33) shows that importance sampling serves to evaluate the departure of the actual likelihood from the likelihood of the approximating model. R is fixed to 250 in practice.

For SML estimation, the approximation in (35.33) is transformed in logarithm. This induces a bias since the expectation of the log of the sample mean is not the log of the corresponding integral in (35.32). The bias is corrected by adding $s_w^2/(2R\bar{w})$ to the log of (35.33), where s_w^2 is the sample variance of the ratios $w^r = f(\boldsymbol{y}|\boldsymbol{h}^r, \boldsymbol{\theta})/g(\boldsymbol{y}|\boldsymbol{h}^r, \boldsymbol{\theta})$ and $\bar{w}$ is the sample mean of the same ratios, i.e. $\bar{w}$ is the sample mean appearing in (35.33). Moreover, Durbin and Koopman (1997) use antithetic and control variables to improve the efficiency of the estimator of the log-likelihood function.

Durbin and Koopman (2000) present several generalizations of MCML (e.g. the case where the state variable in non-Gaussian) and develop analogous methods for Bayesian inference.

MCMC (Kim et al. 1998)

We present briefly the "Mixture Sampler", one of the three algorithms added by Kim et al. (1998) to the six algorithms already in the literature at that time (see their paper for references). They approximate the density of $\epsilon_t = \ln u_t^2$ by a finite mixture of seven Gaussian densities, such that in particular the first four moments of both densities are equal. The approximating density can be written as

$$f_a(\epsilon_t) = \sum_{i=1}^{7} \Pr[s_t = i] f(\epsilon_t|s_t = i) = \sum_{i=1}^{7} \Pr[s_t = i] f_N(\epsilon_t|b_i - 1.2704, c_i^2) \tag{35.34}$$

where s_t is a discrete random variable, while $\Pr[s_t = i]$, b_i and c_i are known constants (independent of t). The constant -1.2704 is the expected value of a $\ln \chi^2(1)$ variable.

The crux of the algorithm is to add $\boldsymbol{s} = \{s_t\}_{t=1}^n$ to $\boldsymbol{\theta}$ and $\boldsymbol{h}$ in the MCMC sampling space. This makes it possible to sample $\boldsymbol{h}|\boldsymbol{s}, \boldsymbol{\theta}, \boldsymbol{y}$, $\boldsymbol{s}|\boldsymbol{h}, \boldsymbol{y}$ and $\boldsymbol{\theta}|\boldsymbol{h}, \boldsymbol{y}$ within a Gibbs sampling algorithm. Remark that $\boldsymbol{s}$ and $\boldsymbol{\theta}$ are independent given $\boldsymbol{h}$ and $\boldsymbol{y}$. Moreover, $\boldsymbol{h}$ can be sampled entirely as a vector. The intuition behind this property is that, once $\boldsymbol{s}$ is known, the relevant term of the mixture (35.34) is known for each observation, and since this is a Gaussian density, the whole apparatus of the Kalman filter can be used. Actually, this a bit more involved since the relevant Gaussian density depends on t, but an augmented Kalman filter is available for this case.

Sampling $\boldsymbol{h}$ as one block is a big progress over previous algorithms, such as in Jacquier et al. (1994), where each element h_t is sampled individually given the other elements of $\boldsymbol{h}$ (plus $\boldsymbol{\theta}$ and $\boldsymbol{y}$). The slow convergence of such algorithms is due to the high correlations between the elements of $\boldsymbol{h}$.

Kim et al. (1998) write the model in state space form, using μ rather than ω or τ as a parameter, i.e.

$$\ln y_t^2 = h_t + \epsilon_t, \tag{35.35}$$

$$h_t - \mu = \beta(h_{t-1} - \mu) + \sigma v_t. \tag{35.36}$$

The "Mixture Sampler" algorithm is summarized in Table 35.7. Notice that once $\boldsymbol{\theta}$ has been sampled, it is easy to transform the draws of μ into equivalent draws of ω or τ by using the relationships between these parameters. Since inference is Bayesian, prior densities must be specified. For σ^2, an inverted gamma prior density is convenient since the conditional posterior is in the same class and easy to simulate. For β, any prior can be used since the conditional posterior is approximated and rejection sampling is used. A beta prior density is advocated by Kim et al. (1998). For μ, a Gaussian or uninformative prior results in a Gaussian conditional posterior.

35.3.3 Application

For illustration, estimates of the canonical SV model parameters are reported in Table 35.8 for a series of 6,107 centred daily returns of the Standard and Poors 500 (SP500) composite price index (period: 02/01/80-30/05/03, source: Datastream). Returns are computed as 100 times the log of the price ratios. The sample mean and standard deviation are equal to 0.03618 and 1.0603, respectively.

We used computer codes provided by the authors cited above. For EIS, we received the code from R. Liesenfeld, for MCML and MCMC we downloaded them from the web site staff.feweb.vu.nl/koopman/sv.

For SML estimation by EIS or MCML, identical initial values ($\beta = 0.96$, $\sigma = 0.15$, $\omega = 0.02$ or $\tau = 0.01$) and optimization algorithms (BFGS) are used, but in different programming environments. Therefore, the computing times are not fully comparable, although a careful rule of thumb is that Ox is two to three times

Table 35.7 Summary of "Mixture Sampler" algorithm

Parameter	Conditional posterior or sampling method
$\boldsymbol{h}$	Gaussian simulation smoother
$\boldsymbol{s}$	Univariate discrete distribution for each s_t
σ^2	Inverted gamma distribution
β	Rejection or Metropolis-Hastings sampler
μ	Normal distribution

Table 35.8 ML and Bayesian estimates of SV model (35.20)

	EIS (ω)	MCML (τ)	MCMC (τ)
ω or τ	−0.00524 (0.00227)	0.863 (0.0469)	0.864 (0.0494)
β	0.982 (0.00385)	0.982 (0.00389)	0.983 (0.00382)
σ	0.149 (0.0138)	0.147 (0.0135)	0.143 (0.0139)
llf	−8023.98	−8023.80	
time	2.36 min.	7.56 min.	6.23 min.
code	Gauss	Ox	Ox

llf: value of log-likelihood function at the reported estimate; EIS, MCML, and MCMC are defined in subsection 35.3.2

faster than Gauss (see Cribari-Neto 1997). Reported execution times imply that EIS appears to be at least six times faster than MCML. This is a reversal of a result reported by Sandman and Koopman (1998) (p 289), but they compared MCML with a precursor of EIS implemented by Danielson (1994). More importantly, the two methods deliver quasi-identical results.

MCMC results are based on 18,000 draws after dropping 2,000 initial draws. The posterior means and standard deviations are also quite close to the ML results. The posterior density of σ (computed by kernel estimation) is shown in Fig. 35.1 together with the large sample normal approximation to the density of the ML estimator using the EIS results. The execution time for MCMC is difficult to compare with the other methods since it depends on the number of Monte Carlo draws. It is however quite competitive since reliable results are obtained in no more time than MCML in this example.

35.3.4 Extensions of the Canonical SV Model

The canonical model presented in (35.20) is too restrictive to fit the excess kurtosis of many return series. Typically, the residuals of the model reveal that the distribution of u_t has fatter tails than the Gaussian distribution. The assumption of normality is most often replaced by the assumption that $u_t \sim t(0, 1, \nu)$, which denotes Student-t distribution with zero mean, unit variance, and degrees of freedom parameter $\nu > 2$. SML estimates of ν are usually between 5 and 15 for stock and foreign currency returns using daily data. Posterior means are larger because the posterior density of has a long tail to the right.

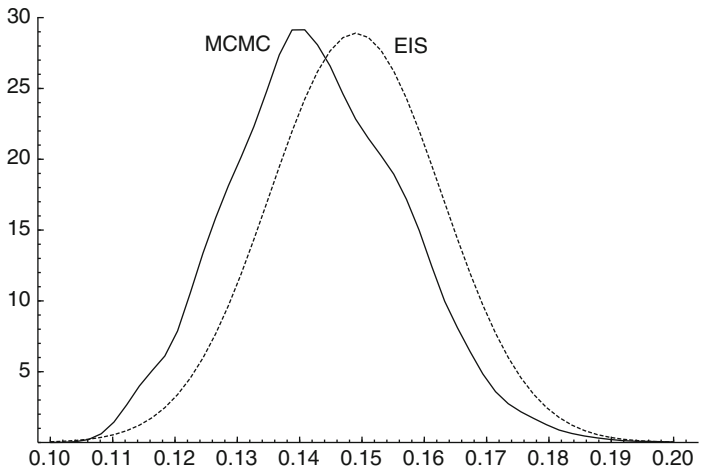


Fig. 35.1 Posterior density of σ and normal density of the MLE

Several other extensions of the simple SV model presented in (35.20) exist in the literature. The mean of y_t need not be zero and may be a function of explanatory variables x_t (often a lag of y_t and an intercept term). Similarly h_t may be a function of observable variables (z_t) in addition to its own lags. An extended model along these lines is

$$\begin{aligned} y_t &= x_t^T \gamma + \exp(h_t/2)\, u_t, \\ h_t &= \omega + z_t^T \alpha + \beta h_{t-1} + \sigma v_t, \end{aligned} \tag{35.37}$$

It should be obvious that all these extensions are very easy to incorporate in EIS (see Liesenfeld and Richard 2003) and MCML (see Sandman and Koopman 1998). Bayesian estimation by MCMC remains quite usable but becomes more demanding in research time to tailor the algorithm for achieving a good level of efficiency of the Markov chain (see Chib et al. 2002, in particular p 301–302, for such comments).

Chib et al. (2002) also include a jump component term $k_t q_t$ in the conditional mean part to allow for irregular, transient movements in returns. The random variable q_t is equal to 1 with unknown probablity κ and zero with probability $1-\kappa$, whereas k_t is the size of the jump when it occurs. These time-varying jump sizes are assumed independent draws of $\ln(1+k_t) \sim N(-0.5\delta^2, \delta^2)$, δ being an unknown parameter representing the standard deviation of the jump size. For daily SP500 returns (period: 03/07/1962-26/08/1997) and a Student-t density for u_t, Chib et al. (2002) report posterior means of 0.002 for κ, and 0.034 for δ (for prior means of 0.02 and 0.05, respectively). This implies that a jump occurs on average every 500 days, and that the variability of the jump size is on average 3.4 per cent. They also find that the removal of the jump component from the model reduces the posterior mean of ν from 15 to 12, which corresponds to the fact that the jumps capture some outliers.

Another extension consists of relaxing the restriction of zero correlation (ρ) between u_t and v_t. This may be useful for stock returns for which a negative correlation corresponds to the leverage effect of the financial literature. If the correlation is negative, a drop of u_t, interpreted as a negative shock on the return, tends to increase v_t and therefore h_t. Hence volatility increases more after a negative shock than after a positive shock of the same absolute value, which is a well-known stylized fact. Sandman and Koopman (1998) estimate such a model by MCML, and report $\hat{\rho} = -0.38$ for daily returns of the SP500 index (period: 02/01/80-30/12/87), while Jacquier et al. (2004) do it by Bayesian inference using MCMC and report a posterior mean of ρ equal to -0.20 on the same data. They use the same reparametrization as in (35.13) to impose that the first diagonal element of the covariance matrix of u_t and σv_t must be equal to 1. This covariance matrix is given by

$$\Sigma = \begin{pmatrix} 1 & \rho\sigma \\ \rho\sigma & \sigma^2 \end{pmatrix} = \begin{pmatrix} 1 & \psi \\ \psi & \phi^2 + \psi^2 \end{pmatrix} \tag{35.38}$$

where the last matrix is a reparametrization. This enables to use a normal prior on the covariance ψ and an inverted gamma prior on ϕ^2, the conditional variance of

σv_t given u_t. The corresponding conditional posteriors are of the same type, so that simulating these parameters in the MCMC algorithm is easy. This approach can also be used if u_t has a Student-t distribution.

Multivariate SV models are also on the agenda of researchers. Liesenfeld and Richard (2003) estimate by EIS a one-factor model introduced by Shephard (1996), using return series of four currencies. Kim et al. (1998), Sect. 6.6, explain how to deal with the multi-factor model case by extending the MCMC algortihm reviewed in subsection 35.3.2.

35.3.5 Stochastic Duration and Intensity Models

Models akin to the SV model have been used for dynamic duration analysis by Bauwens and Veredas (2004) and Bauwens and Hautsch (2006). The context of application is the analysis of a sequence of time spells between events (also called durations) occurring on stock trading systems like the New York Stock Exchange (NYSE). Time stamps of trades are recorded for each stock on the market during trading hours every day, resulting in an ordered series of durations. Marks, such as the price, the exchanged quantity, the prevailing bid and ask prices, and other observed features may also be available, enabling to relate the durations to the marks in a statistical model. See Bauwens and Giot (2001) for a presentation of the issues.

Let $0 = t_0 < t_1 < t_2 < \ldots < t_n$ denote the arrival times and $d_1, d_2 \ldots d_n$ denote the corresponding durations, i.e. $d_i = t_i - t_{i-1}$. The stochastic conditional duration (SCD) model of Bauwens and Veredas (2004) is defined as

$$\begin{aligned} d_i &= \exp(\psi_i)\, u_i, & u_i &\sim D(\gamma), \quad t = 1, \ldots, n, \\ \psi_i &= \omega + \beta \psi_{i-1} + \sigma v_i, & v_i &\sim N(0,1), \end{aligned} \tag{35.39}$$

where $D(\gamma)$ denotes some distribution on the positive real line, possibly depending on a parameter γ. For example, Bauwens and Veredas use the Weibull distribution and the gamma distribution (both with shape parapeter denoted by γ).Assuming that the distribution of u_i is parameterized so that $E(u_i) = 1$, ψ_i is the logarithm of the unobserved mean of d_i, and is modelled by a Gaussian autoregressive process of order one. It is also assumed that $\{u_i\}$ and $\{v_i\}$ are mutually independent sequences. The parameters to be estimated are $(\omega, \beta, \sigma, \gamma)$, jointly denoted $\boldsymbol{\theta}$. The parameter space is $\mathrm{IR} \times (-1, 1) \times \mathrm{IR}_+ \times \mathrm{IR}_+$.

The similarity with the canonical SV model (35.20) is striking. A major difference is the non-normality of u_i since this is by definition a positive random variable. This feature makes it possible to identify γ. Therefore, the estimation methods available for the SV model can be adapted to the estimation of SCD models. Bauwens and Veredas (2004) have estimated the SCD model by the quasi-maximum likelihood (QML) method, since the first equation of the model may be expressed as $\ln d_i = \psi_i + \ln u_i$. If $\ln u_i$ were Gaussian, the model would be a

Gaussian linear state space model and the Kalman filter could be directly applied. QML relies on maximizing the likelihood function as if $\ln u_i$ were Gaussian. The QML estimator is known to be consistent but inefficient relative to the ML estimator which would obtain if the correct distribution of $\ln u_i$ were used to define the likelihood function. Bauwens and Galli (2009) have studied by simulation the loss of efficiency of QML relative to ML. ML estimation assuming a Weibull distribution is done by applying the EIS algorithm. For a sample size of 500 observations, the efficiency loss ranges from 20 to 50 per cent, except for the parameter ω, for which it is very small. They also applied the EIS method using the same data as in Bauwens and Veredas (2004). For example, for a dataset of 1,576 volume durations of the Boeing stock (period: September-November 1996; source: TAQ database of NYSE), the ML estimates are: $\hat{\omega} = -0.028$, $\hat{\beta} = 0.94$, $\hat{\sigma}^2 = 0.0159$, $\hat{\gamma} = 1.73$. They imply a high persistence in the conditional mean process (corresponding to duration clustering), a Weibull distribution with an increasing concave hazard function, and substantial heterogeneity. Notice that an interesting feature of the SCD model is that the distribution of u_i conditional to the past information, but marginalized with respect to the latent process, is a Weibull mixed by a lognormal distribution.

Strickland et al. (2006) have designed a MCMC algorithm for the SCD model (35.39) assuming a standard exponential distribution for u_i. The design of their MCMC algorithm borrows features from Koopman and Durbin's MCML approach and one of the MCMC algorithms used for the SV model. Feng et al. (2004) have extended the SCD model by including the additional term δu_{i-1} in the equation for ψ_i and they use MCML estimation to estimate the parameters (including δ).

As an alternative to modeling the sequence of durations, Bauwens and Hautsch (2006) model directly the arrival times through the intensity function of the point process. Their model specifies a dynamic intensity function, where the intensity function is the product of two intensity components: an observable component that depends on past arrival times, and a latent component. The logarithm of the latter is a Gaussian autoregressive process similar to the second equation in (35.20) and (35.39). The observable component may be a Hawkes process (Hawkes 1971) or an autoregressive intensity model (Russell 1999). When the model is multivariate, there is an observable intensity component specific to each particular point process, while the latent component is common to all particular processes. Interactions between the processes occur through the observable components and through the common component. The latter induces similar dynamics in the particular processes, reflecting the impact of a common factor influencing all processes. Bauwens and Hautsch use intensity-based likelihood inference, with the EIS algorithm to deal with the latent component.

35.4 Finite Mixture Models

Many econometric issues require models that are richer or more flexible than the conventional regression type models. Several possibilities exist. For example, as explained in subsection 35.2.2, the logit model is made more realistic by

generalizing it to a mixed logit. Many models currently used in econometrics can be generalized in such a way.

In this section, we assume that the univariate or multivariate observations $\boldsymbol{y}_j$ are considered as draws of

$$\tilde{f}(\boldsymbol{y}_j) = \sum_{g=1}^{G} \eta_g f(\boldsymbol{y}_j|\boldsymbol{\theta}_g) \tag{35.40}$$

with $\eta_1 + \ldots + \eta_G = 1$. The densities $f(\cdot|\boldsymbol{\theta}_g)$ are called component distributions. The observation $\boldsymbol{y}_j$ comes from one of these component distributions but we do not observe to which component it belongs. The mixture problem involves making inference about the η_g's and the parameters of the component distributions given only a sample from the mixture. The closer the component distributions are to each other, the more difficult this is because of problems of identifiability and computational instability.

35.4.1 Inference and Identification

The structure of (35.40) implies that the likelihood for all the J observations contains G^J terms

$$L(\boldsymbol{\eta}, \boldsymbol{\theta}|\boldsymbol{y}) \propto \prod_{j=1}^{J} \left(\sum_{g=1}^{G} \eta_g f(\boldsymbol{y}_j|\boldsymbol{\theta}_g) \right) \tag{35.41}$$

where $\boldsymbol{\eta} = (\eta_1, \ldots, \eta_G)^T$ and $\boldsymbol{\theta} = (\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_G)^T$ contain all the parameters and $\boldsymbol{y}$ denotes all the data. Maximum likelihood estimation using numerical optimization techniques, requiring many evaluations of the likelihood function, becomes cumbersome, if not unfeasible, for large G and J. This is even worse for multivariate observations.

Bayesian inference on finite mixture distributions by MCMC sampling is explained in Diebolt and Robert (1994). Gibbs sampling on $(\boldsymbol{\eta}, \boldsymbol{\theta})$ is difficult since the posterior distributions of $\boldsymbol{\eta}|\boldsymbol{\theta}, \boldsymbol{y}$ and $\boldsymbol{\theta}|\boldsymbol{\eta}, \boldsymbol{y}$ are generally unknown. For the same reason as for the probit model in Sect. 35.2.1 and the stochastic volatility model in Sect. 35.3, inference on the finite mixture model is straightforward once the state or group of an observation is known. Data augmentation is therefore an appropriate way to render inference easier. Define the state indicator S_j which takes value $s_j = g$ when $\boldsymbol{y}_j$ belongs to state or group g where $g \in \{1, \ldots, G\}$. Denote by $\boldsymbol{S}$ the J-dimensional discrete vector containing all the state indicators. To facilitate the inference, prior independence, that is $\varphi(\boldsymbol{\eta}, \boldsymbol{\theta}, \boldsymbol{S}) = \varphi(\boldsymbol{\eta})\varphi(\boldsymbol{\theta})\varphi(\boldsymbol{S})$, is usually imposed. As shown in the next examples, the posterior distributions $\boldsymbol{S}|\boldsymbol{\eta}, \boldsymbol{\theta}, \boldsymbol{y}$, $\boldsymbol{\theta}|\boldsymbol{\eta}, \boldsymbol{S}, \boldsymbol{y}$ and $\boldsymbol{\eta}|\boldsymbol{\theta}, \boldsymbol{S}, \boldsymbol{y}$ are either known distributions easy to sample from or they

are distributions for which a second, but simpler, MCMC sampler is set up. A Gibbs sampler with three main blocks may therefore be used.

The complete data likelihood of the finite mixture is invariant to a relabeling of the states. This means that we can take the labeling $\{1, 2, \ldots, G\}$ and do a permutation $\{\rho(1), \rho(2), \ldots, \rho(G)\}$ without changing the value of the likelihood function. If the prior is also invariant to relabeling then the posterior has this property also. As a result, the posterior has potentially $G!$ different modes. To solve this identification or label switching problem, identification restrictions have to be imposed.

Note that the inference described here is conditional on G, the number of components. There are two modeling approaches to take care of G. First, one can treat G as an extra parameter in the model as is done in Richardson and Green (1997) who make use of the reversible jump MCMC methods. In this way, the prior information on the number of components can be taken explicitly into account by specifying for example a Poisson distribution on G in such a way that it favors a small number of components. A second approach is to treat the choice of G as a problem of model selection. Bayesian model comparison techniques (see Chap. III.11) can be applied, for instance by the calculation of the Bayes factor, see Cowles and Carlin (1996) and Chib (1995) for more details.

35.4.2 Examples

We review two examples. The first example fits US quarterly GNP data using a mixture autoregressive model. The second example is about the clustering of many GARCH models.

Mixture Autoregressive Model

Frühwirth-Schnatter (2001) uses US quarterly real GNP growth data from 1951:2 to 1984:4. This series was initially used by Hamilton (1989) and is displayed in Fig. 35.2.

The argument is that contracting and expanding periods are generated by the same model but with different parameters.

After some investigation using Bayesian model selection techniques, the adequate specification for the US growth data is found to be the two-state mixture AR(2) model

$$\begin{aligned} y_t &= \beta_{i,1} y_{t-1} + \beta_{i,2} y_{t-2} + \beta_{i,3} \\ &\quad + \epsilon_{t,i} \qquad \epsilon_{t,i} \sim \mathrm{N}(0, \sigma_i^2) \qquad i = 1, 2. \end{aligned} \tag{35.42}$$

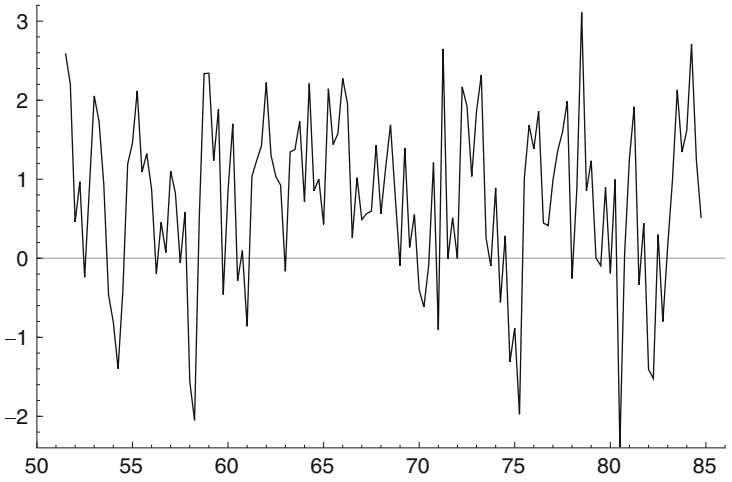


Fig. 35.2 US real GNP growth data in percentages (1951:2–1984:4)

The idea behind the choice of two states is motivated by the contracting (negative growth) and expanding periods (positive growth) in an economy. The conditional posteriors for the σ_i^2's are independent inverted gamma distributions. For the β_i's, the conditional posteriors are independent normal distributions. Inference for the mixture model in (35.42) is done in two steps. The first step is to construct an MCMC sample by running the random permutation sampler. Generally speaking, a draw from the random permutation sampler is obtained as follows:

(1) Draw from the model by use of the Gibbs sampler for example.
(2) Relabel the states randomly.

By so-doing, one samples from the unconstrained parameter space with balanced label switching. Note that in (2), there are $G!$ possibilities to relabel when there are G possible states.

In the second step, this sample is used to identify the model. This is done by visual inspection of the posterior marginal and bivariate densities. Identification restrictions need to be imposed to avoid multimodality of the posterior densities. Once suitable restrictions are found, a final MCMC sample is constructed to obtain the moments of the constrained posterior density. The latter sample is constructed by permutation sampling under the restrictions, which means that (2) is replaced by one permutation defining the constrained parameter space.

In the GNP growth data example, two identification restrictions seem possible, namely $\beta_{1,1} < \beta_{2,1}$ and $\beta_{1,3} < \beta_{2,3}$, see Frühwirth-Schnatter (2001) for details. Table 35.9 provides the posterior means and standard deviations of the $\beta_{i,j}$'s for both identification restrictions.

Table 35.9 Posterior means and standard deviations

	$\beta_{1,1} < \beta_{2,1}$		$\beta_{1,3} < \beta_{2,3}$	
	Contraction	Expansion	Contraction	Expansion
$\beta_{i,1}$	0.166 (.125)	0.33 (0.101)	0.249 (0.164)	0.295 (0.116)
$\beta_{i,2}$	0.469 (.156)	−0.129 (0.093)	0.462 (0.164)	−0.114 (0.098)
$\beta_{i,3}$	−0.479 (.299)	1.07 (0.163)	−0.557 (0.322)	1.06 (0.175)

Table 35.10 Summary of conditional posteriors

Parameter	Conditional posterior or sampling method
$\boldsymbol{S}$	Multinomial distribution
$\boldsymbol{\eta}$	Dirichlet distribution
$\boldsymbol{\theta}$	Griddy-Gibbs sampler

The GNP growth in contraction and expansion periods not only have different unconditional means, they are also driven by different dynamics. Both identification restrictions result in similar posterior moments.

Clustering of Many GARCH Models

Bauwens and Rombouts (2007) focus on the differentiation between the component distributions via different conditional heteroskedasticity structures by the use of GARCH models. In this framework, the observation $\boldsymbol{y}_j$ is multivariate and the $\boldsymbol{\theta}_g$'s are the parameters of GARCH(1,1) models. The purpose is to estimate many, of the order of several hundreds, GARCH models. Each financial time series belongs to one of the G groups but it is not known a priori which series belongs to which cluster.

An additional identification problem arises due to the possibility of empty groups. If a group is empty then the posterior of $\boldsymbol{\theta}_g$ is equal to the prior of $\boldsymbol{\theta}_g$. Therefore an improper prior is not allowed for $\boldsymbol{\theta}_g$. The identification problems are solved by using an informative prior on each $\boldsymbol{\theta}_g$. The identification restrictions use the fact that we work with GARCH models: we select rather non-overlapping supports for the parameters, such that the prior $\varphi(\boldsymbol{\theta}) = \prod_{g=1}^{G} \varphi(\boldsymbol{\theta}_g)$ depends on a labeling choice. Uniform prior densities on each parameter, on finite intervals, possibly subject to stationarity restrictions, are relatively easy to specify.

Bayesian inference is done by use of the Gibbs sampler and data augmentation. Table 35.10 summarizes the three blocks of the sampler.
Because of the prior independence of the $\boldsymbol{\theta}_g$'s, the griddy-Gibbs sampler is applied separately G times.

As an illustration we show the posterior marginals of the following model

$$\tilde{f}(\boldsymbol{y}_j) = \sum_{g=1}^{3} \eta_g f(\boldsymbol{y}_j | \boldsymbol{\theta}_g) \tag{35.43}$$

with $\eta_1 = 0.25$, $\eta_2 = 0.5$, $J = 100$ and $T_j = 1000$. The components are defined more precisely as

$$f(\boldsymbol{y}_j|\boldsymbol{\theta}_g) = \prod_{t=1}^{T_j} f(y_{j,t}|\boldsymbol{\theta}_g, I_{j,t}) \tag{35.44}$$

$$y_{j,t}|\boldsymbol{\theta}_g, I_{j,t} \sim N(0, h_{j,t}) \tag{35.45}$$

$$h_{j,t} = (1 - \alpha_g - \beta_g)\tilde{\omega}_j + \alpha_g (y_{j,t-1})^2 + \beta_g h_{j,t-1} \tag{35.46}$$

where $I_{j,t}$ is the information set until $t-1$ containing (at least) $y_{j,1}, \ldots, y_{j,t-1}$ and initial conditions which are assumed to be known. For the simulation of the data $\tilde{\omega}_j$ is fixed equal to one which implies that the unconditional variance for every generated data series is equal to one. However, the constant $\tilde{\omega}_j$ in the conditional variance is not subject to inference, rather it is fixed at the empirical variance of the data. Table 35.11 presents the true values, the prior intervals on the $\boldsymbol{\theta}_g$'s and posterior results on $\boldsymbol{\eta}$ and $\boldsymbol{\theta}$.

Bauwens and Rombouts (2007) successfully apply this model to return series of 131 US stocks. Comparing the marginal likelihoods for different models, they find that $G = 3$ is the appropriate choice for the number of component distributions.

Other interesting examples of finite mixture modeling exist in the literature. Frühwirth-Schnatter and Kaufmann (2008) develop a panel data model for clustering many short time series. Their economic application concerns convergence clubs of countries based on income data and they find two clusters. Deb and Trivedi (1997) develop a finite mixture negative binomial count model to estimate six measures of medical care demand by the elderly. Chib and Hamilton (2000) offer a

Table 35.11 Posterior results on $\boldsymbol{\eta}$ and $\boldsymbol{\theta}$ ($G = 3$)

		η_1	η_2	η_3	
True value			**0.25**	**0.50**	**0.25**
Mean			0.2166	0.4981	0.2853
Standard deviation			0.0555	0.0763	0.0692
Correlation matrix			1	−0.4851	−0.2677
			−0.4851	1	−0.7127
			−0.2677	−0.7127	1
			$g = 1$	$g = 2$	$g = 3$
True value	α_g		**0.04**	**0.12**	**0.20**
	β_g		**0.90**	**0.60**	**0.40**
Prior interval	α_g		0.001,0.07	0.07,0.15	0.15,0.25
	β_g		0.65,0.97	0.45,0.75	0.20,0.60
Mean	α_g		0.0435	0.1041	0.1975
	β_g		0.8758	0.5917	0.4369
Standard deviation	α_g		0.0060	0.0092	0.0132
	β_g		0.0238	0.0306	0.0350
Correlation α_g, β_g			−0.7849	−0.71409	−0.7184

flexible Bayesian analysis of the problem of causal inference in models with non-randomly assigned treatments. Their approach is illustrated using hospice data and hip fracture data.

35.5 Change Point Models

It is well known that economic time series are not stationary, especially when observed for long periods. A source of non-stationarity is the technological and institutional changes of the economic environment, which may occur more or less abruptly. When changes are gradual, they do not translate immediately into observed data as there may be threshold or ratchet effects. Econometric models with a fixed structural form or constant parameters are thus potentially misspecified. Following Hamilton (1989), Markov switching models allow capturing regime changes in econometric models, by driving the parameter changes through a discrete hidden Markov chain. At each date, the parameters of the model are in a given state determined by a value of a discrete latent variable. At the next date, they can stay in the same state or change to another state, among a few possible values. At a further date, they may switch back to a previous state, i.e. states can be recurrent. In an influential article, Chib (1998), proposes a Markov switching model specification with non-recurrent states as a new approach to the modeling of multiple change points or structural breaks in time series models. This type of change-point model is essentially a Markov switching model in which the transition probabilities are constrained so that the state variable can either stay at the current value or jump to the next higher value associated to the next regime. The last regime is an absorbing state. Several authors have used this change point model formulation for empirical research, see e.g. Kim and Nelson (1999), Pastor and Stambaugh (2001), and Liu and Maheu (2008).

Using this model framework, Pesaran et al. (2006) provide a new approach to forecasting time series that are subject to structural breaks. Using Bayesian inference, they propose a prediction procedure that allows for the possibility of new breaks occurring over the forecast horizon, taking account of the size and duration of past breaks (if any). Predictions are formed by integrating over the parameters from the meta-distribution that characterizes the stochastic break-point process (see Sect. 35.5.1 for details). In their application to U.S. Treasury bill rates, they find that their method leads to better out-of-sample forecasts than a range of alternative methods.

Koop and Potter (2007) develop a related approach to change-point modeling, which, in contrast to Chib's formulation, permits the number of change-points in the observed sample to be a priori unknown. Their model assumes that regime durations have a Poisson distribution and nests the two most common approaches: the time-varying parameter model with a change-point every period, and the change-point model with a small number of regimes. A MCMC sampler is constructed to estimate a version of their model, which allows for change in conditional means and

variances. They show how forecasting can be done in an efficient manner using sequential importance sampling. Their empirical illustration involves U.S. GDP growth and inflation. Yet another approach to modeling breaks in time series is proposed by Giordani and Kohn (2008). They model the break process in a state space representation through mixture distributions for the state innovations. Similarly to Koop and Potter (2007), this allows for a random number of breaks. They illustrate important computational efficiency gains, using an adaptive Metropolis Hastings sampler, in applications using US real interest rate data and US inflation rate data.

Although the literature on the specification of change-point models in economics is about a dozen years old at the time of writing of this section, many questions are still unresolved, such as which approach is most useful for forecasting.

35.5.1 Linking Regime Parameters with a Hierarchical Prior

Let y_t be the time series to be modeled over the sample period $\{1, 2, \ldots, T\}$. Let s_t be an integer latent variable taking its value in the set $\{1, 2, \ldots, K\}$, K being assumed known. This state variable s_t indicates the active regime generating y_t at period t in the sense that y_t is generated from the data density $f(y_t|\boldsymbol{Y}_{t-1}, \boldsymbol{\theta}_{s_t})$, where $\boldsymbol{Y}_{t-1} = (\boldsymbol{Y}_0^T \; y_1 \; \ldots \; y_{t-1})^T$, $\boldsymbol{Y}_0$ being a vector of known initial conditions (observations prior to date $t = 1$), and $\boldsymbol{\theta}_{s_t}$ is a vector of parameters indexing the density. There are potentially K regimes, hence K parameter vectors $\boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_K$.

The active regime at t is assumed to be selected by a discrete first-order Markov process for the s_t process. As in Chib (1998), the transition probability matrix $\boldsymbol{P}_K$ allows either to stay in the regime operating at $t-1$ or to switch to the next regime. Therefore the elements of its main diagonal are the probabilities $p_k = Prob[s_t = k|s_{t-1} = k]$ with $0 < p_k < 1$ for $k = 1, 2, \ldots, K-1$ and $p_K = 1$ (implying that $P[s_T = K] = 1$). The other elements of $\boldsymbol{P}_K$ which are different from 0 are just to the right of each p_k (if $k < K$), and are equal to $1 - p_k = Prob[s_t = k+1|s_{t-1} = k]$. Note that the last regime is an absorbing state over the sample period. Given the zero entries in $\boldsymbol{P}_K$, the Markov chain generates potentially $K-1$ breaks, at random dates τ_k ($k = 1, 2, \ldots, K-1$) defined by τ_k being the integer in $\{1, 2, \ldots, T\}$ such that $s_{\tau_k} = k$ and $s_{\tau_k+1} = k+1$. A posterior density on these dates is therefore a direct by-product of the inference on the state variables. A convenient prior density, based on the assumption of independence between the parameters of the matrix $\boldsymbol{P}_K$, takes the form of a product of identical beta densities with parameters $\underline{a}$ and $\underline{b}$:

$$\varphi(p_1, p_2, \ldots, p_{K-1}) \propto \prod_{i=1}^{K-1} p_i^{\underline{a}-1}(1-p_i)^{\underline{b}-1}. \tag{35.47}$$

The assumption that the beta densities are identical can be easily relaxed. This prior implies that the (strictly positive integer) duration of regime k, $d_k = \tau_k - \tau_{k-1}$

(setting $\tau_0 = 0$ and $\tau_K = T$) is approximately geometrically distributed with parameter p_i and expected value $(\underline{a} + \underline{b})/\underline{a}$.

An essential ingredient of the model specification is the prior assumption that the parameter vectors $\boldsymbol{\theta}_i$ are drawn independently from a common distribution, i.e. $\boldsymbol{\theta}_i \sim \varphi(\boldsymbol{\theta}_i|\boldsymbol{\theta}_0)$ where $\boldsymbol{\theta}_0$ itself is a parameter vector endowed with a prior density $\varphi(\boldsymbol{\theta}_0|\underline{\boldsymbol{A}})$, $\underline{\boldsymbol{A}}$ denoting the prior hyper-parameters. This is called a hierarchical prior or a meta-distribution. For example, if $\boldsymbol{\theta}_i$ contains location parameters and a scale one, the prior can be a normal density on the location parameters and an inverted gamma density on the scale one. Generally, the joint prior on the $\boldsymbol{\theta}$ parameters is

$$\varphi(\boldsymbol{\theta}_0, \boldsymbol{\theta}_1, \boldsymbol{\theta}_2, \ldots, \boldsymbol{\theta}_K) = \varphi(\boldsymbol{\theta}_0|\underline{\boldsymbol{A}}) \prod_{i=1}^{K} \varphi(\boldsymbol{\theta}_i|\boldsymbol{\theta}_0). \tag{35.48}$$

Behind the common prior (35.48) lies the belief that the regime parameters differ and evolve independently of each other. Another possible prior belief is that the regime parameters evolve in a more structured way. For example the conditional mean of y_t could be increasing ($\mu_{k-1} < \mu_k$). This idea can be formalized through a joint normal prior on $(\mu_2 - \mu_1, \mu_3 - \mu_2, \ldots, \mu_K - \mu_{K-1})^T$ with mean vector $\boldsymbol{m}_0$ and covariance matrix $\boldsymbol{V}_0$, where $\boldsymbol{m}_0$ and $\boldsymbol{V}_0$ are the hyper-parameters to be endowed with a prior density implying that $\boldsymbol{m}_0$ is positive with high probability.

The model is fully specified by assuming $f(y_t|\boldsymbol{Y}_{t-1}, \boldsymbol{\theta}_{s_t}) = N(\boldsymbol{x}_t^T \boldsymbol{\beta}_{s_t}, \sigma_{s_t}^2)$, where $\boldsymbol{x}_t$ is a vector of m predetermined variables and $\boldsymbol{\beta}_{s_t}$ a vector of coefficients, so that $\boldsymbol{\theta}_{s_t} = (\boldsymbol{\beta}_{s_t}^T \sigma_{s_t}^2)^T$. In the example of the next section, the model within a regime is autoregressive of order p (AR(p)), i.e. $\boldsymbol{x}_t$ includes the constant 1 and p lags of y_t, hence $m = p + 1$.

The joint posterior density of $\boldsymbol{S}_T = (s_1 \; s_2 \; \ldots, s_T)'$ and the parameters is proportional to

$$\prod_{t=1}^{T} f(y_t|\boldsymbol{Y}_{t-1}, \boldsymbol{\theta} s_t) f(s_t|s_{t-1}, \boldsymbol{P}_K) \prod_{i=1}^{K-1} p_i^{\underline{a}-1}(1-p_i)^{\underline{b}-1} \prod_{i=1}^{K} \varphi(\boldsymbol{\theta}_i|\boldsymbol{\theta}_0)\, \varphi(\boldsymbol{\theta}_0|\underline{\boldsymbol{A}}),$$

where $f(s_t|s_{t-1}, \boldsymbol{P}_K)$ is the transition probability from state $t-1$ to state t and is one of the non null elements of $\boldsymbol{P}_K$. The parameters are $\boldsymbol{\theta}_i$, $i = 0, 1, \ldots, K$, jointly denoted by $\boldsymbol{\Theta}_K$, and the diagonal elements of the matrix $\boldsymbol{P}_K$. This density lends itself to simulation by Gibbs sampling in three blocks corresponding to the full conditional densities:

1. $\varphi(\boldsymbol{S}_T|\boldsymbol{\Theta}_K, \boldsymbol{P}_K, \boldsymbol{Y}_T) \propto \prod_{t=1}^{T} f(y_t|\boldsymbol{Y}_{t-1}, \boldsymbol{\theta}_{s_t}) f(s_t|s_{t-1}, \boldsymbol{P}_K)$,
2. $\varphi(\boldsymbol{P}_K|\boldsymbol{S}_T) \propto \prod_{t=1}^{T} f(s_t|s_{t-1}, \boldsymbol{P}_K) \prod_{i=1}^{K-1} p_i^{\underline{a}-1}(1-p_i)^{\underline{b}-1}$, and
3. $\varphi(\boldsymbol{\Theta}_K|\boldsymbol{S}_T, \boldsymbol{Y}_T) \propto \prod_{t=1}^{T} f(y_t|\boldsymbol{Y}_{t-1}, \boldsymbol{\theta}_{s_t}) \prod_{i=1}^{K} \varphi(\boldsymbol{\theta}_i|\boldsymbol{\theta}_0)\, \varphi(\boldsymbol{\theta}_0|\underline{\boldsymbol{A}})$.

Sampling $\boldsymbol{S}_T$ is done as explained Chib (1998). This algorithm implies that all the states are actually sampled from their joint distribution, not one at a time given the

other states. Sampling $\boldsymbol{P}_K$ is done by simulating each p_i from a beta density with parameters $\underline{a} + T_i$ and $\underline{b} + 1$ where T_i is the number of states equal to i in the sampled $\boldsymbol{S}_T$ vector. Sampling $\boldsymbol{\Theta}_K$ implies usually to break it into sub-blocks and to sample each sub-block given the other and $\boldsymbol{S}_T$. All the details can be found in Bauwens and Rombouts (2009).

The number of break points K plays a crucial role in the change point model. Inference on this parameter is done by maximizing the marginal likelihood over a range for K, say $K \in \{1, 2, \ldots, \bar{K}\}$ where $\bar{K}$ is the largest number of regimes that we wish to consider. To compute the marginal log-likelihood for the data $\boldsymbol{Y}_T$ and the model M_K with parameters $\boldsymbol{\Theta}_K$ and $\boldsymbol{P}_K$, the idea of Chib (1995) is useful. The predictive density is related to the prior, posterior and data density by the equality

$$f(\boldsymbol{Y}_T|M_K) = \frac{f(\boldsymbol{Y}_T|M_K, \boldsymbol{\Theta}_K, \boldsymbol{P}_K)\varphi(\boldsymbol{\Theta}_K, \boldsymbol{P}_K|M_K)}{\varphi(\boldsymbol{\Theta}_K, \boldsymbol{P}_K|M_K, \boldsymbol{Y}_T)}. \tag{35.49}$$

Since this equality holds for any admissible parameter value, we can pick a value $(\boldsymbol{\Theta}_K^*, \boldsymbol{P}_K^*)$ of the parameters, usually the posterior mean or mode, and compute

$$\log f(\boldsymbol{Y}_T|M_K, \boldsymbol{\Theta}_K^*, \boldsymbol{P}_K^*) + \log \varphi(\boldsymbol{\Theta}_K^*, \boldsymbol{P}_K^*|M_K) - \log \varphi(\boldsymbol{\Theta}_K^*, \boldsymbol{P}_K^*|M_K, \boldsymbol{Y}_T) \tag{35.50}$$

to approximate $\log f(\boldsymbol{Y}_T|M_K)$. The evaluation of the last part in (35.50) requires further simulations. Bauwens and Rombouts (2009) perform a Monte Carlo study and find that Chib's method is robust with respect to the choice of the parameter value used in the computations, among posterior mean, mode and quartiles.

Apart from break point detection, the change-point model can be used for forecasting. Using Bayesian inference, Pesaran et al. (2006), show how to compute predictive densities that take into account out-of-sample structural breaks. In fact, if breaks occurred in the past, they are likely to happen in the future as well. The posterior on $\boldsymbol{\theta}_0$ plays a crucial role in the computation of predictive densities that take into account the possibility of future breaks. The predictive densities are obtained as a by-product of the MCMC sampler and therefore do not require extra simulations.

35.5.2 Example

We provide the results of applying the change point model to the US industrial production growth rate series. The sample covers the period from January 1950 to January 2009 (709 observations) and is plotted in Fig. 35.3.

The original series, downloaded from Datastream (USIPTOT.G series), is a volume index (equal to 100 in 2002) of the industrial production of the USA and is

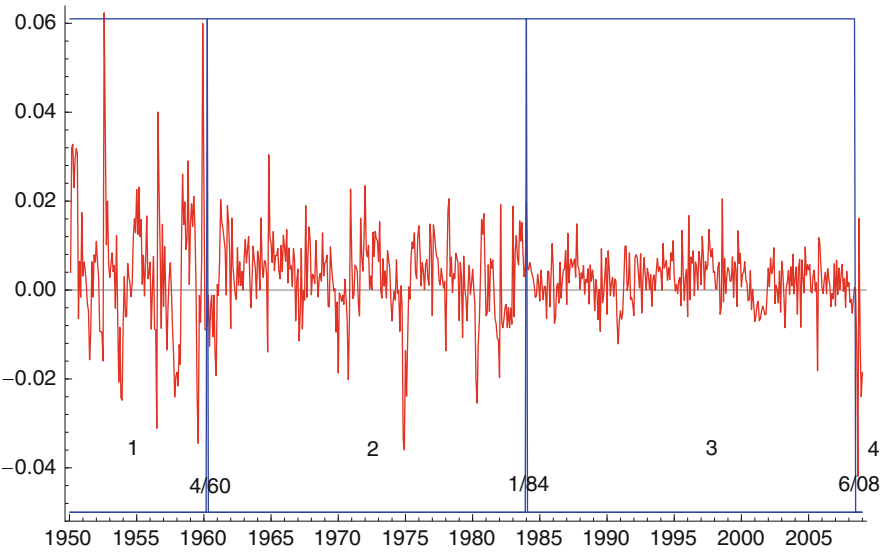


Fig. 35.3 U.S. Industrial production growth from January 1950-January 2009

Table 35.12 MLL of AR(1) models

K	1	2	3	4	5	6	7
	−945.11	−894.61	−882.18	−859.11	−861.68	−864.02	−865.63

MLL: marginal log-likelihood, computed by formula (35.50) using the posterior mean

seasonally adjusted. We use an AR(1) model in each regime and limit the number of regimes to $\bar{K} = 7$ at most. According to the marginal log-likelihood criterion, see Table 35.12, the best model has three change points (or four regimes).

The posterior mean, mode, and quartiles of the parameters of the best model are reported in Table 35.13. The posterior medians of the break dates are April 1960, January 1984, and June 2008, as shown in Fig. 35.3. The first break corresponds to a large reduction in the variance of the growth rate: the posterior mean of the residual variance is divided by three. The second break corresponds to a further reduction (division by 2.5). The last break corresponds clearly to the big recession triggered by the subprime crisis of 2007, with a huge increase of the residual variance and a negative average growth rate. The coefficients of the AR(1) equation are similar in the first two regimes, and change a lot in the last two regimes, though the precision of the estimation for the last regime is low due to the limited information available for that regime. Finally, the probability to remain in any of the first three regimes is close to 1, reflecting the long durations of each of these regimes. It is nevertheless reassuring that the model can capture the last break, even if it does so with a little delay.

Table 35.13 Posterior results for 3-change point AR(1) model

Parameter	Mean	Std. dev.	Mode	Median	q25	q75
β_{11}	0.195	0.157	0.307	0.199	0.103	0.291
β_{12}	0.470	0.093	0.441	0.470	0.415	0.526
β_{21}	0.171	0.057	0.200	0.170	0.136	0.207
β_{22}	0.405	0.062	0.395	0.405	0.368	0.443
β_{31}	0.191	0.034	0.171	0.191	0.168	0.213
β_{32}	0.100	0.067	0.144	0.099	0.059	0.139
β_{41}	−0.200	0.520	−0.036	−0.189	−0.523	0.129
β_{42}	0.184	0.354	0.463	0.179	−0.045	0.400
σ_1^2	2.075	0.364	2.065	2.043	1.868	2.240
σ_2^2	0.680	0.066	0.668	0.678	0.639	0.717
σ_3^2	0.262	0.025	0.283	0.260	0.246	0.275
σ_4^2	5.390	4.174	4.857	4.259	2.993	6.523
p_1	0.988	0.011	0.996	0.991	0.984	0.995
p_2	0.995	0.004	0.998	0.996	0.993	0.998
p_3	0.995	0.004	0.999	0.996	0.993	0.998

Posterior density summarized by its mean, standard deviation (Std. dev.), mode, median, 0.25-quantile (q25) and 0.75-quantile (q75). In regime i, the model is $y_t = \beta_{i1} + \beta_{i2} y_{t-1} + \sigma_i \epsilon_t$ with $\epsilon_t \sim N(0,1)$, and $p_k = P[s_t = k | s_{t-1} = k]$

References

Allenby, G., Rossi, P.: Marketing models of consumer heterogeneity. J. Econometrics **89**, 57–78 (1999)

Bauwens, L., Galli, F.: Efficient importance sampling for ML estimation of SCD models. Comput. Stat. Data Anal. **53**, 1974–1992 (2009)

Bauwens, L., Giot, P.: Econometric Modelling of Stock Market Intraday Activity. Kluwer (2001)

Bauwens, L., Hautsch, N.: Dynamic latent factor models for intensity processes. J. Financ. Econometrics **4**, 450–493 (2006)

Bauwens, L., Rombouts, J.: Bayesian clustering of many GARCH models. Econometrics Rev. **26**, 365–386 (2007)

Bauwens, L., Rombouts, J.: On marginal likelihood computation in change-point models, Forthcoming in Computational Statistics & Data Analysis. CORE DP 2009/61 (2009)

Bauwens, L., Veredas, D.: The stochastic conditional duration model: a latent factor model for the analysis of financial durations. J. Econometrics **119**, 381–412 (2004)

Bhat, C.: Quasi-random maximum simulated likelihood estimation of the mixed multinomial logit model. Transp. Res. Part B **35**, 677–693 (2001)

Bollerslev, T., Engle, R., Nelson, D.: ARCH models. In: Engle, R., McFadden, D. (eds.) Handbook of Econometrics, chapter 4, pp. 2959–3038. North Holland Press, Amsterdam (1994)

Brownstone, D., Train, K.: Forecasting new product penetration with flexible substitution patterns. J. Econometrics **89**, 109–129 (1999)

Chib, S.: Marginal likelihood from the Gibbs output. J. Am. Stat. Assoc. **90**, 1313–1321 (1995)

Chib, S.: Estimation and comparison of multiple change-point models. J. Econometrics **86**, 221–241 (1998)

Chib, S., Greenberg, E.: Analysis of multivariate probit models. Biometrika **85**, 347–361 (1998)

Chib, S., Hamilton, B.: Bayesian analysis of cross-section and clustered data treatment models. J. Econometrics **97**, 25–50 (2000)

Chib, S., Nardari, F., Shephard, N.: Markov chain Monte Carlo methods for stochastic volatility models. J. Econometrics **108**, 291–316 (2002)
Chintagunta, P., Honore, B.: Investigating the effects of marketing variables and unobserved heterogeneity in a multinomial probit model. Int. J. Res. Market. **13**, 1–15 (1996)
Cowles, M., Carlin, B.: Markov chain Monte Carlo convergence diagnostics: A comparative review. J. Am. Stat. Assoc. **91**, 883–904 (1996)
Cribari-Neto, F.: Econometric programming environnments: Gauss, Ox and S-plus. J. Appl. Econometrics 12:77–89 (1997)
Danielson, J.: Stochastic volatility in asset prices: estimation with simulated maximum likelihood. J. Econometrics **61**, 375–400 (1994)
de Jong, P., Shephard, N.: The simulation smoother for time series models. Biometrika **82**, 339–350 (1995)
Deb, P., Trivedi, P.: Demand for medical care by the elderly: A finite mixture approach. J. Appl. Econometrics **12**, 313–336 (1997)
Diebolt, J., Robert, C.: Estimation of finite mixture distributions through Bayesian sampling. J. Roy. Stat. Soc. B **56**, 363–375 (1994)
Durbin, J., Koopman, S.: Monte Carlo maximum likelihood estimation for non-Gaussian state space models. Biometrika 84:669–684 (1997)
Durbin, J., Koopman, S.: Time series analysis of non-Gaussian observations based on state space models from both classical and Bayesian perspectives. J. Roy. Stat. Soc. B **62**, 3–56 (2000)
Fan, J., Yao, Q.: Nonlinear Time Series, (2nd edn.), Springer, Berlin (2006)
Feng, D., Jiang, G.J., Song, P.X.-K.: Stochastic conditional duration models with leverage effect for financial transaction data. J. Financ. Econometrics **2**, 390–421 (2004)
Franses, P., Paap, R.: Quantitative Models in Marketing Research. Cambridge University Press, Cambridge (2001)
Frühwirth-Schnatter, S.: Markov chain Monte Carlo estimation of classical and dynamic switching and mixture models. J. Am. Stat. Assoc. **96**, 194–209 (2001)
Frühwirth-Schnatter, S., Kaufmann, S.: Model-based clustering of multiple time series. J. Bus. Econ. Stat. **26**, 78–89 (2008)
Geweke, J., Keane, M., Runkle, D.: Statistical inference in the multinomial multiperiod probit model. J. Econometrics **80**, 125–165 (1997)
Ghysels, E., Harvey, A., Renault, E.: Stochastic volatility. In: Maddala, G., Rao, C. (eds.) Handbook of Statistics, pp. 119–191. Elsevier Science, Amsterdam (1996)
Giordani, P., Kohn, R.: Efficient bayesian inference for multiple change-point and mixture innovation models. J. Bus. Econ. Stat. **26**, 66–77 (2008)
Gouriéroux, C., Monfort, A.: Simulation-based Econometric Methods. Oxford University Press, Oxford (1997)
Hajivassiliou, V., Mc Fadden, D.: The method of simulated scores for the estimation of LDV models. Econometrica **66**, 863–896 (1998)
Hajivassiliou, V., Ruud, P.: *Classical estimation methods for LDV models using simulation*. In: Handbook of Econometrics, Vol. 4, Chapter 40, North Holland, Amsterdam (1994)
Hamilton, J.: A new approach to the economic analysis of nonstationary time series and the business cycle. Econometrica **57**, 357–384 (1989)
Hawkes, A.G.: Spectra of some self-exciting and mutually exciting point processes. Biometrika **58**, 83–90 (1971)
Horowitz, J.: Semiparametric Methods in Econometrics. Springer, Berlin (1998)
Jacquier, E., Polson, N., Rossi, P.: Bayesian analysis of stochastic volatility models (with discussion). J. Bus. Econ. Stat. **12**, 371–417 (1994)
Jacquier, E., Polson, N., Rossi, P.: Bayesian analysis of stochastic volatility models with fat-tails and correlated errors. J. Econometrics **122**, 185–212 (2004)
Kim, C.-J., Nelson, C.R.: Has the U.S. economy become more stable? a Bayesian approach based on a Markov-switching model of the business cycle. Rev. Econ. Stat. **81**, 608–616 (1999)
Kim, S., Shephard, N., Chib, S.: Stochastic volatility: likelihood inference and comparison with ARCH models. Rev. Econ. Stud. **65**, 361–393 (1998)

Koop, G., Potter, S.: Estimation and forecasting with multiple breaks. Rev. Econ. Stud. **74**, 763–789 (2007)
Li, Q., Racine, J.S.: Nonparametric Econometrics: Theory and Practice. Princeton University Press, USA (2006)
Liesenfeld, R., Richard, J.-F.: Univariate and multivariate stochastic volatility models: estimation and diagnostics. J. Empir. Finance **10**, 505–531 (2003)
Liu, C., Maheu, J.: Are there structural breaks in realized volatility? J. Financ. Econometrics **6**, 326–360 (2008)
Maddala, G.: Limited-dependent and Qualitative Variables in Econometrics. Cambridge University Press, Cambridge (1983)
Mariano, R., Schuermann, T., Weeks, M.: Simulation-based Inference in Econometrics. Cambridge University Press, Cambridge (2000)
McCulloch, R., Polson, N., Rossi, P.: A Bayesian analysis of the multinomial probit model with fully identified parameters. J. Econometrics **99**, 173–193 (2000)
McFadden, D., Train, K.: Mixed MNL models for discrete response. J. Appl. Econometrics **15**, 447–470 (2000)
Paap, R., Franses, P.: A dynamic multinomial probit model for brand choice with different long-run and short-run effects of marketing-mix variables. J. Appl. Econometrics **15**, 717–744 (2000)
Pagan, A., Ullah, A.: Nonparametric Econometrics. Cambridge University Press, Cambridge (1999)
Pastor, L., Stambaugh, R.F.: The equity premium and structural breaks. J. Finan. **56**, 1207–1239 (2001)
Pesaran, M.H., Pettenuzzo, D., Timmermann, A.: Forecasting time series subject to multiple structural breaks. Rev. Econ. Stud. **73**, 1057–1084 (2006)
Richard, J.-F., Zhang, W.: Efficient high-dimensional Monte Carlo importance sampling. J. Econometrics **141**, 1385–1411 (2007)
Richardson, S., Green, P.: On Bayesian analysis of mixtures with an unknown number of components. J. Roy. Stat. Soc. B **59**, 731–792 (1997)
Russell, J.: Econometric modeling of multivariate irregularly-spaced high-frequency data. Manuscript, University of Chicago, Illinois, USA (1999)
Sandman, G., Koopman, S.: Estimation of stochastic volatility models via Monte Carlo maximum likelihood. J. Econometrics **67**, 271–301 (1998)
Shephard, N.: Statistical aspects of ARCH and stochastic volatility, In: Cox, D.R., Hinkley, D.V., Barndorff-Nielsen, O.E. (eds.) chapter 1, pp. 1–67. Time Series Models: In Econometrics, Finance and Other Fields. Chapman & Hall, London (1996)
Strickland, C., Forbes, C., Martin, G.: Bayesian analysis of the stochastic conditional duration model. Comput. Stat. Data Anal. **50**, 2247–2267 (2006)
Tanner, M., Wong, W.: The calculation of posterior distributions by data augmentation. J. Am. Stat. Assoc. **82**, 528–540 (1987)
Train, K.: Discrete Choice Methods with Simulation. Cambridge University Press, Cambridge (2003)

Chapter 36
Statistical and Computational Geometry of Biomolecular Structure

Iosif I. Vaisman

36.1 Introduction

Recent revolutionary developments in genomics and computational structural biology lead to the rapidly increasing amount of data on biomolecular sequences and structures. The deposition rate for both sequence and structure databases continues to grow exponentially. The efficient utilization of this data depends on the availability of methods and tools for biomolecular data analysis. Significant achievements have been made in DNA and protein sequence analysis, now the focus in bioinformatics research is shifting from sequence to structural and functional data. Accurate prediction of protein three-dimensional structure from its primary sequence represents one of the greatest challenges of modern theoretical biology. Detailed knowledge of protein structure is essential for understanding the mechanisms of biological processes at molecular, cellular, and evolutionary levels. The structures of only a fraction of all known primary sequences have been determined experimentally. Several approaches to protein structure prediction have been developed in recent years. Many of these approaches rely on the knowledge derived from the analysis of significant spatial and compositional patterns in known protein structures and understanding of the role these patterns play in the extremely complex processes, like protein folding or protein function. Such an analysis requires an objective definition of nearest neighbor residues that can be provided by the statistical geometry methods.

In the statistical geometry methods the nearest neighbor atoms or groups of atoms are identified by statistical analysis of irregular polyhedra obtained as a result of a specific tessellation in three-dimensional space. Voronoi tessellation

I.I. Vaisman (✉)
Department of Bioinformatics and Computational Biology, George Mason University, Fairfax, VA, USA
e-mail: ivaisman@gmu.edu

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_36,

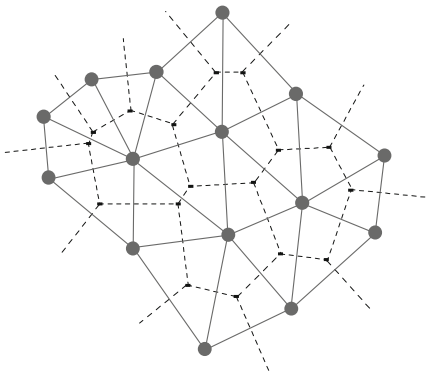


Fig. 36.1 Voronoi/Delaunay tessellation in 2D space; Voronoi tessellation – *dashed line*, Delaunay tessellation – *solid line*

partitions the space into convex polytopes called Voronoi polyhedra (Voronoi 1908). For a molecular system the Voronoi polyhedron is the region of space around an atom, such that each point of this region is closer to the atom than to any other atom of the system. A group of four atoms whose Voronoi polyhedra meet at a common vertex forms another basic topological object called a Delaunay simplex (Delaunay 1934). The results of the procedure for constructing Voronoi polyhedra and Delaunay simplices in two dimensions are illustrated in Fig. 36.1. The topological difference between these objects is that the Voronoi polyhedron represents the environment of individual atoms whereas the Delaunay simplex represents the ensemble of neighboring atoms. The Voronoi polyhedra and the Delaunay simplices are topological duals and they are completely determined by each other. However the Voronoi polyhedra may have different numbers of faces and edges, while the Delaunay simplices are always tetrahedra in three-dimensional space. These tetrahedra can be used to define objectively the nearest neighbor entities in molecular systems.

Delaunay tessellation is a canonical tessellation of space based on nearest neighbors (Aurenhammer 1991; Sugihara and Inagaki 1995). A Delaunay tessellation of a set of points is equivalent to a convex hull of the set in one higher dimension, it can be performed using the Quickhull algorithm developed by Barber et al. (1996). The Quickhull algorithm is a variation of the randomized, incremental algorithm of Clarkson and Shor. The algorithm produces the Delaunay tessellation by computing the convex-hull of this set of points in four dimensions and is shown to be space and time efficient.

36.2 Statistical Geometry of Molecular Systems

A statistical geometry approach to study structure of molecular systems was pioneered by John Bernal, who in the late 1950s suggested that “many of the properties of liquids can be most readily appreciated in terms of the packing of irregular

polyhedra" (Bernal 1959). Bernal pointed out that "it would be most desirable to find the true minimum number of parameters or parametral functions defining the statistical structure of any homogenous irregular assemblage in the way that the lattice vectors define a regular one" (Bernal 1959). Methods of computational geometry, Voronoi and Delaunay tessellations in particular, may be used to address this problem. This approach, based on the Voronoi partitioning of space (Voronoi 1908) occupied by the molecule, was further developed by Finney for liquid and glass studies (Finney 1970). Finney proposed a set of "descriptive parameters" for packing of polyhedra in simple liquids. In the mid-1970s the statistical geometry approach was first applied to study packing and volume distributions in proteins by Richards (1974), Chothia (1975) and Finney (1975).

Richards applied Voronoi tessellation to calculate atomic volumes in the experimentally solved crystallographic structures of ribonulease C and lysozyme (Richards 1974) and Chothia extended the calculations to a larger set of proteins (Chothia 1975). Standard Voronoi tessellation algorithm treats all atoms as points and allocates volume to each atom regardless of the atom size, which leads to the volume overestimate for the small atoms and underestimate for the large ones. Richards introduced changes in the algorithm (Richards 1974) that made Voronoi volumes proportional to the atom sizes, creating chemically more relevant partitioning, however it has been done at the expense of the robustness of the algorithm. The Voronoi polyhedra in this case do not fill all available space. In addition to polyhedra around the atoms Richards' method produces so called vertex polyhedra in the unallocated volumes in the neighborhood of each vertex. As a result the accuracy of the tessellation is reduced. An alternative procedure, the "radical plane" partitioning, which is both chemically appropriate and completely rigorous was designed by Gellatly and Finney (1982) and applied to the calculation of protein volumes. The radical plane of two spheres is the locus of points from which the tangent lengths to both spheres are equal. Using the three-dimensional structure of RNAase-C as an example, they have shown that the differences between the results from the radical plane and Richards' methods are generally smaller than the difference between either of those and Voronoi's method. Both radical plane and Richards' methods are relatively insensitive to the treatment of surface, which makes them preferential to other techniques (Gellatly and Finney 1982). Volume calculation remains one of the most popular applications of Voronoi tessellation to protein structure analysis. It has been used to evaluate the differences in amino acid residue volumes in solution and in the interior of folded proteins (Harpaz et al. 1994), to monitor the atomic volumes in the course of molecular dynamics simulation of a protein (Gerstein et al. 1995), to compare the sizes of atomic groups in proteins and organic compounds (Tsai et al. 1999), to calculate the atomic volumes on protein-protein interfaces (Lo Conte et al. 1999), and to measure sensitivity of residue volumes to the selection of tessellation parameters and protein training set (Tsai and Gerstein 2002). Deviations from standard atomic volumes in proteins determined through Voronoi tessellation correlate with crystallographic resolution and can be used as a quality measure for crystal structures (Pontius et al. 1996). A modified Voronoi algorithm, where dividing planes between the atoms

were replaced by curved surfaces, defined as the set of geometrical loci with equal orthogonal distance to the surfaces of the respective van der Waals spheres, was proposed by Goede et al. (1997). Voronoi cells with hyperbolic surface constructed by this algorithm improve the accuracy of volume and density calculations in proteins (Rother et al. 2003). Another extension of the Voronoi algorithm, the Laguerre polyhedral decomposition was applied to the analysis of the residue packing geometry (Sadoc et al. 2003).

One of the problems in constructing Voronoi diagram for the molecular systems is related to the difficulty of defining a closed polyhedron around the surface atoms, which leads to ambiguities in determining their volumes and densities (a recent example in Quillin and Matthews 2000). This problem can be addressed by the "solvation" of the tessellated molecule in the at least one layer of solvent or by using computed solvent-accessible surface for the external closures of Voronoi polyhedra. The analysis of atomic volumes on the protein surface can be used to adjust parameters of the force field for implicit solvent models, where the solvent is represented by the generalized Born model of electrostatic salvation which require knowledge of the volume of individual solute atoms (Schaefer et al. 2001). Interactions between residues in proteins can be measured using the contact area between atoms defined as the area of intersection of the van der Waals sphere and the Voronoi polyhedron of an atom (Wernisch et al. 1999). Examining the packing of residues in proteins by Voronoi tessellation revealed a strong fivefold local symmetry similar to random packing of hard spheres, suggesting a condensed matter character of folded proteins (Soyer et al. 2000). Correlations between the geometrical parameters of Voronoi cells around residues and residue conformations were discovered by Angelov et al. (2002). Another recent study described application of Voronoi procedure to study atom-atom contacts in proteins (McConkey et al. 2002).

A topological dual to Voronoi partitioning, the Delaunay tessellation (Delaunay 1934) has an additional utility as a method for non-arbitrary identification of neighboring points in the molecular systems represented by the extended sets of points in space. Originally the Delaunay tessellation has been applied to study model (Voloshin et al. 1989) and simple (Medvedev et al. 1987) liquids, as well as water (Vaisman et al. 1993) and aqueous solutions (Vaisman and Berkowitz 1992; Vaisman et al. 1994). The Delaunay tessellation proved to be a particularly convenient and efficient descriptor of water structure, where a natural tetrahedral arrangement of molecules is present in the first hydration shell (Vaisman et al. 1993).

The first application of the Delaunay tessellation for identification of nearest neighbor residues in proteins and derivation of a four-body statistical potential was developed in the mid-1990s (Singh et al. 1996). This potential has been successfully tested for inverse protein folding (Tropsha et al. 1996), fold recognition (Zheng et al. 1997), decoy structure discrimination (Krishnamoorthy and Tropsha 2003; Munson and Singh 1997), protein design (Weberndorfer et al. 1999), protein folding on a lattice (Gan et al. 2001), mutant stability studies (Carter et al. 2001), computational mutagenesis (Masso and Vaisman 2003), protein structure similarity comparison (Bostick and Vaisman 2003), and protein structure classification (Bostick et al.

2004). Statistical compositional analysis of Delaunay simplices revealed highly nonrandom clustering of amino acid residues in folded proteins when all residues were treated separately as well as when they were grouped according to their chemical, structural, or genetic properties (Vaisman et al. 1998). A Delaunay tessellation based alpha-shape procedure for protein structure analysis was developed by Liang et al. (Liang et al. 1998a). Alpha shapes are polytopes that represent generalizations of the convex hull. A real parameter alpha defines the "resolution" of the shape of a point set (Edelsbrunner et al. 1983). Alpha shapes proved to be useful for the detection of cavities and pockets in protein structures (Liang et al. 1998b,c). Several alternative Delaunay and Voronoi based techniques for cavity identification were described by Richards (1985), Bakowies and van Gunsteren (2002) and Chakravarty et al. (2002). Delaunay tessellation has been also applied to compare similarity of protein local substructures (Kobayashi et al. 1997) and to study the mechanical response of a protein under applied pressure (Kobayashi et al. 1997).

36.3 Tetrahedrality of Delaunay Simplices as a Structural Descriptor in Water

Quantitative measurement of local structural order in the computational models of liquid water (and other associated liquids) is an intrinsically difficult problem. Conventional structure descriptors, such as radial distribution functions cannot be used to adequately evaluate structure in the specific regions of complex model systems like multicomponent solutions or solutions of large biological molecules (Vaisman and Berkowitz 1992). Another set of structural descriptors, the geometric and thermodynamic parameters of hydrogen bond network depend on arbitrary values in the hydrogen bond definition (Vaisman et al. 1994). Statistical geometry enables a robust and accurate approach to addressing the problem of structure characterization in water. The snapshot conformations from the molecular simulation of water or aqueous solution by molecular dynamics, Monte Carlo, or other method can be easily tessellated and geometrical parameters of the resulting Delaunay simplices can be measured. Tetrahedrality is a quantitative measure of the degree of distortion of the Delaunay simplices from the regular tetrahedra, that was introduced by Medvedev et al. (1987) for simple liquids, but can be easily extended to water and other systems. Tetrahedrality is calculated as:

$$T = \sum_{i>j} \left(l_i - l_j\right)^2 / 15\bar{l}^2, \tag{36.1}$$

where l_i is the length of the i-th edge, and $\bar{l}$ is the mean length of the edges of the given simplex. For a regular tetrahedron with four equilateral triangular faces, $T = 0$. For any irregular tetrahedron, $T > 0$. In case of a simulated molecular system the tessellation produces a large number of Delaunay simplices for each

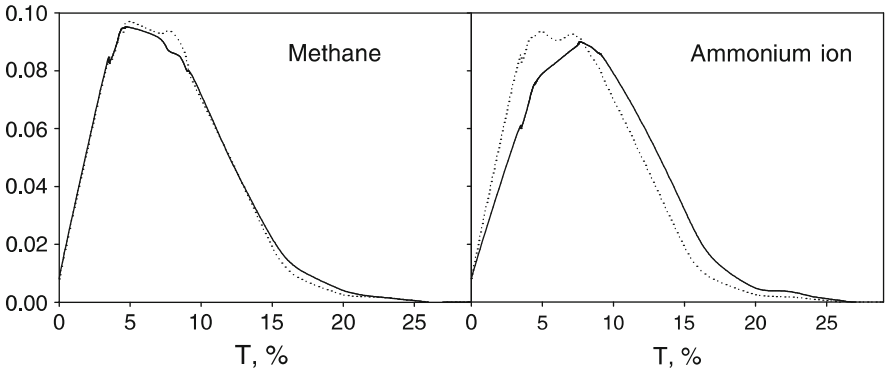


Fig. 36.2 Distribution of tetrahedrality of water around solutes; *solid line* – first hydration shell, *dotted line* – bulk water

snapshot conformation, and a number of such conformations can be very large as well. If the simulated system is at equilibrium, the ergodic theorem applies, and time averages along a system trajectory can be combined with ensemble averages over the phase space. Such a combination increases the number of simplices for the analysis by several orders of magnitude (10^3–10^4 simplices in a conformation multiplied by 10^3–10^4 conformations), which affords high statistical reliability of the results. The nature of this descriptor allows to calculate it separately in confined or limited regions of the simulation system, e.g., in concentric spherical layers around a solute.

The distribution of water tetrahedrality in different layers around solutes depend on the nature of the solute. In the case of charged ions, like ammonium, the difference between tetrahedrality of bulk water and the ammonium hydration water is particularly strong due to the strong hydrogen bonding between water and solute. The peak of the distribution of the tetrahedrality of the ammonium hydration water is shifted to the right which indicates that the hydration water is less tetrahedral than bulk water (Fig. 36.2). Conversely, water in the first hydration shell of methane is just slightly more tetrahedral than the bulk water. Thus, the hydration water around ammonium is significantly more distorted than that around methane as one could expect in the case of hydrophilic and hydrophobic hydration, respectively (Vaisman et al. 1994).

It is worth to note that the presence of hydrophobic solute changes the distribution of water tetrahedrality in the same way as the decrease of temperature. This observation is consistent with the concept of the decrease of ‘structural temperature’ of water, surrounding hydrophobic particles, that has been discussed in the literature for a long time. The decrease in the structural temperature corresponds to the increased structural order of water, because any structural characteristic of liquid must be a monotonically decreasing function of temperature. Distribution of tetrahedrality entirely complies with this requirement.

The influence of both solutes on the water tetrahedrality is almost unobservable beyond the first hydration shell. The distribution of tetrahedrality for both solutions

is similar at both cutoff radii (Vaisman et al. 1994). This result indicates that the distribution of tetrahedrality is not sensitive to the treatment of long-range interactions. Distribution of tetrahedrality beyond the first hydration shell is similar to that in pure water.

36.4 Spatial and Compositional Three-dimensional Patterns in Proteins

Delaunay simplices obtained as a result of the tessellation can be used to define objectively the nearest neighbor residues in 3D protein structures. The most significant feature of Delaunay tessellation, as compared with other methods of nearest neighbor identification, is that the number of nearest neighbors in three dimensions is always four, which represents a fundamental topological property of 3D space. Statistical analysis of the amino acid composition of Delaunay simplices provides information about spatial propensities of all quadruplets of amino acid residues clustered together in folded protein structures. The compositional statistics can be also used to construct four-body empirical contact potentials, which may provide improvement over traditional pairwise statistical potentials (e.g., Miyazawa and Jernigan 2000) for protein structure analysis and prediction.

To perform the tessellation protein residues should be represented by single points located, for example, in the positions of the C_α atoms or the centers of the side chains. Tessellation training set includes high-quality representative protein structures with low primary-sequence identity (Wang and Dunbrack 2003). The tessellated proteins are analyzed by computing various geometrical properties and compositional statistics of Delaunay simplices.

An example of Delaunay tessellation of a folded protein is illustrated on Fig. 36.3 for crambin (1 crn). The tessellation of this 46-residue protein generates

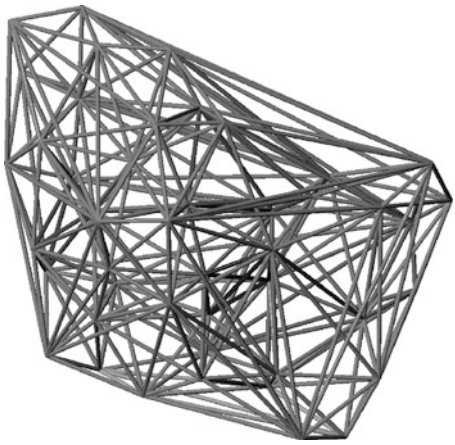


Fig. 36.3 Delaunay tessellation of crambin

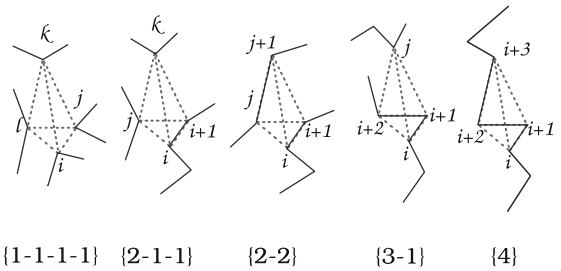


Fig. 36.4 Five classes of Delaunay simplices

an aggregate of 192 nonoverlapping, space-filling irregular tetrahedra (Delaunay simplices). Each Delaunay simplex uniquely defines four nearest neighbor C_α atoms and thus four nearest neighbor amino acid residues.

For the analysis of correlations between the structure and sequence of proteins, we introduced a classification of simplices based on the relative positions of vertex residues in the primary sequence (Singh et al. 1996). Two residues were defined as distant if they were separated by one or more residues in the protein primary sequence. Simplices were divided into five nonredundant classes: class $\{4\}$, where all four residues in the simplex are consecutive in the protein primary sequence; class $\{3, 1\}$, where three residues are consecutive and the fourth is a distant one; class $\{2, 2\}$, where two pairs of consecutive residues are separated in the sequence; class $\{2, 1, 1\}$, where two residues are consecutive, and the other two are distant both from the first two and from each other; and class $\{1, 1, 1, 1\}$ where all four residues are distant from each other (Fig. 36.4). All five classes usually occur in any given protein.

The differences between classes of simplices can be evaluated using geometrical parameters of tetrahedra such as volume and tetrahedrality (36.1). Distributions of volume and tetrahedrality for all five classes of simplices is shown in Fig. 36.5. The sharp narrow peaks correspond to the simplices of classes $\{4\}$ and $\{2, 2\}$. They tend to have well defined distributions of volume and distortion of tetrahedrality. These results suggest that tetrahedra of these two classes may occur in regular protein conformations such as α-helices and may be indicative of a protein fold family. We have calculated the relative frequency of occurrence of tetrahedra of each class in each protein in a small dataset of hundred proteins from different families and plotted the results in Fig. 36.6. The proteins were sorted in the ascending order of fraction of tetrahedra of class $\{4\}$. Noticeably, the content of simplices of class $\{3, 1\}$ decreases with the increase of the content of class $\{4\}$ simplices. According to common classifications of protein fold families (Orengo et al. 1997), at the top level of hierarchy most proteins can be characterized as all-alpha, all-beta, or alpha/beta. The fold families for the proteins in the dataset are also shown in Fig. 36.6. These results suggest that proteins having a high content of tetrahedra of classes $\{4\}$ and $\{2, 2\}$ (i.e., proteins in the right part of the plot in Fig. 36.6) belong to the family of all-alpha proteins. Similarly, proteins having a low content of tetrahedra of classes $\{4\}$ and $\{2, 2\}$ but a high content of tetrahedra of classes $\{2, 2\}$ and $\{3, 1\}$ (i.e., proteins in the left part of the plot in Fig. 36.6) belong to the all-beta protein

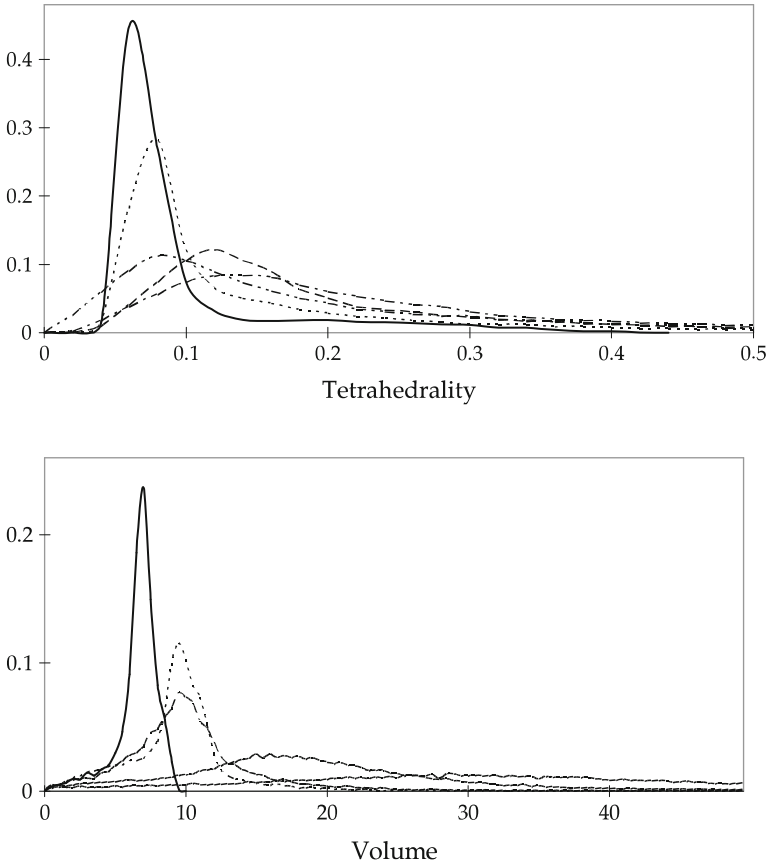


Fig. 36.5 Distribution of tetrahedrality and volume (in $Å^3$) of Delaunay simplices in proteins

fold family. Finally, proteins in the middle of the plot belong to the alpha/beta fold family. Thus, the results of this analysis show that the ratio of tetrahedra of different classes is indicative of the protein fold family.

Identification of significant patterns in biomolecular objects depends on the possibility to distinguish what is likely from what is unlikely to occur by chance (Karlin et al. 1991). Statistical analysis of amino acid composition of the Delaunay simplices provides information about spatial propensities of all quadruplets of amino acid residues to be clustered together in folded protein structures. We analyzed the results of the Delaunay tessellation of these proteins in terms of statistical likelihood of occurrence of four nearest neighbor amino acid residues for all observed quadruplet combinations of 20 natural amino acids. The log-likelihood factor, q, for each quadruplet was calculated using the following equation:

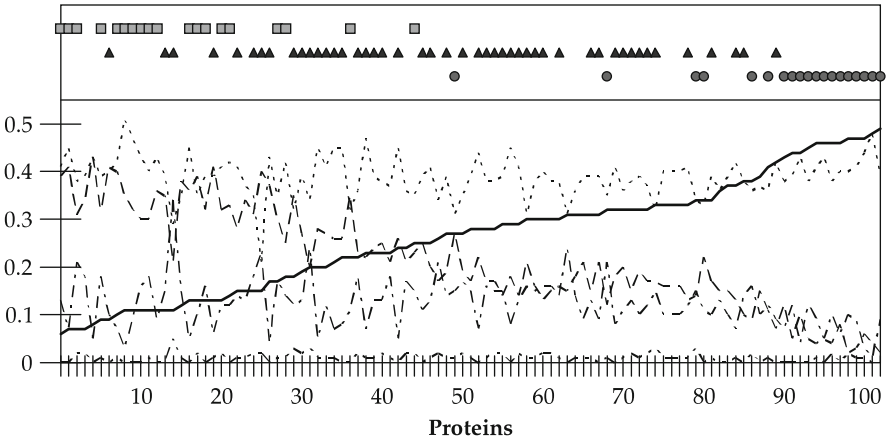


Fig. 36.6 Classes of Delaunay simplices and protein fold families. Contents of simplices of class {4} (*solid line*), class {3, 1} (*dashed line*), class {2, 1} (*dotted line*), class {2, 1} (*dash-dotted line*), class {1, 1, 1, 1} (*dash-dot-dotted line*). *Upper part* of the figure displays fold family assignment: all-alpha (*circles*), all-beta (*squares*), and alpha-beta (*triangles*)

$$q_{ijkl} = \log \frac{f_{ijkl}}{p_{ijkl}} \tag{36.2}$$

where i, j, k, l are any of the 20 natural amino acid residues, f_{ijkl} is the observed normalized frequency of occurrence of a given quadruplet, and p_{ijkl} is the randomly expected frequency of occurrence of a given quadruplet. The q_{ijkl} shows the likelihood of finding four particular residues in one simplex. The f_{ijkl} is calculated by dividing the total number of occurrence of each quadruplet type by the total number of observed quadruplets of all types. The p_{ijkl} was calculated from the following equation:

$$p_{ijkl} = C a_i a_j a_k a_l \tag{36.3}$$

where a_i, a_j, a_k, and a_l are the observed frequencies of occurrence of individual amino acid residue (i.e. total number of occurrences of each residue type divided by the total number of amino acid residues in the dataset), and C is the permutation factor, defined as

$$C = \frac{4!}{\prod_i^n (t_i!)} \tag{36.4}$$

where n is the number of distinct residue types in a quadruplet and t_i is the number of amino acids of type i. The factor C accounts for the permutability of replicated residue types.

Theoretically, the maximum number of all possible quadruplets of 20 natural amino acid residues is 8,855 ($C_{20}^4+3C_{20}^3+2C_{20}^2+C_{20}^2+C_{20}^1$). The first term accounts

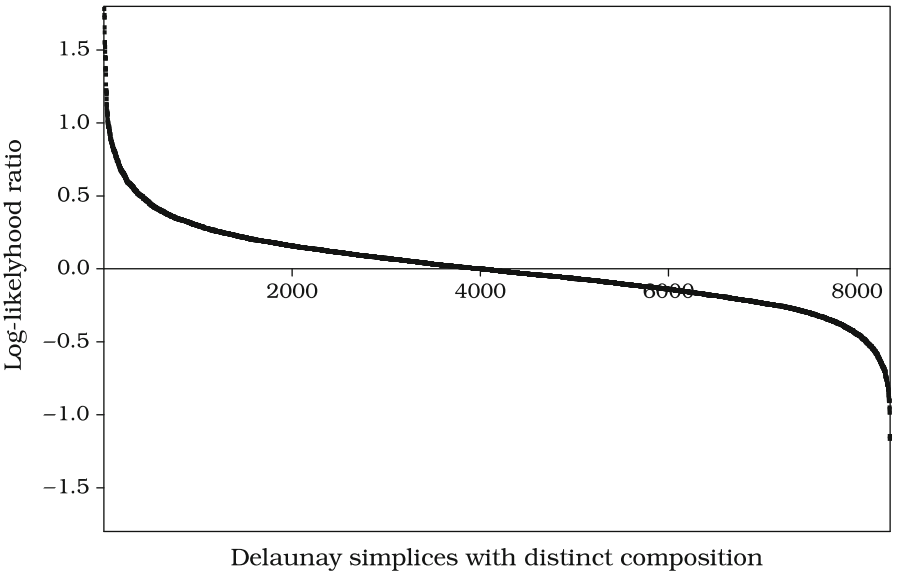


Fig. 36.7 Log-likelihood ratio for the Delaunay simplices

for simplices with four distinct residue types, the second – three types in $1-1-2$ distribution, the third – two types in $1-3$ distribution, the fourth – two types in $2-2$ distribution, and the fifth – four identical residues. The log-likelihood factor q is plotted in Fig. 36.7 for all observed quadruplets of natural amino acids. Each quadruplet is thus characterized by a certain value of the q factor which describes the nonrandom bias for the four amino acid residues to be found in the same Delaunay simplex. This value can be interpreted as a four-body statistical potential energy function. The statistical potential can be used in a variety of structure prediction, protein modeling, and computational mutagenesis applications.

Computational mutagenesis is based on the analysis of a protein potential profile, which is constructed by summing the log-likelihood scores from (36.2) for all simplices in which a particular residue participates. A plot of the potential profile for a small protein, HIV-1 protease, is shown in Fig. 36.8. The shape of the potential profile frequently reflects important features of the protein, for example, the residues in local maxima values of the profile are usually located in the hydrophobic core of the protein and these residues play an important role in maintaining protein stability.

A potential profile can be easily calculated for both wild type and mutant proteins, assuming that the structural differences between them are small and that their tessellation results are similar. In this case the difference between the profiles is defined only by the change in composition of the simplices involving the substitution site. The resulting difference profile provides important insights into the changes in protein energetics due to the mutation.

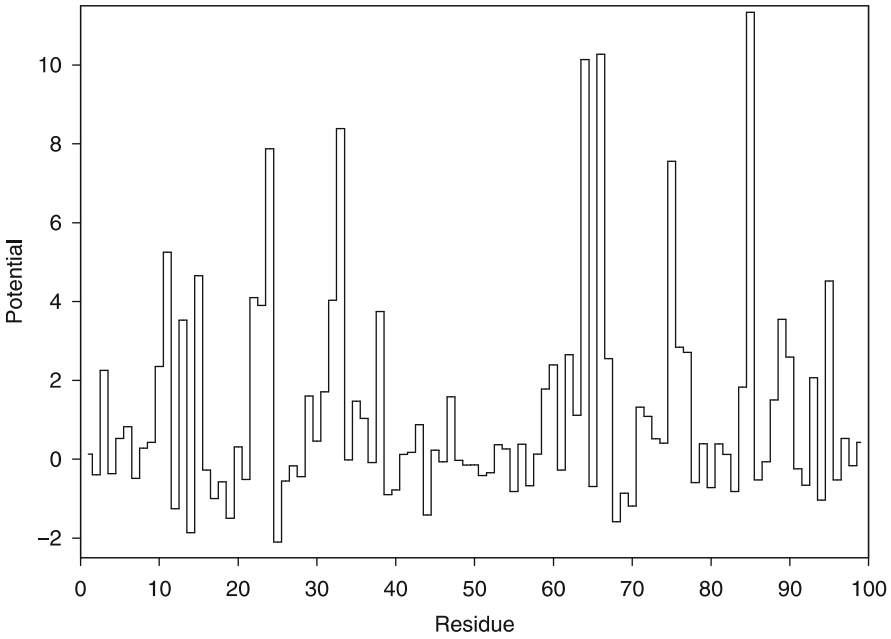


Fig. 36.8 Potential profile of HIV-1 protease

36.5 Protein Structure Comparison and Classification

Using the information from the Delaunay tessellation of a protein's backbone, it is possible to build a statistical representation of that protein, which takes into account the way its sequence must "twist and turn" in order to bring each four-body residue cluster into contact. Each residue – i, j, k, and l of a four-body cluster comprising a simplex are nearest neighbors in Euclidean space as defined by the tessellation, but are separated by the three distances – d_{ij}, d_{jk}, and d_{kl} in sequence space. Based on this idea, we build a 1,000-tuple representation of a single protein by making use of two metrics: (1) the Euclidean metric used to define the Delaunay tessellation of the protein's C_α atomic coordinates and (2) the distance between residues in sequence space.

If we consider a tessellated protein with N residues integrally enumerated according to their position along the primary sequence, the length of a simplex edge in sequence space can be defined as $d_{ij} = j - i - 1$, where d_{ij} is the length of the simplex edge, $\overline{ij}$, corresponding to the ith and jth α-carbons along the sequence. If one considers the graph formed by the union of the simplex edge between the two points i and j and the set of edges between all d_{ij} points along the sequence between i and j, it is seen that the Euclidean simplex edge, $\overline{ij}$, can generally be classified as a far edge (Pandit and Amritkar 1999). Every simplex in the protein's tessellation

will have three such edges associated with its vertices: i, j, k, and l where i, j, k, and l are integers corresponding to C_α atoms enumerated according to their position along the primary sequence. Thus, we proceed to quantify the degree of "farness" in an intuitive way, by applying a transformation, T, which maps the length, d, of each edge to an integer value according to

$$T : d \mapsto \begin{cases} 1 & \text{if } d = 0 \\ 2 & \text{if } d = 1 \\ 3 & \text{if } d = 2 \\ 4 & \text{if } d = 3 \\ 5 & \text{if } 4 \leq d \leq 6 \\ 6 & \text{if } 7 \leq d \leq 11 \\ 7 & \text{if } 12 \leq d \leq 20 \\ 8 & \text{if } 21 \leq d \leq 49 \\ 9 & \text{if } 50 \leq d \leq 100 \\ 10 & \text{if } d \geq 101 \end{cases} \tag{36.5}$$

The reasoning behind the design of the transformation is described by Bostick and Vaisman (2003). This transformation is used to construct an array that is representative of the distribution of combinations of segment lengths along the protein backbone forming four-residue clusters within the protein's structure as defined by the tessellation of its C_α atomic coordinates. Each simplex in the protein's tessellation contributes to a 3D array, M, where M_{npr} is the number of simplices whose edges satisfy the following conditions:

1. The Euclidean length of any one simplex edge is not greater than 10 Å.
2. $d_{ij} = n$
3. $d_{jk} = p$
4. $d_{kl} = r$

Condition 1 is provided because simplices with a Euclidean edge length above 10 Å are generally a result of the positions of α-carbons on the exterior of the protein. We filter out contributions from these simplices to M, because they do not represent physical interactions between the participating residues. The simplices with the long edges are formed due to the absence of solvent and other molecules around the protein in the tessellation, they would not have existed if the protein was solvated. The data structure, M, contains 1,000 elements. The number of elements is invariant with respect to the number of residues of the protein. In order to more easily conceptualize the mapping of the protein topology to the data structure, M, we rewrite it as a 1,000-tuple vector $\overrightarrow{M} = \{M_{000}, M_{001}, \ldots, M_{010}, M_{011}, \ldots, M_{999}\}$.

Given that each element of this vector represents a statistical contribution to the global topology, a comparison of two proteins making use of this mapping must involve the evaluation of the differences in single corresponding elements of the proteins' 1,000-tuples. We define, therefore, a raw topological score, Q,

representative of the topological distance between any two proteins represented by data structures, $\overrightarrow{M}$ and $\overrightarrow{M}'$, as the supremum norm,

$$Q \equiv \left\| \overrightarrow{M} - \overrightarrow{M}' \right\|_{\sup} \equiv \sum_{i=0}^{999} \left| M_i - M_i' \right| . \tag{36.6}$$

This norm is topologically equivalent to the Euclidean norm and has the added advantage that it is less computationally expensive to calculate.

This topological score has an obvious dependence on the sequence length difference between the two proteins being compared due to the following implicit relation for a single protein representation,

$$N_s = \sum_{i=0}^{999} M_i , \tag{36.7}$$

where i is the number of simplices with no edge having a Euclidian length greater than 10 Å, and the M_i are the elements of the protein representation. In other words, since N_s is proportional to the number of residues in the protein, the difference in the length between two compared proteins might provide a systematic extraneous contribution to their score, Q, in (36.6). This is not to say that the sequence length of a protein does not play a role in its topology. In fact, the length should be quite crucial (Bostick et al. 2004). However, the length dependence of our score implied by (36.7) is endemic to our protein representation (derived from its tessellation), and not due to protein topology itself. This length dependence may be removed by first normalizing the vector representation as follows:

$$\underrightarrow{M} = \frac{\overrightarrow{M}}{\left\| \overrightarrow{M} \right\|} \tag{36.8}$$

resulting in the unit-vector representation, $\underrightarrow{M}$. The corresponding normalized topological score,

$$\underline{Q} \equiv \left\| \underrightarrow{M} - \underrightarrow{M} \right\|_{\sup} \tag{36.9}$$

can be expected to be less sensitive to the chain length difference between the two proteins being compared. Despite normalization, however, this score should still have an inherent dependence on the length difference between the compared proteins. A protein's structure must be dependent on the length of its sequence, because the number of configurational degrees of freedom in a polymer's structure is proportional to the number of residues it possesses. Such a dependence on the size of compared proteins is present in geometric methods of comparison such as

structural alignment as well, and in some cases, has been accounted for (Carugo and Pongor 2001).

The results of topological protein structure comparison can be illustrated using an example of proteins that belong to the same family. Six protein families were selected from the FSSP (Families of Structurally Similar Proteins) database for topological evaluation. We selected families that span various levels of secondary structural content. The representatives of these families are as follows: 1alv and 1avm (having greater than 50% α-helical content), 2bbk and 2bpa (having greater than 50% β-sheet content), and 1hfc and 1plc (having at least 50% content that is classified as neither α-helical nor β-sheet). The FSSP database contains the results of the alignments of the extended family of each of these representative chains. Each family in the database consists of all structural neighbors excluding very close homologs (proteins having a sequence identity greater than 70%). The topological score was calculated for each representative in a one-against-all comparison with its neighbors. All of the scores are plotted against RMSD for each of the families in Fig. 36.9. A strong correlation between the topological score and structure similarity and the power-law trend can be seen for all families.

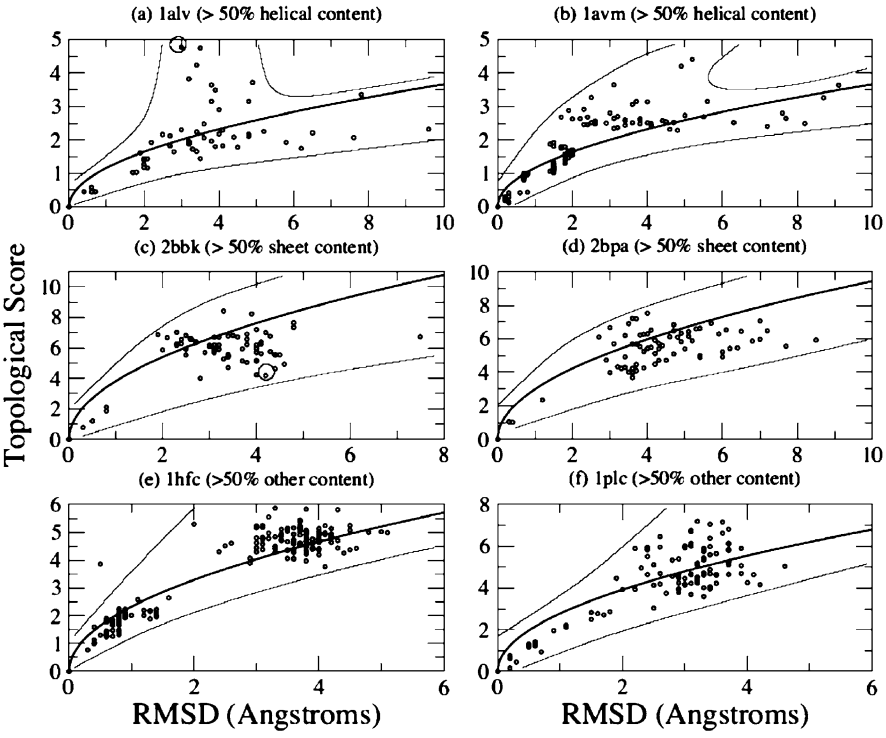


Fig. 36.9 Topological and geometric comparison within 6 protein families

36.6 Conclusions

Methods of statistical and computational geometry, Voronoi and Delaunay tessellation in particular, play an increasingly important role in exploration of complex nature of molecular and biomolecular structure. The range of applications of statistical geometry for biomolecular structural studies has grown significantly in the past decade. As the new experimental structural information on biomolecules becomes available, the need for sophisticated and robust tools for its interpretation will further increase. At the same time more known structure will enable the creation of larger and better training sets for pattern identification. Existing and new statistical geometry algorithms may prove instrumental in future developments of methods for protein structure analysis, ab initio structure prediction, and protein engineering.

References

Angelov, B., Sadoc, J.F., Jullien, R., Soyer, A., Mornon, J.P., Chomilier, J.: Nonatomic solvent-driven Voronoi tessellation of proteins: An open tool to analyze protein folds. Proteins **49**(4), 446–456 (2002)

Aurenhammer, F.: Voronoi diagrams – a survey of a fundamental geometric data structure. ACM Comput. Surv. **23**, 345–405 (1991)

Bakowies, D., van Gunsteren, W.F.: Water in protein cavities: A procedure to identify internal water and exchange pathways and application to fatty acid-binding protein. Proteins **47**(4), 534–545 (2002)

Barber, C.B., Dobkin, D.P., Huhdanpaa, H.T.: The Quickhull algorithm for convex hulls. ACM Trans. Math. Software **22**(4), 469–483 (1996)

Bernal, J.D.: A geometrical approach to the structure of liquids. Nature **183**(4655), 141–147 (1959)

Bostick, D., Vaisman, I.I.: A new topological method to measure protein structure similarity. Biochem. Biophys. Res. Commun. **304**(2), 320–325 (2003)

Bostick, D., Shen, M., Vaisman, I.I.: A simple topological representation of protein structure: Implications for new, fast, and robust structural classification. Proteins 55 (2004)

Carter, C.W. Jr., LeFebvre, B.C., Cammer, S.A., Tropsha, A., Edgell, M.H.: Four-body potentials reveal protein-specific correlations to stability changes caused by hydrophobic core mutations. J. Mol. Biol. **311**(4), 625–638 (2001)

Carugo, O., Pongor, S.: A normalized root-mean-square distance for comparing protein three-dimensional structures. Protein Sci. **10**(7), 1470–1473 (2001)

Chakravarty, S., Bhinge, A., Varadarajan, R.: A procedure for detection and quantitation of cavity volumes proteins. Application to measure the strength of the hydrophobic driving force in protein folding. J. Biol. Chem. **277**(35), 31345–31353 (2002)

Chothia, C.: Structural invariants in protein folding. Nature 1975 **254**(5498), 304–308 (1975)

Delaunay, B.N.: Sur la sphere vide, Izv Akad Nauk SSSR. Otd Mat Est Nauk **7**, 793–800 (1934)

Edelsbrunner, H., Kirkpatrick, D.G., Seidel, R.: On the shape of a set of points in the plane. IEEE Trans. Inform. Theor. **IT-29**(4), 551–559 (1983)

Finney, J.L.: (1970) Random packing and the structure of simple liquids. I. The geometry of random close packing. Proc. Roy. Soc. Lond **A319**(479–493), 495–507 (1970)

Finney, J.L.: Volume occupation, environment and accessibility in proteins. The problem of the protein surface. J. Mol. Biol. **96**(4), 721–732 (1975)

Gan, H.H., Tropsha, A., Schlick, T.: Lattice protein folding with two and four-body statistical potentials. Proteins **43**(2), 161–174 (2001)
Gellatly, B.J., Finney, J.L.: Calculation of protein volumes: an alternative to the Voronoi procedure. J. Mol. Biol. **161**(2), 305–322 (1982)
Gerstein, M., Tsai, J., Levitt, M.: The volume of atoms on the protein surface: calculated from simulation, using Voronoi polyhedra. J. Mol. Biol. **249**(5), 955–966 (1995)
Goede, A., Preissner, R., Frömmel, C.: Voronoi cell: New method for allocation of space among atoms: Elimination of avoidable errors in calculation of atomic volume and density. J. Comput. Chem. **18**(9), 1113–1123 (1997)
Harpaz, Y., Gerstein, M., Chothia, C.: Volume changes on protein folding. Structure **2**(7), 641–649 (1994)
Karlin, S., Bucher, P., Brendel, V.: Altschul S.F., Statistical methods and insights for protein and DNA sequences. Annu. Rev. Biophys. Biophys. Chem. **20**, 175–203 (1991)
Kobayashi, N., Go, N.: A method to search for similar protein local structures at ligand binding sites and its application to adenine recognition. Eur. Biophys. J. **26**(2), 135–144 (1997)
Kobayashi, N., Yamato, T., Go, N.: Mechanical property of a TIM-barrel protein. Proteins **28**(1), 109–116 (1997)
Krishnamoorthy, B., Tropsha, A.: Development of a four-body statistical pseudo-potential to discriminate native from non-native protein conformations. Bioinformatics **19**(12), 1540–1548 (2003)
Liang, J., Edelsbrunner, H., Fu, P., Sudhakar, P.V., Subramaniam, S.: Analytical shape computation of macromolecules: I. Molecular area and volume through alpha shape. Proteins **33**(1), 1–17 (1998a)
Liang, J., Edelsbrunner, H., Fu, P., Sudhakar, P.V., Subramaniam, S.: Analytical shape computation of macromolecules: II. Inaccessible cavities in proteins. Proteins **33**(1), 18–29 (1998b)
Liang, J., Edelsbrunner, H., Woodward, C.: Anatomy of protein pockets and cavities: measurement of binding site geometry and implications for ligand design. Protein Sci. **7**(9), 1884–1897 (1998c)
Lo Conte, L., Chothia, C., Janin, J.: The atomic structure of protein-protein recognition sites. J. Mol. Biol. **285**(5), 2177–2198 (1999)
Masso, M., Vaisman, I.I.: Comprehensive mutagenesis of HIV-1 protease: A computational geometry approach. Biochem. Biophys. Res. Commun. **305**(2), 322–326 (2003)
Medvedev, N.N., Naberukhin, Yu.I.: Analysis of structure of simple liquids and amophous solids by statistical geometry method. Zh Strukt Khimii **28**(3), 117–132 (1987)
Miyazawa, S., Jernigan, R.L.: Identifying sequence-structure pairs undetected by sequence alignments. Protein Eng. **13**(7), 459–475 (2000)
McConkey, B.J., Sobolev, V., Edelman, M.: Quantification of protein surfaces, volumes and atom-atom contacts using a constrained Voronoi procedure. Bioinformatics **18**(10), 1365–1373 (2002)
Munson, P.J., Singh, R.K.: Statistical significance of hierarchical multi-body potentials based on Delaunay tessellation and their application in sequence-structure alignment. Protein Sci. **6**(7), 1467–1481 (1997)
Orengo, C.A., Michie, A.D., Jones, S., Jones, D.T., Swindells, M.B., Thornton, J.M.: CATH – a hierarchic classification of protein domain structures. Structure **5**(8), 1093–1108 (1997)
Pandit, S.A., Amritkar, R.E.: Characterization and control of small-world networks. Phys. Rev. E **60**, 1119–1122 (1999)
Pontius, J., Richelle, J., Wodak, S.J.: Deviations from standard atomic volumes as a quality measure for protein crystal structures. J. Mol. Biol. **264**(1), 121–136 (1996)
Quillin, M.L., Matthews, B.W.: Accurate calculation of the density of proteins. Acta. Crystallogr. D. Biol. Crystallogr. **56**(Pt 7), 791–794 (2000)
Richards, F.M.: The interpretation of protein structures: total volume, group volume distributions and packing density. J. Mol. Biol. **82**(1), 1–14 (1974)
Richards, F.M.: Calculation of molecular volumes and areas for structures of known geometry. Methods Enzymol. **115**, 440–464 (1985)

Rother, K, Preissner, R, Goede, A., Frömmel, C.: Inhomogeneous molecular density: reference packing densities and distribution of cavities within proteins. Bioinformatics **19**(16), 2112–2121 (2003)

Sadoc, J.F., Jullien, R., Rivier, N.: The Laguerre polyhedral decomposition: application to protein folds. Eur. Phys. J. B **33**(3), 355–363 (2003)

Schaefer, M., Bartels, C., Leclerc, F., Karplus, M.: Effective atom volumes for implicit solvent models: comparison between Voronoi volumes and minimum fluctuation volumes. J. Comput. Chem. **22**(15), 1857–1879 (2001)

Singh, R.K., Tropsha, A., Vaisman, I.I.: Delaunay tessellation of proteins: four body nearest-neighbor propensities of amino acid residues. J. Comput. Biol. **3**(2), 213–222 (1996)

Soyer, A., Chomilier, J., Mornon, J.P., Jullien, R., Sadoc, J.F.: Voronoi tessellation reveals the condensed matter character of folded proteins. Phys. Rev. Lett. **85**(16), 3532–3535 (2000)

Sugihara, K., Inagaki, H.: Why is the 3D Delaunay triangulation difficult to construct? Inform. Proces. Lett. **54**, 275–280 (1995)

Tropsha, A., Singh, R.K., Vaisman, I.I., Zheng, W.: Statistical geometry analysis of proteins: implications for inverted structure prediction. Pac. Symp. Biocomput. 614–623 (1996)

Tsai, J., Taylor, R., Chothia, C., Gerstein, M.: The packing density in proteins: standard radii and volumes. J. Mol. Biol. **290**(1), 253–266 (1999)

Tsai, J., Gerstein, M.: Calculations of protein volumes: sensitivity analysis and parameter database. Bioinformatics **18**(7), 985–995 (2002)

Vaisman, I.I., Perera, L., Berkowitz, M.L.: Mobility of stretched water. J. Chem. Phys. **98**(12), 9859–9862 (1993)

Vaisman, I.I., Berkowitz, M.L.: Local structural order and molecular associations in water-DMSO mixtures. Molecular dynamics study. J. Am. Chem. Soc. **114**(20), 7889–7896 (1992)

Vaisman, I.I., Brown, F.K., Tropsha, A.: Distance dependence of water structure around model solutes. J. Phys. Chem. **98**(21), 5559–5564 (1994)

Vaisman, I.I., Tropsha, A., Zheng, W.: Compositional preferences in quadruplets of nearest neighbor residues in protein structures: Statistical geometry analysis. Proc. IEEE Symposia Intell. Syst. 163–168 (1998)

Voloshin, V.P., Naberukhin, Y.I., Medvedev, N.N.: Can various classes of atomic configurations (Delaunay simplices) be distinguished in random close packing of spherical particles?. Molec. Simul. **4**, 209–227 (1989)

Voronoi, G.F.: Nouvelles applications des paramétres continus à la théorie des formes quadratiques. Deuxiéme Mémorie: Recherches sur les paralléloèdres primitifs. J. Reine Angew. Math. **134**, 198–287 (1908)

Wang, G., Dunbrack, R.L. Jr.: PISCES: A protein sequence culling server. Bioinformatics **19**(12), 1589–1591 (2003)

Weberndorfer, G., Hofacker, I.L., Stadler, P.F.: An efficient potential for protein sequence design. Proc. German Conf. Bioinform. 107–112 (1999)

Wernisch, L., Hunting, M., Wodak, S.J.: Identification of structural domains in proteins by a graph heuristic. Proteins **35**(3), 338–352 (1999)

Zheng, W., Cho, S.J., Vaisman, I.I., Tropsha, A.: A new approach to protein fold recognition based on Delaunay tessellation of protein structure. Pac. Symp. Biocomput. 486–497 (1997)

Chapter 37
Functional Magnetic Resonance Imaging

William F. Eddy and Rebecca L. McNamee

37.1 Introduction: Overview and Purpose of fMRI

The 2003 Nobel Prize in Medicine went to Paul Lauterbur and Sir Peter Mansfield for the invention of magnetic resonance imaging (MRI) in the 1970s. Since its invention MRI has rapidly changed the world of medicine; there are currently more than 20,000 MRI scanners in the world and many millions of images are generated by them each year. In the early 1990s, Ogawa et al. (1992), Belliveau et al. (1991) and Kwong et al. (1992) showed that MRI could be used for the detection of brain function. Because the technique is non-invasive and does not require the injection of dyes or radioactive tracers, functional MRI (fMRI), has opened up opportunities that were never before possible for studying the living human brain in its working state.

One of the primary uses for fMRI is the mapping of brain function onto brain structure. This is done by engaging a subject in a specific motor, sensory, or cognitive task while collecting MR images of the brain. The regions of increased activity are presumed to be those which perform the task. A particular example is given in Fig. 37.1.

Although mapping of function to structure is an important use of fMRI, the possibilities of its application for investigating the dynamics of brain function are many. Researchers have recently begun using fMRI to study brain development in both normal and pathological situations (Gaillard et al. 2001). The method can also be used to examine the aging brain (Rosen et al. 2002), as well as to study the brain under situations of learning (Poldrack 2000) and injury (McAllister et al. 1999).

Scientific fields other than psychology and neuroscience are also developing an interest in fMRI research. For example, pharmaceutical applications may use fMRI

W.F. Eddy (✉) · R.L. McNamee
Department of Statistics, Carnegie Mellon University, Pittsburgh, PA, USA
e-mail: bill@stat.cmu.edu

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_37,

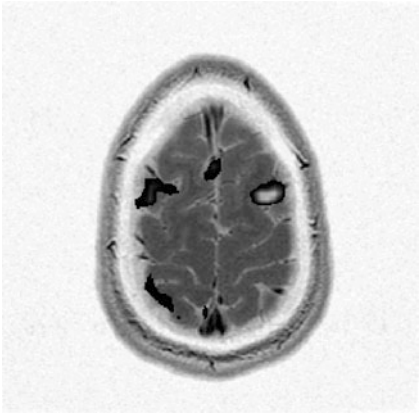


Fig. 37.1 Brain activity while performing a short term memory task, in a high school athlete with mild traumatic brain injury. This single slice shows only a portion of the activity in the entire brain. Because it was derived by thresholding a test statistic there may be both false positive and false negative pixels. The physical contiguity of the regions of activity suggests that there are not any false positives

to investigate the brain before and after the administration of a drug, and geneticists may be interested in how the expression of similar or different genotypes may alter brain functioning in one individual versus another.

As the field of fMRI grows, the problems that it presents for statisticians and other quantitative scientists are also growing. There are several reviews of fMRI work in the statistical literature; see, e.g., Eddy et al. (1999), Lange (1996) and Lazar et al. (2001). While collecting the data from subjects has become easier, the data sets are usually very large (100 MB or more) and are especially variable, containing both systematic and random noise. Storage, processing, and analysis of fMRI data are complicated, and the computational problems are legion. In this chapter a brief background of fMRI is first given from a physics and psychology point of view. A full description of the fMRI data as well as the challenges that it presents from a computational statistics viewpoint are then discussed in detail.

37.2 Background

37.2.1 Magnetic Resonance (MR)

Atomic nuclei spin like gyroscopes. When placed into a magnetic field, atomic nuclei that are affected by magnetism (those having an odd number of protons or neutrons or both) align their axis of rotation with the magnetic field. Like a gyroscope the axis of rotation itself rotates; this rotation is called precession. Each nucleus precesses within the magnetic field. The frequency of this precession, the Larmor frequency, is proportional to the strength of the magnetic field, with the constant of proportionality being determined by the gyromagnetic ratio of the atomic species. The relationship can be described by the Larmor equation

$$\omega_0 = \gamma B_0$$

where ω_0 is the Larmor frequency, γ is the gyromagnetic ratio, and B_0 is the strength of the applied magnetic field.

Hydrogen is currently the most widely used element for MR imaging because of its abundance in biological tissue and the strength of the emitted signal. Hydrogen atoms have a gyromagnetic ratio of approximately 42 MHz per Tesla. A Tesla is a measure of the strength of the magnetic field (B_0) with one Tesla equal to 10,000 gauss. For reference, one-half gauss is roughly the strength of the earth's magnetic field. An MR scanner that is used for functional imaging has a typical field strength of 3 Tesla. The resulting Larmor frequency is about 126 MHz; for comparison a kitchen microwave oven operates at about 2,450 MHz.

Some values of the constants γ for other atomic species can be found at http://www.stat.cmu.edu/~fiasco/index.php?ref=reference/ref_constants.shtml.

As the nuclei of the hydrogen atoms precess within the magnetic field, the atoms will either line up along the field or against it (that is, the atoms will line up at either 0 or 180 °). The strength of the magnetic field and the energy state of the system affect the number of atoms that line up accordingly. Then, while the atoms precess in a steady-state fashion within the magnetic field, a pulse of energy is injected into the system in the form of a transient radio-frequency (rf) pulse perpendicular to the B_0 field at the Larmor frequency (ω_0). This rf pulse excites the atoms at their resonant frequency, causing them to tilt out of alignment with the magnetic field.

As these excited atoms return to equilibrium within the magnetic field they emit rf energy which is collected by an antenna and receiver. Their return to steady-state is known as relaxation, and the signal that the atoms emit is known as the free-induction decay (FID) signal. The FID signal reflects the distribution of hydrogen atoms in the tissue and is used to construct images (see, e.g., Buxton 2002).

37.2.2 Magnetic Resonance Imaging (MRI)

As described, the basic theory of MR can be used to create images based on the distribution of hydrogen atoms or protons in the tissue sample. Other types of atoms can also be imaged. In these cases, the applied rf pulse must be applied at the Larmor frequency of the atoms of interest in the tissue sample.

In order to create images using MR, an FID signal must be encoded for each tissue dimension, and certain considerations must be made to recognize spatial information in the tissue sample from which the FID signals are being recorded. As outlined above, the resonant frequency at which the atoms must be excited to flip them is dependent on the magnetic field strength. Thus, by adjusting the magnetic field strength at certain locations in the tissue and sending rf pulses at the corresponding resonant frequency, only atoms at the location of interest will be excited. In this manner, spatial information can be resolved.

To aid in the understanding of this principle, consider slice selection through a sample of tissue. As shown in Fig. 37.2, the object under consideration (in this case a person) is placed with the xy-slice axis perpendicular to the magnetic field.

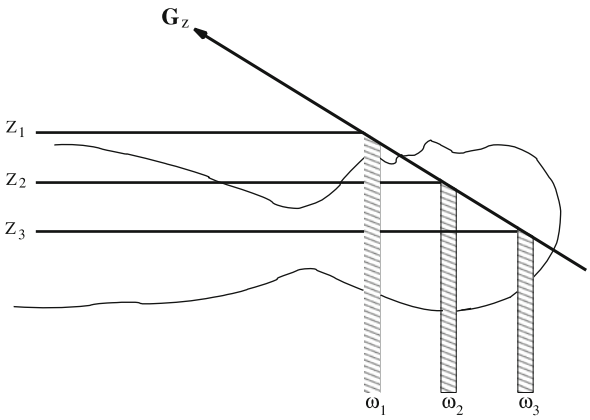


Fig. 37.2 This figure (redrawn from Lazar et al. 2001) is a schematic showing how slice selection takes place

A linear magnetic field gradient is then applied in a direction parallel to the bore of the magnet (z-direction). In this manner, each xy-slice of the tissue is subjected to a slightly different magnetic field strength. If the linear gradient at z is equal to $azB_1 + B_0$, then the Larmor frequency at $z = 1$ becomes ω_1, where:

$$\omega_1 = \gamma(aB_1 + B_0)\,.$$

The magnetic field strength along each slice of the tissue is now slightly different, so the resonant frequency for each slice of the tissue will be slightly different. By adjusting the rf pulse to correspond to ω_1, only slices of interest will be excited and imaged. This same principle can be used to define spatial locations in the xy plane as well. The interested reader should refer, for example, to Buxton's 2002 book *Introduction to Functional Magnetic Resonance Imaging: Principles & Techniques*.

The signal intensity from the tissue is a function of the density of hydrogen atoms or protons in the tissue. The more protons in the tissue, the greater the FID signal, and the higher the intensity value. Because different types of tissues vary in their proton density, tissue contrast can be achieved in the images. Contrast also depends on the tissue specific parameters, longitudinal relaxation time (T_1), and transverse relaxation time (T_2). The amount of time that it takes for the longitudinal magnetization of the tissue to return to 63% of its original value after an rf pulse is applied is denoted T_1, and the time that it takes for the transverse magnetization to return to 37% of its original value after the applied rf pulse is denoted T_2.

The imaging parameters of TR (time of repetition of the rf pulse) and TE (time of echo before FID signal is measured) can be varied to achieve the maximum contrast for the tissues of interest by considering their T_1 and T_2 properties. This process is also known as weighting or contrasting, and images are usually T_1-weighted, T_2-weighted, or proton density weighted. For example, short TRs and TEs lead to T_1 weighted images, because at this time the differences due to the T_1 tissue properties will be the most apparent. In T_1 weighted images, the intensity of bone tissue is typically bright while fluids are dark.

On the other hand, to achieve the best contrast using T_2 properties, a long T_R and a long T_E are used. In T_2 weighted images, bone tissue is dark and fluid areas are light. Proton density weighting is achieved by using a long T_R and a short T_E. With this type of weighting, areas that have the highest proton density are the brightest. These areas would include the cerebral spinal fluid (CSF) and gray matter. Again, see Buxton (2002) for details; a statistical approach is given in Glad and Sebastiani (1995).

37.2.3 Functional MRI

Early Brain Research

The mysteries of the human brain have perplexed researchers for many centuries. Early ideas about brain functioning date at least as far back as the second century to Galen (and even earlier), who associated imagination, intellect, and memory with brain substance. The notion that the brain consisted of functionally discrete areas did not become an accepted idea until the nineteenth century with the work of Franz Joseph Gall (Finger 1994). Ensuing research involved examining the relationships between the location of brain lesions and deficits and/or changes in behavior as a way to attribute brain function to structure. Although the technique was effective, this method for studying the brain was not without limitations. Since that time, however, the field of neuroscience has grown because of the development of new methods to explore the human brain in its living working state. These new techniques have been given the general term functional neuroimaging.

Functional Neuroimaging

Functional neuroimaging is the term applied to techniques that can map the activity of the living working brain in space and time. Non-invasive approaches to this mapping have included electrophysiological measurements and metabolic measurements. Techniques to measure the electrophysiological signals include the electroencephalogram (EEG) and the magnetoencephalogram (MEG) (National Research Council, 1992). These methods are thought to record (a weighted integral of) the actual neural activity that is taking place in the brain. Although both EEG and MEG have excellent temporal resolution, in their most common form the measured output signals are an integration of activity from many possible unknown sources. Furthermore, for EEGs these integrated signals are only realized after being filtered through layers of blood vessels, fat, and bone. On the other hand, MEG generally only measures the component of the magnetic field which is perpendicular to the surface of the skull. Both methods typically record only a few hundred different locations; a typical functional MRI study measures the signal at 100,000 locations or so. Thus, the spatial resolution of both EEG and MEG is quite poor. Source

localization is a research area devoted to trying to map the locations at which these signals originate, but this has proven to be a very difficult task.

Functional neuroimaging measurements also include Positron Emission Tomography (PET) and fMRI. Both of these techniques have good spatial resolution, but unlike EEG and MEG they record responses to changes in blood flow rather than the direct neural activity. Because of this, these techniques have relatively poor temporal resolution.

PET imaging is carried out by labeling molecules of compounds of interest with positron-emitting isotopes. Isotopes that are often used include Carbon-11, Nitrogen-13, or Oxygen-15. These labeled modules are termed "probes" or "tracers". The tracers are distributed in the brain according to their delivery, uptake, metabolism, and excretion properties. As these isotopes decay they emit a positron and a neutrino. The neutrino cannot be detected, but each positron collides with an electron and is annihilated. The annihilation converts the electron and positron from mass into energy in the form of gamma rays. These gamma rays are then detected by the scanner. PET can provide excellent measures of metabolic activity of the brain under conditions of normal and abnormal functioning and has therefore been a highly useful tool in studying brain activity. However, one of the main disadvantages of PET is that it requires the injection of ionizing radiation thereby limiting its use for human subject populations (Cherry and Phelps 1996).

Functional MRI

Functional MRI uses the same principles as MRI, but it is based on the idea that the magnetic state of the hemoglobin in the bloodstream is dependent on whether or not oxygen is bound to it. Deoxygenated blood is paramagnetic, meaning that the unpaired heme groups cause it to have more magnetic susceptibility than it does when oxygen is attached to the heme groups. In fact, the magnetic susceptibility of blood has been shown to vary linearly with blood oxygenation; see, Thulborn et al. (1982), Buxton (2002), Turner (1995), or Weisskoff and Kiihne (1992).

When neurons in the brain are used, their metabolic demands increase. This begins with an increase in local glucose consumption and leads to an increase in local cerebral blood flow. However, for reasons that are unclear, the increase in cerebral blood flow exceeds the increase in metabolic consumption of local oxygen. Therefore the ratio of oxygenated blood to deoxygenated blood increases in the local blood vessels. The change in this ratio of oxygenated blood to deoxygenated blood leads to changes in the local MR signal. Modulations in the MR signal due to this phenomenon can be detected by the scanner and are known as Blood Oxygen Level Dependent (BOLD) contrast. BOLD contrast is currently the main basis of fMRI; see, e.g., Villringer (2000), Logothetis (2002), Ogawa et al. (1990), Buxton (2002), or Turner (1995).

Several informational limitations are imposed by fMRI that should be considered when carrying out a neuroimaging study. Leading these is the fact that fMRI is an indirect measure of brain activity, and its exact physiological mechanism is not

Fig. 37.3 Model of brain activity and fMRI data

known. A model of the interface between actual brain activity and the fMRI signal is shown in Fig. 37.3. Also, the measured activity obtained with fMRI can include many types of local brain cells because the activation that is measured is essentially a combination of all the "brain activity" in the area. Information is thus blurred during fMRI, since the resolution is based on the "smallest measurable vascular unit." Finally, as mentioned earlier, fMRI has poor temporal resolution; see, e.g., Villringer (2000).

In spite of these limitations, fMRI has many advantages over previously used methods for studying the brain. fMRI has much better spatial resolution than EEG or MEG. In fact, although the activity is lumped into small regions, these regions can provide accuracy in the range of 1 mm or so. Next, and perhaps most importantly, fMRI does not require the use of ionizing radiation. This allows it to be used experimentally for many different subject types and under many different types of situations. Other benefits include the fact that the data can be collected fairly easily, and analysis tools are becoming more readily available as the field grows.

37.3 fMRI Data

37.3.1 Design of an fMRI Experiment

There are at least three aspects to the design of an fMRI experiment: (1) the underlying psychological question, (2) the MR physics that will drive the data collection, and (3) traditional statistical design issues. These factors are not exclusive of each other and must be carefully considered prior to the initiation of the study.

The psychological component of design depends on the type of experimental subjects in question as well as the nature and location of the expected response. For example, regions of brain activity could be explored for a single group of subjects, or the location and extent of brain activation could be compared in two different subject groups. The experimental task must also be designed in order to elicit a functional response specific to the area of interest in the brain. A control state (such as a fixation in a visually guided saccade task) is typically alternated with a functional state (the saccade) in order to compare the two states and find the differentially active brain areas.

The MR physics component depends on the nature of the psychological stimuli, the particular MR scanner being used, and the location of the expected response. Several scanning parameters, such as the number and orientation of slices to be collected, the echo time, T_E, the time of repetition, T_R, the flip angle, and others, must be first chosen. The physics of the scan will depend on these chosen parameters. The physical method for collecting each slice of data, termed the pulse sequence, must also be selected. The pulse sequence is a small program run in the scanner to manipulate the magnetic and rf fields, and is thus dependent on the type of MR scanner and the pulse sequences available for use.

The statistical aspects of design will help to determine how the data will be analyzed after collection. For example, as mentioned above, the experimental design is often set so that the task state is alternated with a control state. The two conditions can then be statistically compared using methods which rely on hypothesis testing (such as t-tests). This type of experimental design is called a block design. The block design is robust in that many repetitions of the two conditions can be carried out, and the experimenter can then average all trials for each voxel. Although this technique can find spatial areas of activation, it has the disadvantage of losing most temporal aspects of the data.

In order to capture the time-dependent features of the data as well as the spatial aspects, single trial fMRI experiments have recently become popular. These experiments examine fMRI signal changes over time as the task is being performed (on a voxel-by-voxel basis) rather than relying on the averaging of large time blocks. The results from these single trial experiments are generally not as robust as those for block designs. Furthermore, since fMRI data is typically very noisy, the data must be pre-processed prior to analysis. The techniques for analyzing single trial fMRI data often model those used for processing evoked potentials EEG data (such as filtering or time-averaging of trials). Because of this, single trial fMRI experiments have often been mislabeled as "event-related" fMRI experiment. Figure 37.4 shows the temporal BOLD response that is generally expected from these types of experiments.

37.3.2 Data Collection

The raw data from an MR scanner is spatial frequency data. That is, the data are the coefficients of the Fourier representation of the object being imaged. Alternatively we can say that the data are the inverse Fourier transform of the object. The spatial frequency domain has been called "Fourier Space", "frequency space", or most popularly "k-space". The process of taking the inverse Fourier transform to obtain the image has been termed "data reconstruction". Letting $\mathcal{F}$ be the Fourier transform, we have for an $n \times m$ pixel image I,

$$\mathcal{F}(I) = \widehat{I}\left(k_x, k_y\right) = n^{-1}m^{-1}\sum_{x}^{n}\sum_{y}^{m} I(x, y)\exp(-i2\pi(xk_x + yk_y))$$

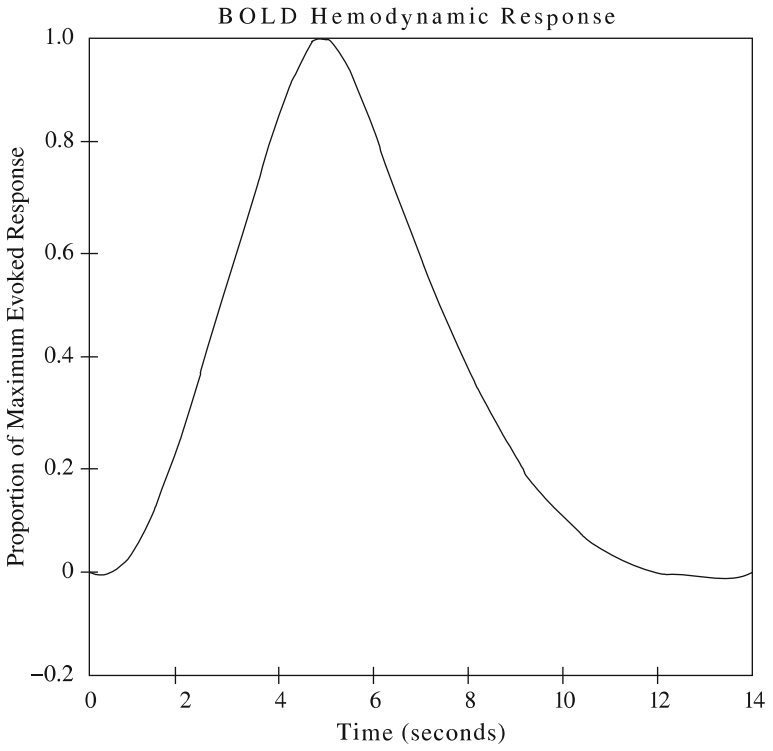


Fig. 37.4 BOLD hemodynamic response curve showing the expected contrast changes elicited from a single event. Because the contrast changes are actually due to blood flow changes rather than a direct measure of neural activity, the time scale over which they occur is relatively slow

In k-space, the low frequencies are located in the center of the image with the frequencies increasing outward with distance from the origin. In a typical k-space plot (see Fig. 37.5), the bulky features of the image lie in the lower frequencies of k-space while the higher frequencies contain the details of the image. In fMRI both the low and high frequency information are important, and the pulse sequence should be designed accordingly.

A typical fMRI data set might consist of a 128 by 128 array of 16 bit complex values recorded for each of 32 two-dimensional slices at each of 450 time points spaced about 1.5 s apart. This yields a data set of $2 \times 2 \times 128 \times 128 \times 32 \times 450 = 943718400$ bytes, that is approximately 1 gigabyte of data collected in less than 12 minutes. If many experiments are performed on a single subject within the period of an hour or so, and several subjects are examined over time, the necessary storage requirements can become quite extensive. In one of our current studies, we anticipate collecting a total of about 700 GB of data. To help deal with this quantity, offline data storage systems are useful. For example, optical disks, CDs, or DVDs can be used to store large amounts of data with minimal effort.

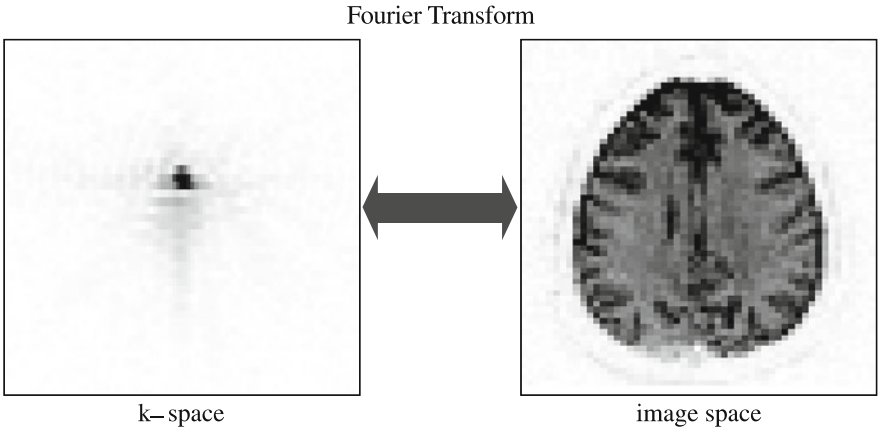


Fig. 37.5 Collected fMRI data. The plot on the *left* shows the modulus of the k-space data, and the plot on the *right* shows the modulus of the image. Darker pixels indicate larger values (the opposite of the "radiological convention" derived from X-ray images on photographic film)

37.3.3 Sources of Bias and Variance in the Data

Areas of brain activity that are found due to specific tasks are dependent on the image to image changes in the measurements within a voxel. Therefore, to produce valid results these changes must be specifically attributable to functional differences in the brain elicited by the performed tasks. Unfortunately, fMRI data is beset with many sources of bias and variability, which can lead to erroneous regions of brain activity and false conclusions about the study. Problems in the data can arise from many sources including the MR scanner itself, the experimental subject, and external interference. Each of these will be discussed with a brief description of the errors that they introduce into the data. The sources of noise in fMRI data can be quite extensive. Although many are covered here, this summary is not exhaustive.

Noise from the Equipment

One main source of bias and systematic variation in fMRI data arises from the MR scanner. The performance of an MR scanner can vary, which can introduce fluctuations in the data, even when the stability measures are well within the instrumental norms (Weisskoff 1996). Noise from the equipment can occur as systematic or random errors.

Sources of systematic error in the data from the equipment include DC shifts and Nyquist ghosts. DC shifts are also known as baseline errors. This source of data bias is caused by the miscalibration of the analog-to-digital (A/D) converter; the baseline value is not reported as zero. Nyquist ghosts, which are present only in echo-planar imaging, also produce systematic bias in the data. Echo-planar pulse sequences traverse k-space on a boustrophedonic path (back-and-forth as the

ox plows the field). Nyquist ghosts are introduced through the mistiming of the oscillating magnetic gradients. The exact time at which the gradient crosses zero is incorrect. This timing error causes an aliasing effect in the reconstructed image and is most prominent in the phase-encode or y direction of the fMRI scan (leading to a ghost of the image repeated at the top and bottom of the true image). Both DC shift errors and Nyquist ghosts that are present in the fMRI data can be corrected to a reasonable extent.

Random errors from the equipment can also cause introduce problems in the fMRI data. One source of unpredictable instability results from inhomogeneities in the static magnetic field of the equipment. Magnetic field inhomogeneities have been reported as one of the most prominent sources of distortion in fMRI studies (Jezzard 1999). Local variations in the static magnetic field during fMRI will lead to blurring and pixel shifts, which can introduce gross geometric distortions in the images. This problem is especially prominent at regional boundaries in the sample containing different magnetic susceptibility properties, for example, air-tissue interfaces around the frontal lobes and bone-tissue interfaces (Eden and Zeffiro 1997; Jezzard 1999).

Additionally, random instability in the MR machine can result from imperfections in the B1 field. The B1 field is ideally a linear magnetic gradient that selects certain regions of tissue to be excited, thereby leading to the collection of single slices. Again, problems with this linear magnetic field can lead to blurring and geometric distortions in the data.

Noise from the Experimental Subject

As with other types of human studies, the experimental subjects can lead to large amounts of bias and variability in the data. While the subjects themselves have a great deal of intrinsic variability due to differences in brain sizes, shapes, and functionality in general, the subjects can also introduce additional variability that will "drown out" the desired results from brain activity if the investigator is not careful.

One important source of noise from the experimental subject is due to head motion. As previously described, BOLD fMRI studies compare very small regions of brain tissue across a sequence of images that are taken over the course of several minutes. While BOLD has the advantage that it requires no exogenous contrast agents, its measurable effects are very small. Typical changes in the MR signal due to BOLD are on the order of $1-5\%$, making this technique highly susceptible to noise. If the subject makes a small movement during the scan, adjacent voxels, which can vary in signal value by more than 10%, cause distortions in the recorded signal information and can lead to false negative and false positive regions of activation (Eddy et al. 1996b; Eddy and Young 2000).

Thus, to obtain valid fMRI data, the subject must remain motionless throughout the scanning period. Motion has been shown to be correlated with stimulus related events during visual and motion stimulation, thereby contributing to the likelihood that the computed regions of activation are due to motion artifact rather than neural

activity (Hajnal et al. 1994). The amount of subject motion has also been shown to increase over time during the course of a scanning session (Green et al. 1994). Additionally, children, elderly subjects, and subjects with mental disorders tend to move more than healthy young adults, thereby increasing the difficulty of studying these subjects using fMRI.

A second source of error from the experimental subject is due to "physiological noise", which is noise that results from the subject's heart beat and respiration. This type of complex noise is thought to interfere with the MR data through various mechanisms. For example, the pulsatile motions of the brain and cerebral spinal fluid (CSF) induced from pressure changes during both the cardiac and respiratory cycle lead to volume changes within the head which cause displacement of tissue; see Dagli et al. (1999). Large organ movements due to respiration are also thought to cause fluctuations in the magnetic field, and effects of the oscillating cardiac cycle on the BOLD signal response are unknown; see, e.g., Hu et al. (1995), Stenger et al. (1999), Dagli et al. (1999).

There are many sources of noise associated with the experimental subject. Thermal noise is caused by atomic vibration that occurs at any temperature above absolute zero. Susceptibility artifacts arise from local sharp changes in magnetic susceptibility; these occur at the boundaries of tissue types and are typically greatest at air/tissue boundaries. Chemical shift artifacts arise from small changes in the Larmor frequency caused by the local chemical environment. For example, hydrogen as a component of water has a resonant frequency at 3 Tesla that is about 200 Hz higher than hydrogen as a component of fat. Typical pulse sequences include a "fat saturation pulse" to eliminate this effect.

External Noise

Interference from outside sources can also lead to distortions and artifacts in the data. Examples of interference sources include mechanical vibrations from other equipment in the building or passing vehicles, and 60 (or 50) Hertz RF noise from other nearby electrical equipment. These sources are usually considered before installing the MR machines, and precautions are normally taken. For example, an isolated foundation will reduce the effect of external sources of vibration; copper shielding will reduce the effect of nearby sources of microwave radiation, and iron shielding will reduce the effect of nearby electrical equipment (and help contain the magnetic field itself).

37.4 Modeling and Analysis

37.4.1 An Ideal Model

From a simple statistical perspective, fMRI data can be considered as a large number of individual voxel time series. If these individual times series were independent,

one could use any of the models from traditional time series analysis. For example, one could treat the brain as a linear system modulating the stimulus as its input. A model for this system would simply estimate the coefficients of the transfer function. Of course, there are other inputs such as heartbeat and respiration as well as other sources of data interference which would also need to be included. Thus, an ideal model would consist of an equation for each voxel that accounts for the true brain signal as well as all sources of systematic and random noise. By estimating all sources of noise and removing them appropriately for each voxel, an accurate estimate of brain activity could be found.

As can be imagined, an ideal model for fMRI data activation is highly impractical. A typical fMRI experiment may consist of $128 \times 128 \times 32$ voxels; therefore, a model that consists of a separate equation for each voxel would be quite cumbersome. Furthermore, the mere identification of each and every noise source in an fMRI experiment alone is a difficult task. Thereby a precise quantification of the effects of each noise source would be nearly impossible. In order to model and analyze fMRI data in a practical manner, researchers often take an approach that is similar to that of solving any giant problem; that is, by breaking it down into piecewise, practical steps. This approach allows for easier understanding of each step that is carried out (since they are performed one at a time) while also making the analysis computationally manageable. The piece-wise analysis steps involve first modeling and removing identifiable noise sources, then statistically evaluating the corrected data to estimate the amount of brain activation.

37.4.2 A Practical Approach

Preprocessing of the Data: Removal of Bias and Variance

Basically, two approaches can be employed to correct for known sources of bias and variance in fMRI data. These are correction of the data at the time of collection (proprocessing) and correction after data collection (post-processing). We use both approaches, often in tandem. For example, we use various forms of head restraint to (partially) eliminate head motion (proprocessing) and in addition we use post-processing to correct the data for head motion in addition.

We now give several examples of how data correction can be done in a post-processing manner. Some of these examples outline how many fMRI data processing steps are currently carried out using FIASCO (Functional Image Analysis Software – Computational Olio), which is a software collection that has been developed by the authors of this paper together with others and is currently used by numerous groups who analyze fMRI data. Details can be found at http://www.stat.cmu.edu/~fiasco.

These examples also demonstrate that large amounts of computation are often needed to remove bias and variance in fMRI data.

Noise from the equipment will first be addressed. Baseline noise or DC-shift noise can be corrected fairly easily, as it is a well understood source of noise

in the data. To adjust for baseline noise, the idea that the k-space data should (approximately) be oscillating around 0 at the highest spatial frequencies is taken into account. If this is not the case in the true data, the mean value at which the high frequency data is oscillating is computed, and the k-space data is shifted by a constant prior to correct for this source of noise (see Eddy et al. 1996a).

Nyquist ghosts are also well understood and can therefore be corrected fairly easily. These ghosts arise from small mistimings in the gradients with respect to the data collection. To correct for these ghosts, a phase shift is applied to each line (the same shift for each line in the x-direction) of the complex-valued k-space data in order to move the data back into its correct location. The best phase shift for correction is estimated from the data by finding a value which minimizes the magnitude of the ghosts. Typically, the center portion of the top and bottom few lines is chosen as the target (see Eddy et al. 1996a).

Magnetic field inhomogeneities may or may not vary with time. Those which do not vary with time can be corrected by "shimming" the magnet. Small magnetic objects are placed around the magnet in such a way as to reduce the inhomogeneity. Inhomogeneities associated with the subject being scanned can be corrected by dynamic shimming of the field. This is a procedure performed by the technologist at the time of the experiment. For further details see, e.g., Jezzard (1999) or Eden and Zeffiro (1997).

To reduce the effects of magnetic field distortions in a post-processing manner, the following procedure can be carried out. First a phase map can be computed from the image information in the fMRI data. This map is thought to give an estimate of regions of inhomogeneities in the data. Next, a 2-D polynomial can be fit to the field map, which is then converted into a pixel shift map. The shifted pixels are moved back to their proper locations in order to correct for the magnetic field inhomogeneities; further details may be found in Jezzard and Balaban (1995).

Noise from the experimental subject should also be corrected to improve data quality. As mentioned in the previous section, head motion is a major problem in fMRI, since even relatively small amounts of head motion can lead to false positive and false negative regions of activation. Many techniques have been considered to reduce the amount of head motion in fMRI data. These (again) can be classified into the categories of proprocessing and post-processing.

To reduce the amount of head motion that occurs at the time of the fMRI scan, external restraining devices are often used. These devices range from pillows and straps to dental bite bars and even thermoplastic face masks; see Green et al. (1994). Head restraints can greatly reduce the amount of head motion but unfortunately cannot alleviate head motion (or at least brain motion) altogether; see Friston et al. (1996). In fact, Zeffiro (1996) have suggested that some types of restraints can paradoxically increase head motion because of the discomfort they cause at pressure points. Furthermore, some types of restraints may not be appropriate for certain subject types such as small children and mentally disturbed subjects.

Also experimental design can be used to reduce the amount of apparent head motion. Often head motion is associated with presentation of a stimulus or with

physical response to the stimulus. Careful experimental design can eliminate or largely alleviate such effects.

A second way to reduce the effect of head motion at the time of the scan is through the use of navigator echos. Navigator echos are used before slice acquisition in order to detect displacements of the head. These displacements are then used to adjust the plane of excitation of the collected image accordingly. Examples include the use of navigator echos to correct for displacements in the z-direction (see Lee et al. 1996), and the use of navigator echos to correct for inter-image head rotation (see Lee et al. 1998).

A final example of a prospective method to reduce head motion is a visual feedback system that was developed by Thulborn (1999). This system provides a subject with information about their head location through a visual feedback system. By using this device the subject can tell if their head has moved during the scan and can immediately correct for it.

Proprocessing techniques have many benefits. For example, the collected fMRI data presumably has less head motion than it would have without the adaptation to reduce for head motion. Therefore, there is less need for post-processing of the data and less need to resample or interpolate the fMRI data; interpolation of fMRI data can introduce additional errors. On the other hand, these techniques often require specialized collection sequences or hardware, which can further necessitate specialized analysis software. These techniques can also lead to the need for longer data collection times.

Post-processing techniques are often more feasible for fMRI researchers because they do not require specialized collection sequences or hardware. Post-processing is mainly carried out through image registration, which involves two main steps. The first is estimation of how much motion has occurred in the collected data, and the second is correction of the estimated motion through resampling (or interpolation). This process can be carried out in two dimensions for estimation and correction of in-plane motion or three dimensions for whole head motion.

To estimate how much head motion has occurred in an fMRI data set, a reference image must first be chosen. Any image can be chosen for this purpose (typically, the first or middle image in the series), but a composite image such as the mean can also be used. Next, each image in the series is compared to the reference image using a chosen feature of the images. Image intensity is often the chosen feature, but anatomical landmarks can also be used. Mathematically, the comparison process finds the geometrical shift required between the reference image and the target image to minimize an optimization criteria. The shift that minimizes this criteria is considered to be the amount of motion that has occurred.

For two dimensional rigid motion between the current image and the reference image we consider translations in x and y, and in-plane rotations of α. In three dimensions rigid motion is even more complicated because translations can occur in x, y, and z, and rotations can occur in the α, $\boldsymbol{\theta}$, or γ directions. Criteria to be minimized between the reference image and the current image have included weighted mean square error (see Eddy et al. 1996b), variance of the image ratios (Woods et al. 1992), and mutual information (Kim et al. 1999).

Once the motion estimates are computed, the data are moved back to their appropriate locations. This is carried out by resampling or interpolation. If data relocation is not performed in the best possible way, false correlations between image voxels can be introduced. Eddy et al. (1996b) developed a Fourier interpolation method to prevent introduction of these errors. Fourier interpolation is based on the Fourier shift theorem and uses the fact that a phase shift in k-space is equal to a voxel shift in image space. By this property, translations are corrected using phase shifts, and analogously rotations are corrected by k-space rotations. Rotations in k-space are implemented by factoring the two-dimensional rotation into a product of three shearing matrices. A two-dimensional rotation matrix

$$\begin{pmatrix} \cos\alpha & -\sin\alpha \\ \sin\alpha & \cos\alpha \end{pmatrix}$$

can be written as

$$\begin{pmatrix} 1 & -\tan\frac{\alpha}{2} \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ \sin\alpha & 1 \end{pmatrix} \begin{pmatrix} 1 & -\tan\frac{\alpha}{2} \\ 0 & 1 \end{pmatrix}.$$

Once the rotations are represented by three shearing steps, these too can be corrected using phase shifts in Fourier space; for details see Eddy et al. (1996b). In a comparison of different types of fMRI data interpolation algorithms, Eddy and Young (2000) found that full Fourier interpolation was the only method which completely preserved the original data properties. Table 37.1 is taken from that paper and shows the error introduced by each method in performing the following motion. Each image was rotated by $\pi/64$ radians, translated $1/4$ pixel on the diagonal, translated back $1/4$ pixel, and rotated back $\pi/64$ radians. Many of the algorithms

Table 37.1 Mean Square Difference between original brain image and "motion-corrected" image averaged over the central 40 by 40 pixel sub-image of the original 64 by 64 pixel image. The image was rotated $\pi/64$ radians, shifted $1/4$ pixel diagonally and then translated back and rotated back to its original position; both motions were performed by the same algorithm so any differences from the original are due to the algorithm

Method	MSD
Fourier	0.00
WS16	742.86
WS8	1452.98
WS4	3136.68
NN	3830.08
Quintic	8906.20
Cubic	13864.46
WS2	28455.73
Linear	28949.22

had major errors at the edges of the image so the statistics were only computed over a central portion where no edge effects occurred.

Although three dimensional head motion seems more realistic for correction of subject movements in fMRI research, the process of estimating the six motion parameters that jointly minimize the optimization criteria can be computationally expensive. Also note that these six parameters only account for rigid body head motion. Data is actually collected one slice at a time; each slice is collected at a slightly different time. Consequently, there is no three-dimensional image of the head which can be used as a target for three dimensional registration. In spite of these considerations, the use of post-processing motion correction techniques does not necessitate specialized collection equipment, and if done correctly, can be quite accurate for small motions of the head. Large head movements, greater than say a millimeter or two, usually affect the data so much that it cannot reasonably be used.

A second significant noise source from the experimental subject is physiological noise, which is mainly attributed to the subject's heart beat and respiration. Many methods have been introduced to reduce the noise and variability caused by physiological noise. These again have included techniques that address the problem in a proprocessing manner, and techniques that reduce this noise during post-processing of the data by various modeling and subtraction techniques (Biswal et al. 1996; Chuang and Chen 2001; Glover et al. 2000; Hu et al. 1995).

Acquisition gating is most commonly used for the collection of cardiac MRI data, but it can also be used for fMRI. This proprocessing technique only collects the MRI data at certain points of the cardiac cycle so that noise effects due to cycle variations can be reduced (Guimaraes et al. 1996). Although useful for cardiac MRI, it is not as practical for fMRI research because it does not allow for continuous and rapid data collection, which is usually desirable for fMRI research.

Post-processing techniques to correct for physiological noise have included retrospective image sorting according to the phase of the cardiac cycle (Dagli et al. 1999), phase correction through the assumption that the phase is uniform for each point in k-space over many images (Wowk et al. 1997), and digital filtering (Biswal et al. 1996). Each of these techniques have certain benefits in physiological noise correction; however, each can also compromise the fMRI data. For example, image reordering results in loss of temporal resolution, and digital filtering requires the ability to acquire images rapidly as compared to the physiological noise signals.

Other approaches to physiological noise correction have included modeling of the physiological data signals in the fMRI data and subtracting out their effects. For example, Hu modeled respiration with a truncated Fourier Series and subtracted this curve from the magnitude and phase components of the k-space data (Hu et al. 1995). Glover et al. (2000) carried out a similar Fourier modeling and subtraction procedure in image space. Alternatively, Mitra and Pesaran modeled cardiac and respiratory effects as slow amplitude, frequency-modulated sinusoids and have removed these components from an fMRI principal component time series, which was obtained using a spatial frequency singular value decomposition Mitra and Pesaran (1999).

In a more recent study carried out by the authors, cardiac data and respiratory data were found to be significantly correlated with the k-space fMRI data. These correlations are a function of both time and spatial frequency (McNamee and Eddy 2003). Because of these correlations, the physiological effects can simply be reduced by first collecting the physiological data along with the fMRI data, regressing the fMRI data onto the physiological data (with the appropriate temporal considerations), and subtracting out the fitted effects (McNamee and Eddy 2004). Figure 37.6 shows the cross-correlations between the k-space phase at one spatial frequency point and the cardiac data for one experiment.

In addition to direct modeling of known sources of noise, we also attempt to reduce the effect of unknown sources of noise. For example, we routinely replace outliers with less extreme values. As another example we currently remove a linear temporal trend from each voxel times series, although we have no explanation for the source of this linear trend.

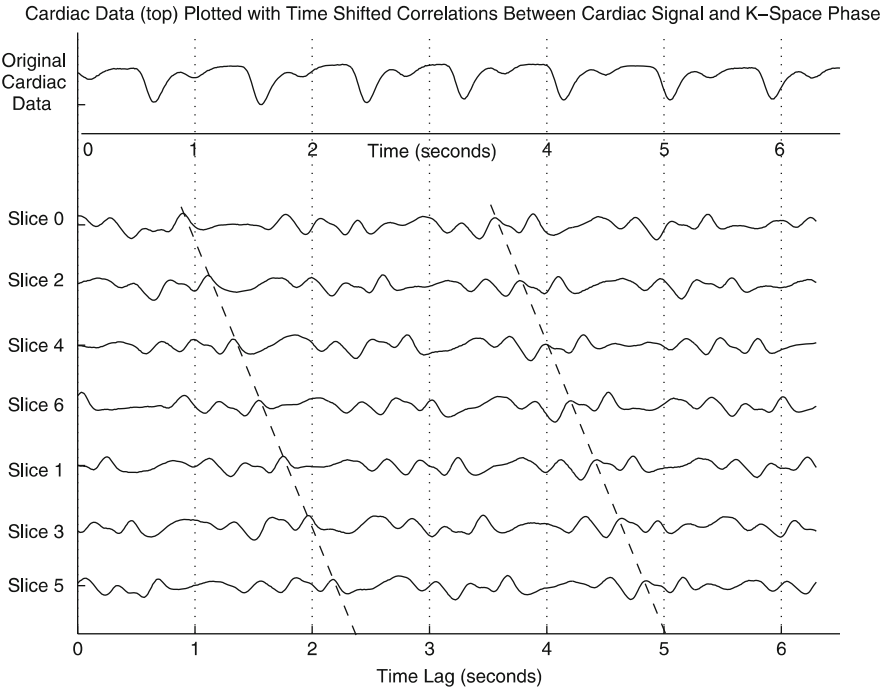


Fig. 37.6 Cardiac data shown in relation to the correlation coefficients computed between k-space phase and cardiac data as a function of time lag for a spatial frequency point in k-space. To demonstrate that the time lag is slice dependent, slices are shown in the order of collection rather than in anatomical order. Each tick-mark on the y-axis represents a unit value of 0.5. It can be clearly noted that the correlations between the phase and cardiac data are cyclic with a period equal to the TR (1.5 s in this case) of the study. Thus the *parallel diagonal lines* in the plot have a slope of $(1.5/7 = 0.214$ s/slice)

Modeling and Analysis of the Data

The very first fMRI experiments claimed to have discovered brain activation based on simply taking the difference between the average image in one experimental condition from the average in an alternate experimental condition. There was no notion of variability. It quickly became apparent that accounting for the variability was very important and fMRI analyses started using t-tests, independently on each voxel. It then became clear that multiple testing was a very serious problem; we will discuss this in more detail below. The t-test (and for experiments with more than two levels, F-test) became the standard analysis method. However, it is clear that although the stimulus switches instantaneously between experimental conditions, the brain response (presumably being continuous) must transition to the new state and there will be some time lag before the transition is complete. This has led to a variety of ad hoc "models" for the shape and lag of this transition (Gamma functions, Poissons, etc.) (see, for example, Lange and Zeger 1997).

There has been some work developing reasonable non-parametric models for the voxel time courses (Genovese 2000) and there have been a number of "time series" modeling approaches, using spectral analysis (Lange and Zeger 1997) and AR models (Harrison et al. 2003).

Because a typical brain image contains more than 100,000 voxels, it is clear that choosing a significance level of 0.05 will, even under the null hypothesis of no difference in brain activity, lead to 5,000 or more voxels being declared active. The earliest attempt to deal with this problem in fMRI was the split t-test, wherein the data were divided into two (or more) temporal subsets. T-tests were calculated separately within each subset and a voxel was declared active only if it was active in all subsets (Schneider et al. 1993). This certainly reduced the number of false positives, but obviously caused considerable loss of power.

Researchers quickly converged on the Bonferonni correction where one simply divides the significance level by the number of tests as a safe way of dealing with the problem. The loss of power is huge and some researchers started developing methods to compensate. Forman et al. (1995) proposed the contiguity threshold method, which relies on the presumption that if one voxel is active then adjacent voxels are likely to be active.

The false discovery rate (FDR) controlling procedure was then introduced as an alternate method for thresholding in the presence of multiple comparison testing. The FDR procedure allows the user to select the maximum tolerable FDR, and the procedures guarantees that *on average* the FDR will be no larger than this value. Details of the application of FDR to fMRI data can be found in Genovese et al. (2002).

As researchers in fMRI have discovered various general classes of statistical models, they have been applied to these data sets. They have had varying degrees of success and it is still the case that a linear model, often with both fixed and random effects, is the model of choice. It is reasonably well-understood and is not too far from what is actually believed. One critical problem is that as experimental designs have become more complex analyses have too; often researchers are fitting models

blindly without fully understanding some of the underlying statistical issues. The more cautious return to their linear models.

37.5 Computational Issues

As mentioned previously, a typical fMRI data set might be about 1 GB in size and would take less than 15 minutes to collect. Four or more such data sets may be collected in a single experimental session with a single subject. An entire experiment might include 20 or more subjects, and each subject might be tested twice or more. Thus there would be 160 1 GB datasets to be analyzed. (We are currently running an experiment with 300 subjects and each will be tested at least twice; one has already been tested four times.)

With such large amounts of data, storage and analysis becomes an important issue. Standard analysis of the data can take many hours, and an organized storage system is recommended. Our recent experience is that we can process the data at the rate of 2−3 MB per minute which implies that the entire experiment just described would require on the order of 1,000 h of processing time.

Several packages are available for analysis of fMRI data. A few of these will be discussed in the following section. An important point to mention before discussing these packages is that users of fMRI software should spend some time getting to know and understand the package they are using before carrying out their data processing. Questions that are important to consider, for example, may be the following. Are bias and variance in the data corrected as a routine part of data processing? If motion correction is carried out, how is this implemented? Is any additional error being introduced into the data as a result of this routine processing? What kind of modeling and comparisons of the data are being carried out, and what sort of thresholding is applied? We feel that these issues should be understood before drawing conclusions about the fMRI data, as variations in the processing and analysis may lead to variations in the results of the study.

37.5.1 Software Packages for fMRI Data Analysis

The large degree of complexity of fMRI data necessitates the use of pre-packaged software tools (unless, of course, one is an extremely ambitious programmer). Several packaged tools are described briefly below. Each software package has its benefits and limitations; we do not provide a detailed comparison here. We will focus a bit more on FIASCO, as we are part of the team that has developed and maintained this package; it is in wide use and we describe several of its unique tools.

One such software program for fMRI data processing is AFNI (Analysis of Functional NeuroImages). This software was developed by Robert Cox, formerly of the Medical College of Wisconsin, now at the National Institutes of Health; it is free

to all users. The software consists of a set of C programs for processing, analyzing, and displaying fMRI data. Currently, AFNI will run on most operating systems excluding Windows-based platforms. The program is interactive, and several of its benefits include its ability to read and switch between several different data formats, 3D viewing of data, the ability to transform data into Talairach coordinates, interactive thresholding of functional overlays onto structural images, and finally, a new feature entitled SUMA (Surface Mapping with AFNI) that adds the ability to perform cortical surface based functional imaging analysis using AFNI. The homepage for AFNI is located at http://afni.nimh.nih.gov/afni/.

Brain Voyager is a commercially available tool developed by Brain Innovation that can analyze and visualize both functional and structural MRI datasets. Brain Voyager can run on all major computer platforms including all current versions of Windows, as well as Linux/Unix and Macintosh systems. A user-interface is provided, and the program boasts several up-to-date features such as the ability to perform thresholding using the FDR technique, the ability to perform automatic brain segmentation, brain surface reconstruction, and cortex inflation and flattening, and the ability to analyze and integrate diffusion tensor imaging data with other data types. More information about this product can be found at http://www.brainvoyager.com/.

The Statistical Parametric Mapping (SPM) software package is a suite of programs, originally developed by Karl Friston, to analyze SPECT/PET and fMRI data using a voxel-based approach. The SPM program is also free but requires Matlab, a licensed Mathworks product, to run. Typical analysis steps performed by SPM include spatial normalization and smoothing of the images, parametric statistical modeling at each voxel through the use of a general linear model, and assessment of the computed statistical images. New functionality of SPM (released in SPM2) can take into account more complex issues in fMRI research such as non-sphericity, which does not restrict the user to the assumption of identically and independently distributed errors in the selected data models. For more details about this and other SPM specifics, the reader should refer to http://www.fil.ion.ucl.ac.uk/spm/.

The VoxBo package advertises itself as “the software behind the brains”. The VoxBo software is free and was developed at the University of Pennsylvania through funding from NIDA and NIMH via a Human Brain Project/Neuroinformatics grant. A Unix-based platform is currently required to run VoxBo, and a unique feature of this software includes a job scheduling system for organizational purposes. Benefits of VoxBo also include the ability carry out several data pre-processing steps such as three dimensional motion correction as well as the ability to manipulate and model the data in its time-series form. A web page is available at http://www.voxbo.org/.

A group including the current authors developed an fMRI analysis package named FIASCO. This is a collection of software tools written primarily in C, designed to analyze fMRI data using a series of processing steps. Originally, FIASCO’s main purpose was to read the raw fMRI data, process it in a series of steps to reduce sources of systematic error in the data, carry out statistical analysis, and create final brain maps showing regions of neural activation. While users still

run their fMRI data through the standard FIASCO pipeline, FIASCO has expanded into a complex set of software tools in order to accommodate many other issues and problems that have come up during the past decade as the field of fMRI has grown.

One unique feature of FIASCO is that several of the data-processing steps for the purpose of reducing bias and noise are carried out in k-space. As mentioned previously, k-space is essentially a frequency space and is the domain in which the raw data is collected. Carrying out data correction in k-space can be beneficial for many reasons. For example, certain types of noise in the data (such as Nyquist ghosts, phase drifts, and physiological noise) can be more accurately modeled and removed in k-space than in image space (McNamee and Eddy 2004). Also, images can be resampled in k-space without the need for interpolation or smoothing, both of which can introduce additional problems into the images (Eddy and Young 2000).

If users elect to the use the typical FIASCO pipeline for fMRI data analysis, tools that implement k-space correction are first carried out prior to performing the Fourier Transform. Both EPI and spiral data can be analyzed, and since both have different types of noise, unique pipelines are used for each. For example, typical FIASCO steps for processing EPI data include baseline correction (aka. correction of DC shift), removal of Nyquist ghosts, motion correction, physiological noise correction, removal of outliers and removal of unexplained data trends. All of these correction steps with the exception of the last two are carried out in k-space. The final steps are to perform statistical analysis and create brain maps showing the activated areas. Users can elect to skip any of the steps of the FIASCO process. And, they have the opportunity to add their own unique steps to the process.

As the field of fMRI grows, so does the desire to be able to perform more complex analysis procedures on the data. The FIASCO programmers have accommodated these needs with the creation of several unique general purpose tools. For example, certain tools allow the user to easily manipulate fMRI data sets by cutting, pasting, permuting, or sorting the data. More complex general purpose tools include an rpn-math feature with built-in functions; this is essentially a calculator which allows the user to perform arithmetic calculations using Reverse Polish Notation looping over all the voxels in a data set. Another tool allows users to perform matrix multiplication on fMRI data sets, and a third tool can compute eigenvalues and eigenvectors of real symmetric matrices.

In addition to general purpose tools, FIASCO also has many special purpose tools that can perform very specific tasks in relation to the fMRI data. Some of these include tools that can convert between different data formats, tools that perform specific noise reduction steps, and tools that can compute summary statistics or perform different types of hypothesis tests on the data. A complete list of FIASCO's tools can be found on the web page at http://www.stat.cmu.edu/~fiasco.

The features and tools of FIASCO have allowed its users to manipulate and experiment with fMRI data sets in unique and interesting ways. Recently, FIASCO has been applied to many other kinds of data: genetic microarrays, protein gels, video, PET, CT, etc.

37.5.2 Other Computational Issues

Aside from fMRI analysis software, other types of useful computational packages in fMRI may include software that allows the researcher to easily design and implement a desired experimental tasks for subjects to carry out while in the MRI scanner. For example, an fMRI study may focus on activation patterns during a short-term memory task. Thus an experiment engaging short-term memory would need to be carefully designed and projected into the small space of the MRI machine. The scientist would also need some sort of feedback from the subject to ensure that the task was being performed properly.

37.6 Conclusions

Functional MRI is a new and exciting method for studying the human brain as it functions in its living, natural state. As the field grows and the methodology improves, so do the many computational and statistical problems that accompany the storage, processing, analysis, and interpretation of the data. In this chapter we have briefly summarized some of the complexities and limitations of the method as well as describing some of the approaches available for addressing these complexities and limitations. The field of fMRI will, no doubt, continue to broaden and expand. As it does so, the continued integration of scientists and researcher from many disciplines will be necessary for fMRI to reach its full potential and to help to uncover one of the greatest mysteries of humankind.

References

Belliveau, J.W., Kennedy, D.N., McKinstry, R.C., Buchbinder, B.R., Weisskoff, R.M., Cohen, M.S., Vevea, J.M., Brady, T.J., Rosen, B.R.: Functional mapping of the human visual cortex by magnetic resonance imaging. Science **254**, 716–719 (1991)

Biswal, B., DeYoe, E.A., Hyde, J.S.: Reduction of physiological fluctuations in fMRI using digital filters. Magn. Reson. Med. **35**, 107–113 (1996)

Buxton, R.B.: *Introduction to Functional Magnetic Resonance Imaging: Principles and Techniques*, Cambridge University Press, Cambridge (2002)

Cherry, S.R., Phelps, M.E.: Imaging brain function with positron emission tomography. In: Toga, A.W., Mazziotta, J.C. (eds.) Brain Mapping: The Methods. Academic Press, New York (1996)

Chuang, K.H., Chen, J.H.: IMPACT: Image-based physiological artifacts estimation and correction technique for functional MRI. Magn. Reson. Med. **46**, 344–353 (2001)

Dagli, M.S., Ingeholm, J.E., Haxby, J.V.: Localization of cardiac-induced signal change in fMRI. NeuroImage **9**, 407–415 (1999)

Eddy, W.F., Fitzgerald, M., Genovese, C.R., Mockus, A., Noll, D.C.: Functional imaging analysis software – computational olio. In: Prat, A. (eds.) Proceedings in Computational Statistics, Physica, Heidelberg (1996a)

Eddy, W.F., Fitzgerald, M., Noll, D.C.: Improved image registration by using Fourier interpolation. Magn. Reson. Med. **36**, 923–931 (1996b)

Eddy, W.F., Fitzgerald, M., Genovese, C., Lazar, N., Mockus, A., Welling, J.: The challenge of functional magnetic resonance imaging. J. Comp. Graph. Stat. **8**(3), 545–558 (1999); Adapted from: The challenge of functional magnetic resonance imaging. In Massive Data Sets, Proceedings of a Workshop, pp. 39–45. National Academy Press, Washington, DC (1996)

Eddy, W.F., Young, T.K.: Optimizing the resampling of registered images. In: Bankman, I.N. (eds.) Handbook of Medical Imaging: Processing and Analysis. Academic Press, New York (2000)

Eden, G.F., Zeffiro, T.A.: PET and fMRI in the detection of task-related brain activity: Implications for the study of brain development. In: Thacher, R.W., Lyon, G.R., Rumsey, J., Krasnegor, N.K. (eds.) Developmental NeuroImaging: Mapping the Development of Brain and Behavior, Academic Press, San Diego (1997)

Finger, S.: Origins of Neuroscience: A History of Explorations Into Brain Function, Oxford University Press, Oxford (1994)

Forman, S.D., Cohen, J.D., Fitzgerald, M., Eddy, W.F., Mintun, M.A., Noll, D.C.: Improved assessment of significant change in functional magnetic resonance imaging (fMRI): Use of a cluster size threshold. Magn. Reson. Med. **33**, 636–647 (1995)

Friston, K.J., Williams, S., Howard, R., Frackowiak, R.S.J., Turner, R.: Movement-related effects in fMRI time-series. Magn. Reson. Med. **35**, 346–355 (1996)

Gaillard, W.D., Grandin, C.B., Xu, B.: Developmental aspects of pediatric fMRI: Considerations for image acquisition, analysis, and interpretation. NeuroImage **13**, 239–249 (2001)

Genovese, C.R.: A Bayesian time-course model for functional magnetic resonance imaging (with discussion). J. Am. Stat. Assoc. **95**, 691–703 (2000)

Genovese, C.R., Lazar, N.A., Nichols, T.E.: Thresholding of statistical maps in neuroimaging using the false discovery rate. NeuroImage **15**, 870–878 (2002)

Glad, I., Sebastiani, G.: A Bayesian approach to synthetic magnetic resonance imaging. Biometrika **82**, 237–250 (1995)

Glover, G.H., Li, T.Q., Ress, D.: Image-based method for retrospective correction of physiological motion effects in fMRI: RETROICOR. Magn. Reson. Med. **44**, 162–167 (2000)

Green, M.V., Seidel, J., Stein, S.D., Tedder, T.E., Kempner, K.M., Kertzman, C., Zeffiro, T.A.: Head movement in normal subjects during simulated PET brain imaging with and without head restraint. J. Nucl. Med. **35**(9), 1538–1546 (1994)

Guimaraes, A.R., Melcher, J.R., Talavage, T.M., Baker, A.J., Rosen, B.R., Weisskoff, R.M.: Detection of inferior colliculus activity during auditory stimulation using cardiac-gated functional MRI with T_1 correction. In: NeuroImage, 2nd International Conference on Functional Mapping of the Human Brain (1996)

Hajnal, J.V., Myers,R., Oatridge, A., Schwieso, J.E., Young, I.R., Bydder, G.M.: Artifacts due to stimulus correlated motion in functional imaging of the brain. Magn. Reson. Med. **31**, 283–291 (1994)

Harrison, L., Penny, W.D., Friston, K.: Multivariate Autoregressive Modeling of fMRI time series. NeuroImage **19(4)**, 1477–1491 (2003)

Hu, X., Le, T.H., Parrish, R., Erhard, P.: Retrospective estimation and correction of physiological fluctuation in functional MRI. Magn. Reson. Med. **34**, 201–212 (1995)

Jezzard, P.: Sources of distortion in functional MRI data. Hum. Brain Map. **8**, 80–85 (1999)

Jezzard, P., Balaban, R.S.: Correction for geometric distortions in echo planar images from B_0 field variations. Magn. Reson. Med. **34**, 65–73 (1995)

Kim, B., Boes, J.L., Bland, P.H., Chenevert, T.L., Meyer, C.R.: Motion correction in fMRI via registration of individual slices into an anatomical volume. Magn. Reson. Med. **41**, 964–972 (1999)

Kwong, K.K., Belliveau, J.W., Chesler, D.A., Goldberg, I.E., Weisskoff, R.M., Poncelet, B.P., Kennedy, D.N., Hoppel, B.E., Cohen, M.S., Turner, R., Cheng, H., Brady, T.J., Rosen, B.R.: Dynamic magnetic resonance imaging of human brain activity during primary sensory stimulation. Proc. Natl. Acad. Sci. USA **89**, 5675 (1992)

Lange, N.: Statistical approaches to human brain mapping by functional magnetic resonance imaging. Stat. Med. **15**, 389–428 (1996)

Lange, N., Zeger, S.L.: Non-linear Fourier time series analysis for human brain mapping by functional magnetic resonance imaging (with discussion). Appl. Stat. **46**, 1–29 (1997)
Lazar, N.A., Genovese, C.R., Eddy, W.F., Welling, J.: Statistical issues in fMRI for brain imaging. Int. Stat. Rev. **69**, 105–127 (2001)
Lee, C.C., Jack, C.R., Grimm, R.C., Rossman, P.J., Felmlee, J.P., Ehman, R.L., Riederer, S.J.: Real-time adaptive motion correction in functional MRI. Magn. Reson. Med. **36**, 436–444 (1996)
Lee, C.C., Grimm, R.C., Manduca, A., Felmlee, J.P., Ehman, R.L., Riederer, S.J., Jack, C.R.: A prospective approach to correct for inter-image head rotation in FMRI. Magn. Reson. Med. **39**, 234–243 (1998)
Logothetis, N.K.: The neural basis of the blood-oxygen level dependent functional magnetic resonance imaging signal. Philos. Trans. R. Soc. Lond. B. Biol. Sci. **357**(1424), 1003–1037 (2002)
McAllister, T.W., Saykin, A.J., Flashman, L.A., Sparling, M.B., Johnson, S.C., Guerin, S.J., Mamourian, A.C., Weaver, J.B., Yanofsky, N.: Brain activation during working memory 1 month after mild traumatic brain injury: A functional MRI study. Neurol. **53**, 1300–1308 (1999)
McNamee, R.L., Eddy, W.F.: Correlations between cardiac data and fMRI data as a function of spatial frequency and time. In: Proceedings of the 2003 25th Annual International Conference of the IEEE-EMBS Society. Cancun, Mexico (2003)
McNamee, R.L., Eddy, W.F.: Examination and removal of the spatial and time-related effects of physiological noise in fMRI data. (in preparation) (2004)
Mitra, P.P., Pesaran, B.: Analysis of dynamic brain imaging data. Biophys. J. **78**, 691–708 (1999)
Ogawa, S., Lee T.-M., Nayak, A.S., Glynn, P.: Oxygenation-sensitive contrast in magnetic resonance image of rodent brain at high magnetic fields. Magn. Reson. Med. **14**, 680–78 (1990)
Ogawa, S., Tank, D.W., Menon, D.W., Ellermann, J.M., Kim, S., Merkle, H., Ugurbil, K.: Intrinsic signal changes accompanying sensory stimulation: Functional brain mapping using MRI. Proc. Natl. Acad. Sci. USA **89**, 5951–5955 (1992)
Poldrack, R.A.: Imaging brain plasticity: Conceptual and methodological issues – A theoretical review. NeuroImage **12**, 1–13 (2000)
Rosen, A.C., Prull, M.W., O'Hara, R., Race, E.A., Desmond, J.E., Glover, G.H., Yesavage, J.A., Gabrieli, J.D.: Variable effects of aging on frontal lobe contributions to memory. NeuroReport **3**(8), 2425–2428 (2002)
Schneider, W., Noll, D.C., Cohen, J.D.: Functional topographic mapping of the cortical ribbon in human vision with conventional MRI scanners. Nature **365**, 150–153 (1993)
Stenger, V.A., Peltier, S., Boada, F.E., Noll, D.C.: 3D spiral cardiac/respiratory ordered fMRI data acquisition at 3 Tesla. Magn. Reson. Med. **41**, 983–991 (1999)
Thulborn, K.R., Waterton, J.C., Matthews, P.M., Radda, G.K.: Oxygenation dependence of the transverse relaxation time of water protons in whole blood at high field. Biochem. Biophys. Acta. **714**, 265–270 (1982)
Thulborn, K.R.: Visual feedback to stabilize head position for fMRI. Magn. Reson. Med. **41**, 1039–1043 (1999)
Turner, R.: Functional mapping of the human brain with magnetic resonance imaging. Sem. Neurosci. **7**, 179–194 (1995)
Villringer, A.: Physiological Changes During Brain Activation. In: Moonen, C.T.W., Bandettini, P.A. (eds.) Functional MRI, Springer, Berlin (2000)
Weisskoff, R.M.: Simple measurement of scanner stability for functional NMR imaging of activation in the brain. Magn. Reson. Med. **36**, 643–645 (1996)
Weisskoff, R.M., Kiihne, S.: MRI susceptometry: Image-based measurement of absolute susceptibility of MR contrast agents and human blood. Magn. Reson. Med. **24**, 375–83 (1992)
Woods, R., Cherry, S., Mazziotta, J.: Rapid automated algorithm for aligning and reslicing PET images. J. Comp. Ass. Tomog. **16**, 620–633 (1992)
Wowk, B., McIntyre, M.C., Saunders, J.K.: k-space detection and correction of physiological artifacts in fMRI. Magn. Reson. Med. **38**, 1029–1034 (1997)
Zeffiro, T.: Clinical functional image analysis: Artifact detection and reduction. NeuroImage **4**, S95–S100 (1996)

Chapter 38
Network Intrusion Detection

David J. Marchette

38.1 Introduction

Attacks against computers and the Internet are in the news every week. These primarily take the form of malicious code such as viruses and worms, or denial of service attacks. Less commonly reported are attacks which gain access to computers, either for the purpose of producing damage (such as defacing web sites or deleting data) or for the opportunities such access provides to the attacker, such as access to bank accounts or control systems of power stations. In a perspectives article in Science (Wulf and Jones 2009) the authors argue that computer systems are getting less secure, not more, and that traditional models of security based on perimeter defenses are not working. They argue for a "defense in depth" approach, but not one based on more layers of perimeter defense, but rather on different types of defense; different security protocols; different types of detection methodologies; security protocols and defenses designed for the specific applications. Our view is that one of the tools needed is statistical analysis, in particular for detecting suspicious activity. This chapter will discuss some of the areas in which computational statistics can be applied to these and related problems. We focus exclusively on the detection of attacks, rather than defense (such as firewalls and other security protocols).

Several books are available that describe the basic ideas in intrusion detection. These include Amoroso (1999), anonymous (1997), Bace (2000), Escamilla (1998), Marchette (2001), Northcutt et al. (2001) and Proctor (2001). Intrusion detection is typically split into two separate problems. Network intrusion detection typically looks at traffic on the network, while host based intrusion detection involves collecting data on a single host. Both involve very large and complex data sets, and

D.J. Marchette (✉)
Naval Surface Warfare Center, Dahlgren, VA, USA
e-mail: david.marchette@navy.mil

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3_38,

both have aspects that lend themselves to statistical solutions. We will only touch on a few such; the reader is encouraged to investigate the references.

There are two basic approaches to network intrusion detection. Most existing systems rely on signatures of attacks. This approach relies on some set of features that can be extracted from the data that indicate the existence of an attack. This is analogous to the virus scanners, which look for a sequence of bytes that are indicative of a virus. In the network realm, this could be attempts to access services that are denied, malformed packets, too many failed attempts to log in, et cetera. The second approach is anomaly detection. The "normal" activity of the network is modeled, and outliers are indicative of attacks. The definition of "normal" is dependent on the type of attacks that one is interested in, and requires statistical models.

This chapter will first describe the basics of the TCP/IP protocol, sufficient to understand the data and the examples given. Then we will look at detecting denial of service attacks, and estimating the number of attacks on the Internet. Network data is streaming data, and we will discuss this and some areas in which computational statistics can play a part. This will lead to a discussion of simple visualization techniques applied to network data, with some discussion of the types of insights that can be gained from this. We will then take a detour from network data and consider profiling. This will illustrate a type of anomaly detection, which will then be discussed within a network context. Finally we discuss some statistical techniques for anomaly detection in network security.

38.2 Basic TCP/IP

When you visit a web site, your request and the response data are sent as a series of packets, each consisting of a header containing addressing and sequencing information, and a payload or data section in which the information resides. Packets are typically relatively small (less than 1500 bytes). In order to analyze the traffic and detect attacks, one needs to collect the packets, and may need to process either the header or the payload. We will (somewhat arbitrarily) denote an attack that can be detected by investigating the header only a "network attack" while leaving those that require investigation of the payload in the "host attack" realm.

One reason for this distinction is encryption. If the data are encrypted (for example, data from a secure web site), the header remains in the clear, and so this information is still available for analysis by the statistician. The payload is inaccessible (assuming a sufficiently strong encryption scheme) and so cannot be used to detect attacks until it is decrypted at the destination host. For this reason (and others), we consider any attack that requires investigation of the data in a packet to be better detected at the host than on the network.

There are several protocols used on the Internet to ensure a level of performance or reliability in the communication. We will briefly discuss TCP (the Transmission Control Protocol), since it is one of the most important ones, and will allow us to

discuss a class of denial of service attacks. For more information about the various protocols, see Stevens (1994).

First, however, it is necessary that we discuss the Internet Protocol (IP). This protocol is not reliable, in the sense that there is no mechanism in place to ensure that packets are received. The IP header contains the source and destination IP addresses, which are 32-bit integers identifying the sending and receiving computer for the packet. There are other fields in the packet that are used to control the routing of the packet, et cetera, but we will not dwell on these here. As always, interested readers should investigate Stevens (1994) or any of the many books on the TCP/IP protocol suite.

Since IP is unreliable, a packet sent may or may not reach its destination, and if it does not, there is no guarantee that anyone will notice. Thus, a more reliable protocol is required. TCP implements a reliable two way communication channel, and is used for web, email, and many other user applications. The TCP header is shown in Fig. 38.1. The important fields, for this discussion, are the ports, sequence numbers and flags.

The ports are a method for identifying a specific session, and can be thought of as a 16-bit addition to the IP address that uniquely determines the session. Ports are also used to identify the application requested. For example, port 80 is the standard web port, and web browsers know that in order to obtain a web page from a server they need to make a connection on this port.

To initiate and maintain a connection, the flags and sequence numbers are used. The TCP protocol requires a three-way handshake to initiate a connection. First the client sends a SYN packet (in this manner we will denote a packet with only the SYN flag set; similarly with other flag combinations) to the server. The server responds with a SYN/ACK packet, acknowledging the connection. The client then finalizes the connection with an ACK packet. Sequence numbers are also passed, and tracked to ensure that all sent packets are received and acknowledged, and to allow the reconstruction of the session in the correct order. Packets that are not acknowledged are resent, to ensure that they are ultimately received and processed.

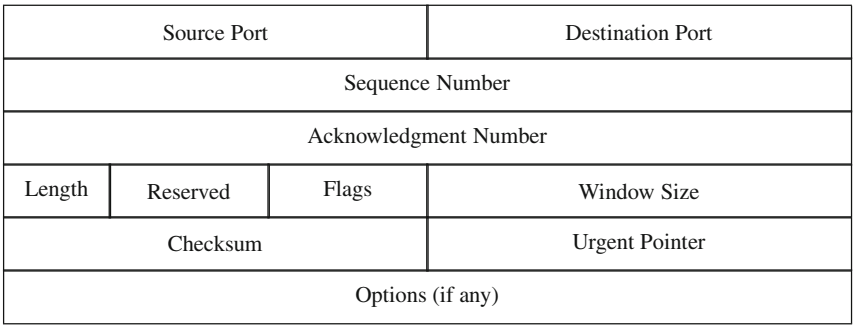


Fig. 38.1 The TCP header. The header is to be read left to right, top to bottom. A row corresponds to 32 bits

Once a session has been instantiated through the three-way handshake, packets are acknowledged with packets in which the ACK flag is set. In this manner the protocol can determine which packets have been received and which need to be resent. If a packet has not been acknowledged within a given time, the packet is resent, and this can happen several times before the system determines that something has gone wrong and the session is dropped (usually by sending a reset (RST) packet). Note that this means that if there is no response to the SYN/ACK packet acknowledging the initiation of the session there will be a period (of several seconds) in which the session is kept open by the destination host as it tries resending the SYN/ACK hoping for a response. This is the basis of some denial of service attacks, which we will discuss in the next section.

38.3 Passive Sensing of Denial of Service Attacks

The TCP protocol provides a simple (and popular) method for denial of service attacks. The server has a finite number of connections that it can handle at a time, and will refuse connections when its table is full. Thus, if an attacker can fill the table with bogus connections, legitimate users will be locked out.

This attack relies on two fundamental flaws in the protocols. The first is that the source IP address is never checked, and thus can be "spoofed" by putting an arbitrary 32 bit number in its place. Second, the three-way handshake requires the third (acknowledgment) packet, and the server will wait several seconds before timing out a connection. With each requested connection, the server allocates a space in its table and waits for the final acknowledgment (or for the connection to time out). The attacker can easily fill the table and keep it filled by sending spoofed SYN packets to the server.

Thus, the attacker sends many SYN packets to the server, spoofed to appear to come from a large number of different hosts. The server responds with SYN/ACK packets to these hosts, and puts the connection in its table to await the final ACK, or a time-out (usually several seconds). Since the ACK packets are not forthcoming, the table quickly fills up, and stays full for as long as the attacker continues to send packets.

There are clever ways to mitigate this problem, which can keep the table from filling up. One, the "SYN-cookie" involves encoding the sequence number of the SYN/ACK in a way that allows the server to recognize legitimate ACK packets without needing to save a spot in the table for the connection. However, even these can be defeated through a sufficiently high volume attack.

These unsolicited SYN/ACK packets can be observed by any network sensor, and thus provide a method for estimating the number and severity of such attacks throughout the Internet. These unsolicited packets are referred to as *backscatter*. They may take other forms than SYN/ACK packets, depending on the type of packet sent in the attack. See Moore et al. (2001), Marchette (2002), Marchette for more information.

Typically, the attacker first compromises a large number of computers, using special distributed attack software, and it is these computers that launch the attack. This makes it very difficult to block the attack, and essentially impossible to track down the attacker, at least through information available to the victim.

Backscatter packets provide several opportunities for statistical analysis. They allow the estimation of the number of attacks on the Internet in real time. One may be able to estimate the severity of the attacks and number of attackers. Finally, it may be possible to characterize different types of attacks or different attack tools and identify them from the pattern of the packets. Some initial work describing some of these ideas is found in Giles et al. (2003).

A network sensor is a computer that captures packets (usually just the packet headers) as they traverse the network. These are usually placed either just before or just after a firewall to collect all the packets coming into a network. Through such a system, one can observe all the unsolicited SYN/ACK packets addressed to one of the IP addresses owned by the network.

Note that this means that only a fraction of the backscatter packets resulting from the attack are seen by any sensor. If we assume that the sensor is monitoring a class B network (an address space of $65,536$ IP addresses), then we observe a random sample of $1/65,536$ of the packets, assuming the attack selects randomly from all 2^{32} possible IP addresses. This points to several areas of interest to statisticians: we observe a subset of the packets sent to a subset of the victims, and wish to estimate the number of victims, the number of packets sent to any given victim, and the number of attackers for any given victim.

38.4 Streaming Data

Network packets are streaming data. Standard statistical and data mining methods deal with a fixed data set. There is a concept of the size of the data set (usually denoted n) and algorithms are chosen based in part on their performance as a function of n. In streaming data there is no n: the data are continually captured and must be processed as they arrive. While one may collect a set of data to use to develop algorithms, the nonstationarity of the data requires methods that can handle the streaming data directly, and update their models on the fly.

Consider the problem of estimating the average amount of data transferred in a session for a web server. This is not stationary: there are diurnal effects; there may be seasonal effects (for example at a university); there may be changes in the content at the server. We'd like a number calculated on a window of time that allows us to track (and account for) the normal trends and detect changes from this normal activity.

This requires some type of windowing or recursive technique. The recursive version of the sample mean is well known:

$$\bar{X}_n = \frac{n-1}{n}\bar{X}_{n-1} + \frac{1}{n}X_n.$$

Replacing n on the right hand side with a fixed constant N implements an exponential window on the mean. This was exploited in the NIDES intrusion detection system (Anderson et al. 1995). Similar techniques can be used to compute other moments. An alternative formulation is:

$$\hat{X}_n = (1 - \theta)X_{n+1} + \theta\hat{x}_{n-1},$$

for $0 < \theta < 1$. θ may be fixed or may itself change based on some statistic of the data.

In fact, the kernel density estimator has a simple recursive version, that allows the recursive estimate of the kernel density estimator at a fixed grid of points. Yamato (1971), Wegman and Davies (1979) give two versions of this:

$$\hat{f}_n(x) = \frac{n-1}{n}\hat{f}_{n-1}(x) + \frac{1}{nh_n}K\left(\frac{x - X_n}{h_n}\right)$$

$$\check{f}_n(x) = \frac{n-1}{n}\left(\frac{h_{n-1}}{h_n}\right)^{\frac{1}{2}}\check{f}_{n-1}(x) + \frac{1}{nh_n}K\left(\frac{x - X_n}{h_n}\right).$$

In either case, fixing n at a constant and h_n either at a constant or a recursively estimated value implements an exponentially windowed version of the kernel estimator. (Similarly, one can phrase this in terms of θ as was done with the mean; see Wegman and Marchette 2004). These can in turn be used to estimate the "normal" activity of various measurements on the network, and provide a mechanism for detecting changes from normal, which in turn may indicate attacks. More information on these issues can be found in Wegman and Marchette (2004).

Similar approaches can be implemented for other density estimation techniques. In particular, the adaptive mixtures approach of Priebe (1994) has a simple recursive formulation that can be adapted to streaming data.

There are several applications of density estimation to intrusion detection that one might consider. It is obvious that unusually large downloads (or uploads) may be suspicious in some environments. While it is not clear that density estimation is needed for this application, there might be some value in detecting changes in upload/download behavior. This can be detected through the tracking of the number of bytes transferred per session.

Perhaps a more compelling application is the detection of trojan programs. A trojan is a program that appears to be a legitimate program (such as a telnet server) but acts maliciously, for example to allow access to the computer by unauthorized users. Obviously the detection of trojans is an important aspect of computer security.

Most applications (web, email, ftp, et cetera) have assigned ports on which they operate. Other applications may choose to use fixed ports, or may choose any available port. Detecting new activity on a given port is a simple way to detect a trojan program. More sophisticated trojans will replace a legitimate application, such as a web server. It is thus desirable to determine when a legitimate application is acting in a manner that is unusual.

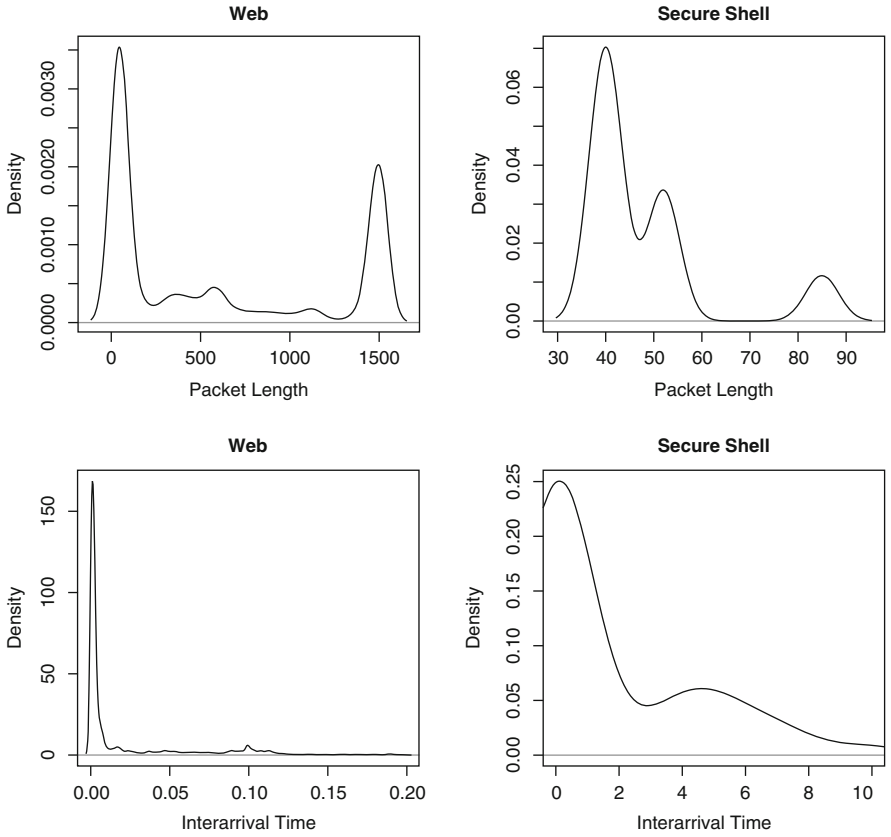


Fig. 38.2 Packet length in bytes (*top*) and packet inter arrival times in seconds (*bottom*) for web (*left*) and secure shell (*right*) sessions. Kernel estimators were used to estimate the densities. The inter arrival times were truncated to show the bulk of the data

Consider Fig. 38.2. We have collected data for two applications (web and secure shell) over a period of 1 h, and estimated the densities of the packet length and inter arrival times. As can be seen, the two applications have very different patterns for these two measures. This is because they have different purposes: secure shell is a terminal service which essentially sends a packet for every character typed (there is also a data transfer mode to secure shell, but this mode was not present in these data); web has a data transfer component with a terminal-like user interaction.

By monitoring these and other parameters, it is possible to distinguish between many of the common applications. This can then be used to detect when an application is acting in an unusual manner, such as when a web server is being used to provide telnet services. See Early and Brodley (2003) for a more extensive discussion of this.

Note that web traffic has two main peaks at either end of the extremes in packet size. These are the requests, which are typically small, and the responses, which

are pages or images and are broken up into the largest packets possible. The mass between the peaks mostly represent the last packets of transfers which are not a multiple of the maximum packet size, and small transfers that fit within a single packet.

The inter packet arrival times for secure shell also have two peaks. The short times correspond to responses (such as the response to a directory list command) and to characters typed quickly. The later bump probably corresponds to the pauses between commands, as the user processes the response. These arrival times are very heavy tailed because of the nature of secure shell. Sessions can be left open indefinitely, and if no activity occurs for a sufficiently long time, "keep alive" packets are sent to ensure that the session is still valid.

In Early and Brodley (2003) it is shown, in fact, that differences in the counts for the TCP flags can be used to differentiate applications. These, combined with mean inter packet arrival times and packet lengths (all computed on a window of n packets for various values of n), do a very creditable job of distinguishing applications. This is clearly an area in which recursive methods like those mentioned above would be of value. It also is reasonable to hypothesize that estimating densities, rather then only computing the mean, would improve the performance.

By detecting changes in the densities of applications it may be possible to detect when they have been compromised (or replaced) by a trojan program. It may also be possible to detect programs that are not performing as advertised (web servers acting like telnet servers, for example).

38.5 Visualization

Visualization of complex data is important but difficult. This is especially true of streaming data. While many complex techniques for visualization have been developed, simple scatter plots can be used effectively, and should not be shunned.

Figure 38.3 shows a scatter plot of source port against time for an 8 h period of time. These are all the SYN packets coming in to a class B network (an address space of 65,536 possible IP addresses). This graphic, while simple, provides quite a few interesting insights.

Note that there are a number of curves in the plot. These are a result of the fact that each time a client initiates a session with a server, it chooses a new source port, and this corresponds to the previous source port used by the client incremented by one. Contiguous curves correspond to connections by a single source IP. Vertical gaps in the curves indicate that the IP visited other servers between visits to the network. It is also easy to see the start of the work day in this plot, indicated by the heavy over plotting on the right hand side.

The source ports range from 1,024 to 65,536. Different applications and operating systems select ports from different ranges, so one can learn quite a bit from investigating plots like this.

The plot of Fig. 38.3 is static. Figure 38.4 is meant to illustrate a dynamic plot. This is analogous to the waterfall plots used in signal processing. It displays a

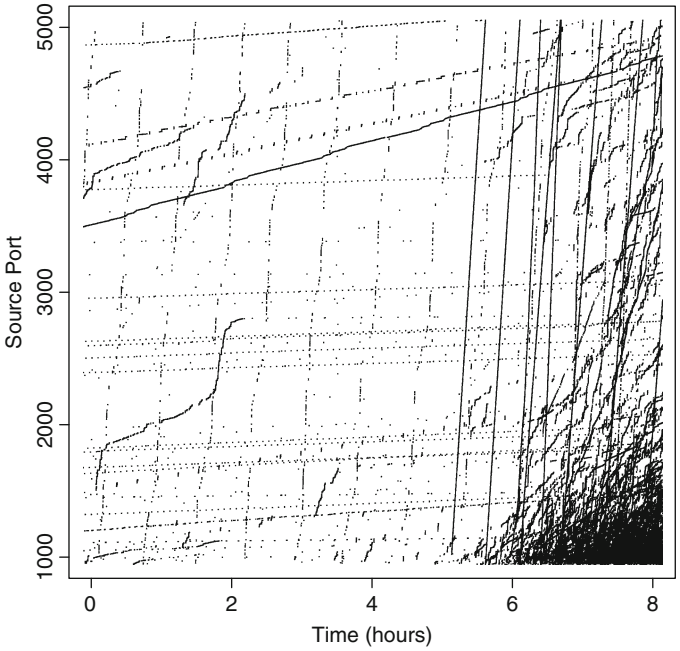


Fig. 38.3 Source port versus time for all the incoming SYN packets for an 8 h period

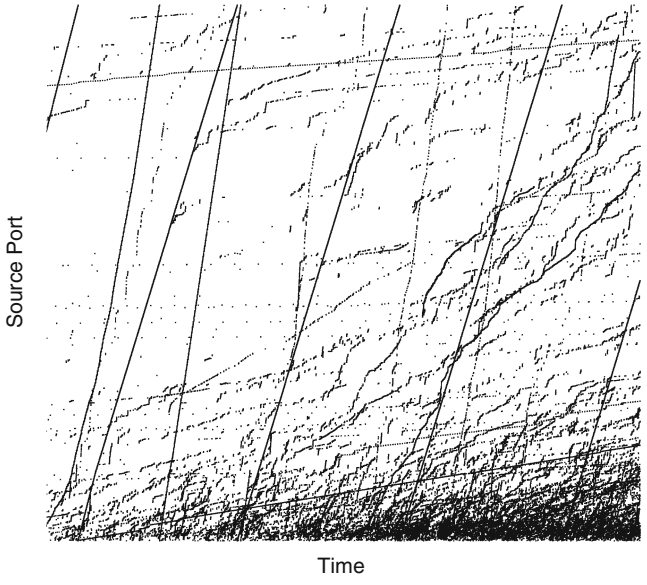


Fig. 38.4 Source port versus time for a short time period, the last two hours from Fig. 38.3. As time progresses, the plot shifts from right to left, dropping the left most column and adding a new column on the right

snapshot in time that is continuously updated. As new observations are obtained they are plotted on the right, with the rest of the data shifting left, dropping the left most column. Plots like this are required for streaming data.

Simple plots can also be used to investigate various types of attacks. In Fig. 38.5 is plotted spoofed IP address against time for a denial of service attack against a single server. Each point corresponds to a single unsolicited SYN/ACK packet received at the sensor from a single source. This plot provides evidence that there where actually two distinct attacks against this server. The left side of the plot shows a distinctive stripped pattern, indicating that the spoofed IP addresses have been selected in a systematic manner. On the right, the pattern appears to be gone, and we observe what looks like a random pattern, giving evidence that the spoofed addresses are selected at random (a common practice for distributed denial of service tools). Between about 0.03 and 0.06 there is evidence of overlap of the attacks, indicating that this server was under attack from at least two distinct programs simultaneously.

Another use of scatter plots for analysis of network data is depicted in Fig. 38.6. These data were collected on completed sessions. The number of packets is plotted against the number of bytes. Clearly there should be a (linear) relationship between

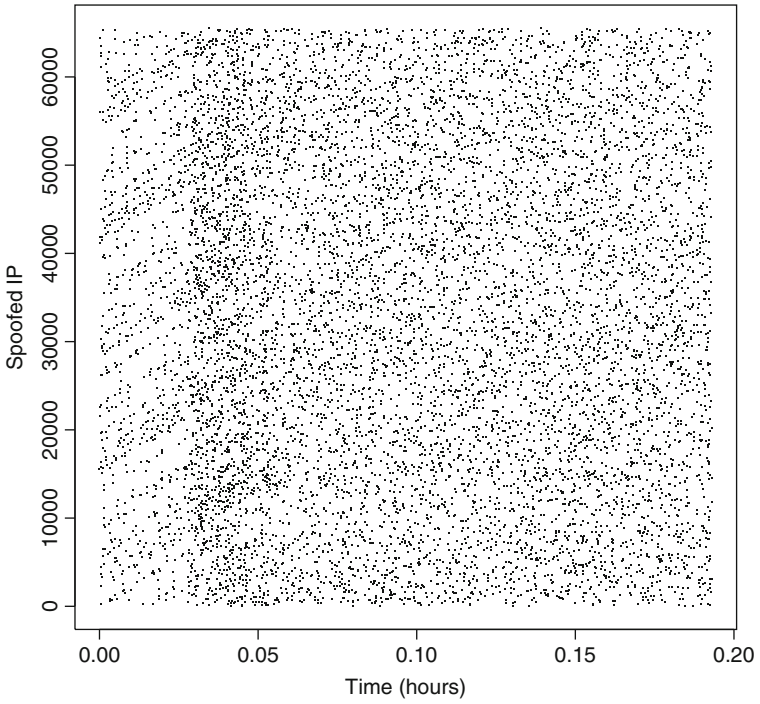


Fig. 38.5 Plot of spoofed IP address against time for backscatter packets from a denial of service attack against a single server. The IP addresses have been converted to 16-bit numbers, since in this case they correspond to the final two octets of the IP address

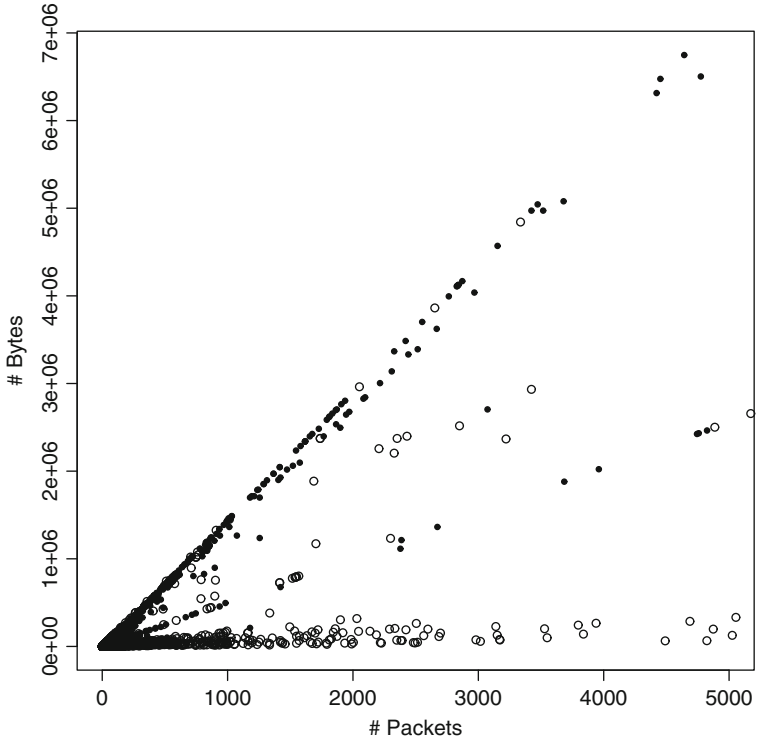


Fig. 38.6 Number of bytes transferred within a completed session plotted against the number of packets within the session. Solid dots correspond to email sessions, circles correspond to all other applications

these. The interesting observation is that there are several linear relationships. This is similar to the observations made about Fig. 38.2, in which it was noted that different applications use different packet lengths.

Figure 38.7 shows the number of bytes transferred within a session plotted against the start time of the session. There is a lot of horizontal banding in this plot, corresponding mostly to email traffic. It is unknown whether the distinctive repetitive patterns are a result of spam (many email messages all the same size) or whether there are other explanations for this. Since these data are constructed from packet headers only, we do not have access to the payload and cannot check this hypothesis for these data. Figure 38.8 shows a zoom of the data. The band just below 400 bytes correspond to telnet sessions. These are most likely failed login attempts. This is the kind of thing that one would like to detect. The ability to drill down the plots, zooming and selecting observations to examine the original data, is critical to intrusion detection.

High dimensional visualization techniques are clearly needed. Parallel coordinates is one solution to this. In Fig. 38.9 we see session statistics for four different applications plotted using parallel coordinates.

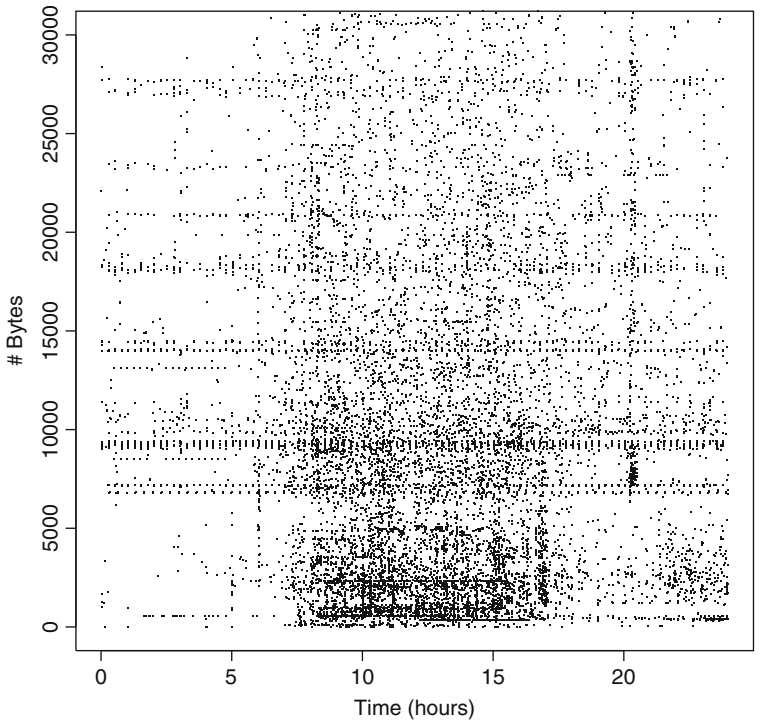


Fig. 38.7 Number of bytes transferred for each session plotted against the starting time of the session for a single day

One problem with plots like this is that of over plotting. Wegman solves this via the use of color saturation (see Wegman and Dorfman 2001; Wilhelm et al. 1999). Without techniques such as this it is extremely difficult to display large amounts of data. Figure 38.9 illustrates this problem in two ways. First, consider the secure shell data in the upper left corner. It would be reasonable to conclude from this plot that secure shell sessions are of short duration, as compared with other sessions. This is an artifact of the data. For these data there are only 10 secure shell sessions, and they all happen to be of short duration. Thus, we really need to look at a lot of data to see the true distribution for this applications. Next, look at the email plot in the upper right. Most of the plot is black, showing extensive over plotting. Beyond the observation that these email sessions have heavy tails in the size and duration of the sessions, little can be gleaned from this plot.

A further point should be made about the web sessions. Some of the sessions which are relatively small in terms of number of packets and bytes transferred have relatively long durations. This is a result of the fact that often web sessions will not be closed off at the end of a transfer. They are only closed when the browser goes to another web server, or a time-out occurs. This is an interesting fact about the web application which is easy to see in these plots.

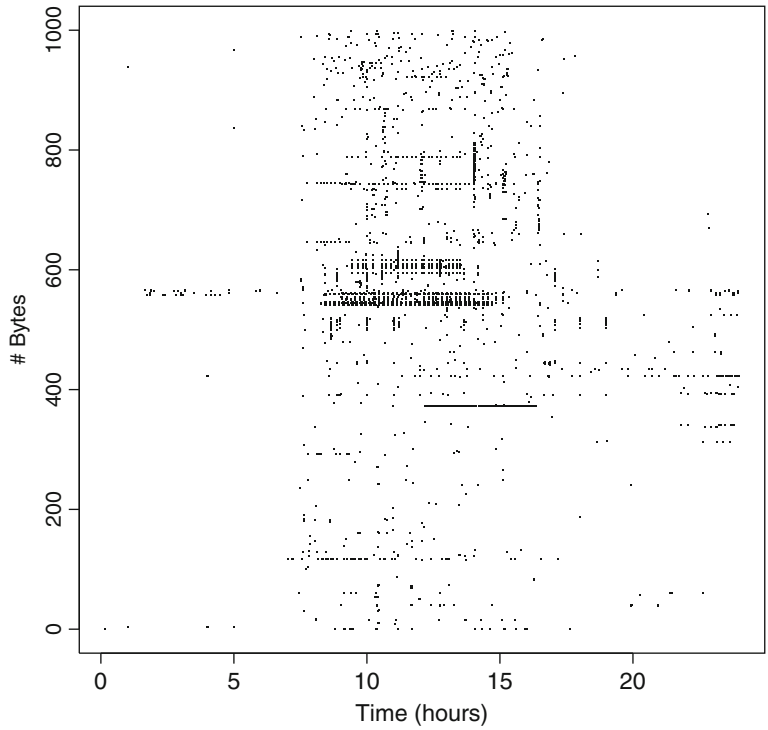


Fig. 38.8 The portion of the sessions in Fig. 38.7 which were less than 1,000 bytes

38.6 Profiling and Anomaly Detection

We will now briefly consider host based intrusion detection. While the data considered is not network data, the statistical techniques used are applicable to network problems, as will be discussed.

One of the important problems of computer security is user authentication. This is usually performed by requiring the user to type a password at the initial login. Once a user is logged in, there are generally no checks to ensure that the person using the terminal is still the authorized person. User profiling seeks to address this by extracting "person specific" information as the user interacts with the computer. By comparing the user's activity with a profile of the user, it is hoped that masqueraders can be detected and locked out before they are able to do any damage.

We will discuss the usual host-based user profiling problem first, and then discuss a network based profiling application that has a similar flavor. The mathematics and statistics used for the two problems are very similar, only the data are different.

Several attempts have been made on this problem. Early work focused on utilizing keystroke timings. It was hoped that people had characteristic patterns of typing that could be discovered through measurement of the time between

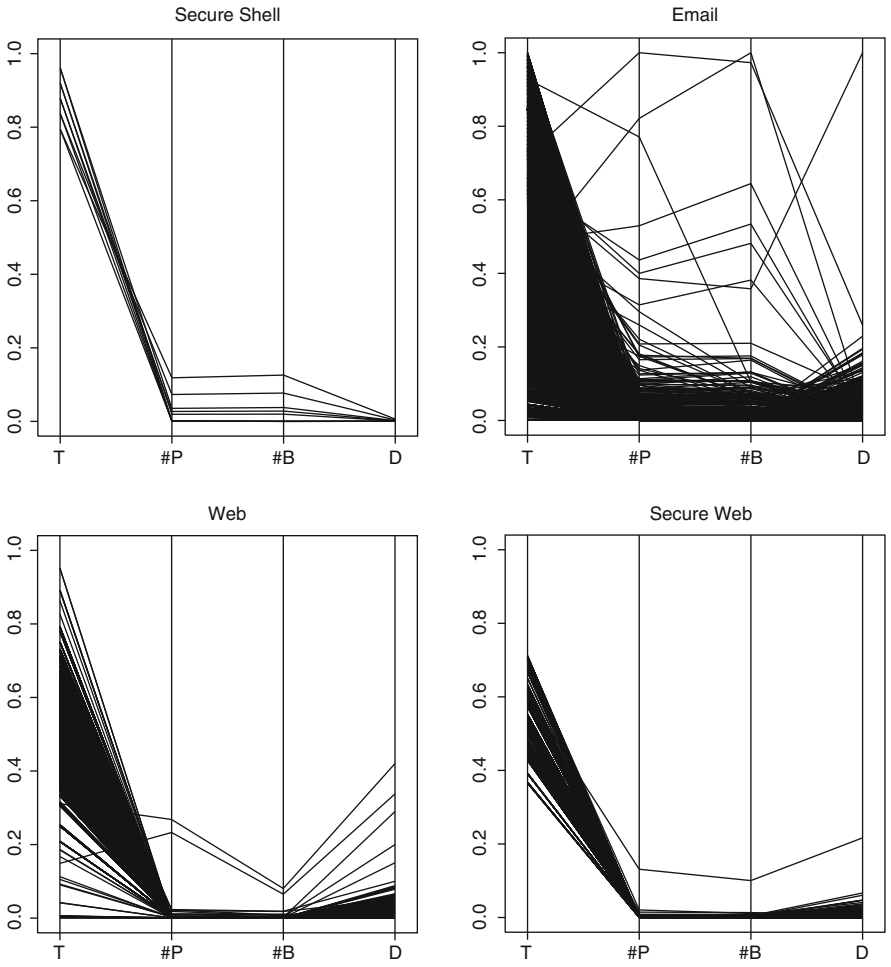


Fig. 38.9 Parallel coordinates plots of session statistics for four different applications. From left to right, top to bottom they are: secure shell, email, web and secure web. The coordinates are the time of the initiating SYN packet, the total number of packets, the total number of bytes sent and the duration of the session. The axes are all scaled the same among the plots

keystrokes for words or phrases. See for example Bleha et al. (1990), Obaidat and Sadoun (1997), Lin (1997) and Robinson et al. (1998).

This type of approach has been applied at the network level to crack passwords. Song et al. (2001) describes using simple statistical techniques applied to packet arrival timings to determine the length of passwords in secure shell, and even to allow for the cracking of passwords. Secure shell is an application that allows remote login via an encrypted pathway. It sends a packet for each character typed, to minimize the delay for the user. Thus, by timing the packets, one can get an idea

of what key combinations are being sent (it takes longer to type two characters with the same finger than it does if the characters are typed by fingers on different hands, for example). By utilizing statistics such as these, the authors were able to show that they could dramatically reduce the search space needed to crack the passwords.

A related approach looks at extracting information from encrypted communications, in particular encrypted voice over IP (VoIP). VoIP works by digitizing speech, compressing segments of the speech, then encrypting the segments and sending these out as packets over the Internet. If the encryption used retains the sizes of the packets, this information can be used to extract information about the text. In a Wright et al. (2007) the authors show that by analyzing the pattern of packet sizes, a classifier can be constructed to determine the language being spoken, even though the conversation itself remains encrypted. In fact, known phrases can be detected in the conversations. Figure 38.10 depicts one (English) utterance in the VoIP data. As

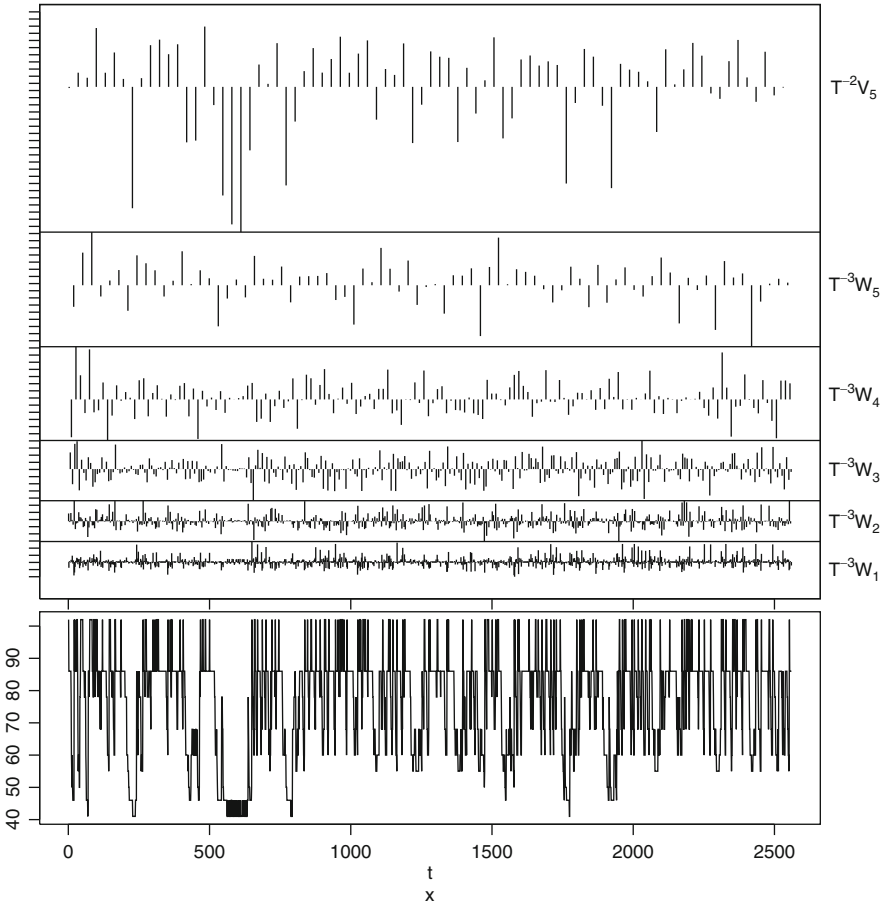


Fig. 38.10 Wavelet transform of a section of speech from the VoIP data

can be seen, there is quite a bit of structure in the data, and Wright et al. (2007) show that this structure can be exploited to extract information about the conversation. This kind of statistical analysis of encrypted information has important implications for privacy as well as security.

Other profiling work focuses on tracking user commands. The idea is that the command streams that users type (ignoring the arguments to the commands) could be used to authenticate the user in much the same way that keystroke timings could. A good discussion of this for statisticians can be found in Schonlau et al. (2001). See also Maxion (2003), Maxion and Townsend (2002) for some critiques of this work and extensions. The former paper considers arguments to the commands as well.

For Microsoft Windows operating systems, user command sequences are generally not applicable. Instead, window titles may be used. These correspond roughly to the same information that is contained in the Unix command lines. They typically contain the application name and the arguments to the applications such as the file open, the email subject, the web page visited, et cetera.

To illustrate this, we consider a set of data taken from six users on seven Windows NT machines over a period of several months. All window titles generated from the login to the logout were retained for each user/host pair (only one of the users was observed on a second host). Each time a window became active it was recorded. These data are a subset of a larger set. More information on these data, with some analysis of the data and performance of various classifiers can be found in DeVault et al. (2003).

Table 38.1 shows some statistics on these data. Three sessions are shown for each user/host pair. The length of the login session (in seconds), the name of the first and last applications used within the session, and the number of distinct applications, windows and window titles are shown. The task is to extract statistics from a completed login session that allow one to determine whether the user was the authorized user indicated by the userid. This is an easier problem than masquerader detection, in which one tries to detect the masquerader (or authenticate the user) as the session progresses, and it is not assumed that the entire session corresponds to a single user (or masquerader).

The table indicates that there is some variability among the sessions of individual users, and this is born out by further analysis. Table 38.2 shows the most common window titles. The number of times the title occurs in the data set, the number of login sessions in which the title occurs, and the title itself are shown. Some words in the titles have been obfuscated by replacement with numbers in double brackets, to protect the privacy of the users. All common application and operating system words were left alone. The obfuscation is consistent across all sessions: there is a bijection between numbers and words that holds throughout the data.

Figure 38.11 shows part of a single login session. The rows and columns correspond to the list of words (as they appear in the titles) and a dot is placed where the word appears in both the row and column. The blocks of diagonal lines are characteristic of a single window in session. The “plus” in the lower left corner shows a case of the user switching windows, then switching back. This type of behavior is seen throughout the data.

Table 38.1 Session statistics for three login sessions for each user/host pair

User	Session	Login Length	1st App	Last App	#Apps	#Wins	#Titles
User1-host19	3	30,794	msoffice	msoffice	6	13	134
User1-host19	5	28,788	msoffice	msoffice	8	15	194
User1-host19	6	19,902	msoffice	msoffice	10	25	267
User1-host5	1	3,472.47	explorer	explorer	3	6	34
User1-host5	2	142.98	explorer	explorer	2	3	6
User1-host5	40	21,912.79	explorer	explorer	7	25	187
User19-host10	5	31,432.5	msoffice	msoffice	7	8	133
User19-host10	6	16,886.3	msoffice	msoffice	6	7	75
User19-host10	11	2,615.55	msoffice	acrord32	6	8	45
User25-host4	2	28,362.82	explorer	explorer	4	19	382
User25-host4	3	45,578.82	explorer	explorer	5	16	316
User25-host4	12	6,788.44	explorer	explorer	4	11	102
User4-host17	10	19,445.96	wscript	explorer	8	21	452
User4-host17	30	6,310.72	explorer	explorer	3	5	60
User4-host17	44	17,326.21	explorer	winword	8	10	138
User7-host20	10	23,163.6	outlook	outlook	5	7	51
User7-host20	11	44,004.11	wscript	mapisp32	5	5	72
User7-host20	12	33,125.27	wscript	outlook	5	7	166
User8-host6	1	31,395.08	wscript	explorer	7	14	116
User8-host6	4	1,207.84	outlook	explorer	4	4	14
User8-host6	21	134.01	cmd	explorer	3	4	13

Table 38.2 Window title usage

#	#Sessions	Window Title
7002	425	Inbox – Microsoft Outlook
2525	411	Program Manager
2188	215	Microsoft Word
792	126	Netscape
704	156	Print
672	213	Microsoft Outlook
639	156	<<12761>> <<9227>>
592	170	<<16193>> – Message (<<16184>> <<5748>>)
555	174	<<6893>> <<13916>>
414	297	Microsoft(<<3142>>) Outlook(<<3142>>) <<7469>>
413	36	<<13683>> <<3653>> – Microsoft Internet Explorer
403	33	<<13683>> <<10676>> – Microsoft Internet Explorer
402	309	– Microsoft Outlook
401	61	Microsoft PowerPoint
198	84	http://<<1718>>.<<7267>>.<<4601>>/<<16345>>

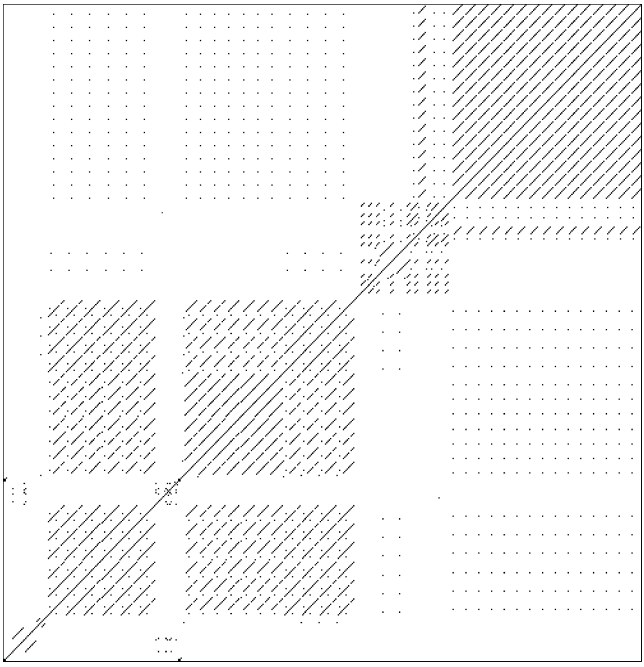


Fig. 38.11 First 500 words from a single session. The rows and columns correspond to words in the order in which they appear (*with duplicates*). A dot is plotted in (i, j) if the same word is in row i and column j

Many features were extracted from the data, and several feature selection and dimensionality reduction techniques were tried. The results for these approaches were not impressive. See DeVault et al. (2003) for more discussion.

The classifiers that worked best with these data were simple intersection classifiers. For each session, the total set of window titles used (without regard to order) was collected. Then to classify a new session, the intersection of its title set with those from user sessions was computed, and the user with the largest intersection was deemed to be the user of the session. Various variations on this theme were tried, all of which performed in the mid to high 90 percent range for correct classification.

Much more needs to be done to produce a usable system. Most importantly, the approach must move from the session level to within-session calculations. Further, it is not important to classify the user as one of a list of users, but to simply state whether the user's activity matches that of the userid. It may be straight forward to modify the intersection classifier (for example, set a threshold and if the intersection is below the threshold, raise an alarm) but it is not clear how well this will work.

We can state a few generalities about user profiling systems. Users are quite variable, and such systems tend to have an unacceptably high false alarm rate. Keystroke timings tend to be much more useful when used with a password or pass

phrase than in free typing. No single technique exists which can be used reliably to authenticate users as they work.

The intersection classifier leads to interesting statistics. We can construct graphs using these intersections, each node of the graph corresponding to a session, with an edge between two nodes if their sets intersect nontrivially (or have an intersection of size at least T).

In another context (profiling the web server usage of users) Marchette (2003) discusses various analyses that can be done on these graphs. This uses network data, extracting the source and destination IP addresses from the sessions. In these data there is a one-to-one correspondence between source IP address and user, since all the machines considered were single user machines.

In this case the nodes correspond to users and the sets consist of the web servers visited by the user within a period of a week. A random graph model, first described in Karonski et al. (1999) is used as the null hypothesis corresponding to random selection of servers. The model assumes a set $\mathcal{S}$ of servers from which the users draw. To define the set of servers for a given user, each server is drawn with probability p. Thus, given the observations of the sets S_i drawn by the users, we must estimate the two parameters of the model: $m = |\mathcal{S}|$ and p. These can be estimated using maximum likelihood (see also Marchette 2004 for discussion of this and other types of intersection graphs). With the notation

$$\begin{aligned} k_i &= |S_i| \\ M_i &= |\bigcup_{j=1}^{i} S_j| \\ u_i &= M_i - M_{i-1}, \end{aligned}$$

the likelihood is easily shown to be

$$L = \prod_{j=1}^{n} \binom{M_{j-1}}{k_j - u_j} \binom{m - M_{j-1}}{u_j} p^{k_j} (1-p)^{m-k_j}.$$

Using data collected for several months, Marchette (2003) computed the probability of any given edge, under the null hypothesis, and retained those that had a significantly large intersection (after correcting for the multiple hypotheses tested). The most common of these were retained, and the resulting graph is shown in Fig. 38.12.

There are two triangles in Fig. 38.12, and it turns out that the users in these correspond to physicists working on fluid dynamics problems. Users A, D and E are system administrators. Thus, there is some reason to believe that the relationships we have discovered are interesting.

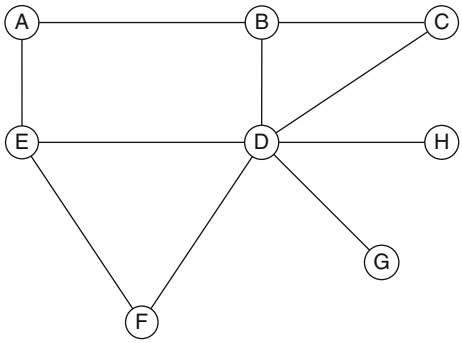


Fig. 38.12 A graph of the users with significantly large intersections. The edges for which the intersection size was statistically significant for 95% of the weeks are shown

The model is simplistic, perhaps overly so. It is reasonable to assume that users have different values of p, and some preliminary investigation (described in Marchette 2003) bears this out. This is an easy modification to make. Further, intuition tells us that perhaps all web servers should not have the same probabilities either. This is more problematic, since we cannot have a separate probability for each server and hope to be able to estimate them all. A reasonable compromise might be to group servers into common/rare groups or something similar.

The above discussion illustrates one of the methodologies used for anomaly detection. For determining when a service, server, or user is acting in an unusual manner, one first groups the entities using some model, then raises an alert when an entity appears to leave the group. Alternatively, one can have a single entity, for example "the network" or a given server, and build a model of the behavior of that entity under normal conditions. When the behavior deviates from these conditions by a significant amount, an alert is raised.

Other researchers have investigated the profiling of program execution, for the purpose of detecting attacks such as buffer overflows which can cause the program to act in an unusual way. See for example Forrest et al. (1994), Forrest et al. (1997), Forrest and Hofmeyr (In press), Tan and Maxion (2002). Programs execute sequences of system calls, and the patterns of system calls that occur under normal conditions are used to detect abnormal execution.

One of the first anomaly detection systems for intrusion detection was the Next-generation Intrusion Detection Expert System (NIDES) (Anderson et al. 1995). This system took a collection of statistics computed on both network and host data such as load average and usage statistics, packet rates, files accessed, protocols utilized, and so on, and combined them into a single statistic that was modeled as a Chi-squared statistic. Alarms were sounded for large values of the statistic, and the critical value could be determined either by reference to the Chi-squared distribution or empirically. A number of variations on this theme have been investigated.

One example is that of Ye et al. (2002). They propose using the Hotelling or χ^2 test: given a multivariate statistic X computed on a stream of data (such as one of

the collections of variables used in the NIDES approach), compute

$$T^2 = \frac{n(n-p)}{p(n+1)(n-1)}(X - \overline{X})' S^{-1}(X - \overline{X}) \tag{38.1}$$

$$\chi^2 = \sum_{j=1}^{p} \frac{(X_j - \overline{X}_j)^2}{\overline{X}_j} \tag{38.2}$$

where $\overline{X}$ is the mean and S is the sample covariance. If everything is normally distributed, these have known distributions, and critical values can be set for detection of anomalies. As always, in real problems it is advisable to set the critical values empirically, or at least to verify that the theoretical distributions are approximately correct. A similar approach is described in Oshima et al. (2009) for detection of denial of service attacks, and in Zhou et al. (2006) for user profiling.

Much work has been done profiling email activity, mostly for the detection of spam and phishing. These approaches may take a text-analysis approach (see for example Sasaki and Shinnou 2005), or a link analysis approach (Benczúr et al. 2005). A toy example of text mining approaches to spam detection is depicted in Fig. 38.13. Here, the text of the emails has been processed using a vector space

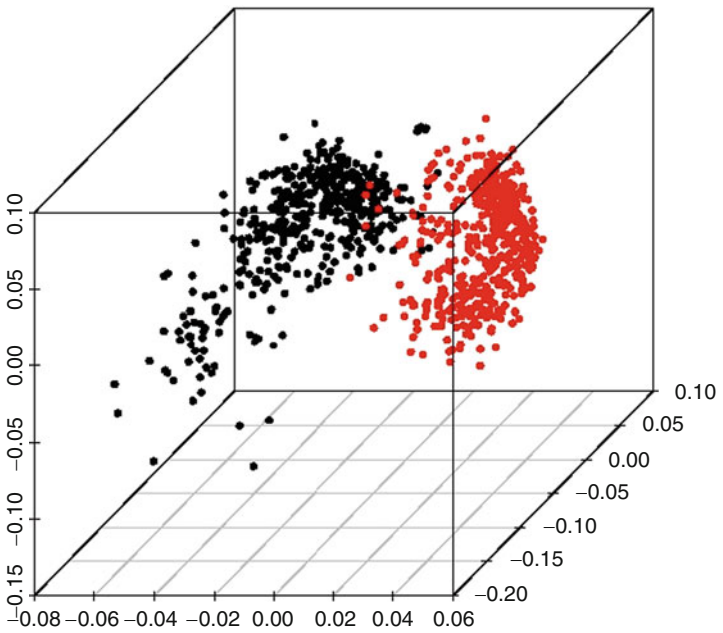


Fig. 38.13 A vector space model of email text on a small set of spam documents. Spam emails colored red, non-spam emails colored black

(bag-of-words) model. Each email message d is represented as a vector of values for each possible word w:

$$X[d,w] = \log\left(\frac{f(w,d)}{f(w,C)}\right), \tag{38.3}$$

where C is the corpus of all emails, and f denotes frequency: thus $f(w,d)$ is the frequency with which the word w occurs in the email (document) d, and $f(w,C)$ is the frequency with which the word occurs overall in the corpus of emails. Here, X corresponds to mutual information. As can be seen in the figure, for this tiny example corpus (about 400 emails), the spam separates quite well from the non-spam.

Spam has many other properties that must be taken into account to turn a simple like this into a useful approach to the problem, however. Spammers are constantly modifying their emails to defeat anti-spam filters. Adaptive approaches, and more sophisticated text analysis approaches, must be combined with approaches that utilize the way spam is addressed and routed to produce good reliable spam detection algorithms.

Email activity can also be profiled to look for unusual activity that could be indicative of either a virus propagating or an insider threat. Related work can be found in Priebe et al. (2005), where the problem of detecting an unusual pattern of activity in an email communications graph is considered. This work defined a scan statistic on graphs, analogous to the scan statistic approach used in many other applications for one and two dimensional data (Glaz et al. 2010). This approach uses scan statistics to detect regions in the time series of Enron communications graphs in which Enron executives had an unusual amount of communication ("chatter"). The Enron graphs correspond to directed graphs computed weekly: and edge exists from one email address to another if there was at least one email sent during the corresponding week.

The scan statistic for graphs is defined as follows. Given a directed graph G with vertex set $V = V(G)$ and edge set $E = E(G)$, the digraph distance $d(v,w)$ between two vertices $v, w \in V$ is defined to be the minimum directed path length from v to w in G. For a non-negative integer k and vertex $v \in V$ consider the closed kth-order neighborhood of v in G, denoted $N_k[v;G] = \{w \in V(G) : d(v,w) \leq k\}$. The scan region is the induced subdigraph of this neighborhood, denoted $\Omega(N_k[v;G])$. A locality statistic at location v and scale k is any specified digraph invariant $\Psi_k(v;G)$ of the scan region $\Omega(N_k[v;G])$. The scan statistic is the maximum of the locality statistics over the vertices.

In Priebe et al. (2005), a time series of graphs is considered, $G_1, \ldots, G_T$, and the locality statistics are normalized: given a window width τ, define

$$\mu(v,t) = \frac{1}{\tau}\sum_{i=1}^{\tau} \Psi_k(v;G_{t-i}) \tag{38.4}$$

$$\sigma^2(v,t) = \frac{1}{\tau - 1} \sum_{i=1}^{\tau} (\Psi_k(v; G_{t-i}) - \mu(v,t))^2 \tag{38.5}$$

$$\widetilde{\Psi}_k(v; G_t) = \frac{\Psi_k(v; G_t) - \mu(v,t)}{\max(1, \sigma(v,t))}. \tag{38.6}$$

Since the authors were looking for unusually large numbers of communications, the locality statistic was the size, the number of edges in the induced subgraph. By looking for large values of $\widetilde{\Psi}_k(v; G_t)$, anomalies can be detected. This work could be adapted to computer intrusion detection in several ways. The graph statistic could be modified to take the email content into account, which could be used to detect spam and viruses; other graphs invariants could be used to detect other types of patterns of activity; communications between computers could replace email communications between people.

Figure 38.14 shows an anomaly detection on the Enron graphs. Here "scan0" corresponds to vertex degree. Note that there is no detection when considering either vertex degree or a scan region with radius 1. It is only when one considers the 2-neighborhood that the detection arises. The statistics have been scaled (analogously to the scaling in (38.6)), and a detection threshold of 4 standard deviations is indicated in the plot.

Figure 38.15 depicts the 2-neighborhood corresponding to the detection in Fig. 38.14. The plot on the right shows the same vertices (with one other vertex) in the previous week. Note that in the previous week the 2-neighborhood was much smaller, and hence had fewer edges.

This detection is interesting, because the events discussed during this week are related to the California energy crisis. A white paper was circulated by economists indicating that the crisis was not the result of market forces. The Enron executives felt that they would be better served if they could find an economist willing to argue that the market was indeed the driving force, rather than manipulations by energy companies. While this is not the sole content of the emails during this week, it does provide an anecdote for the detection.

Adding information about the content of the emails turns the graph into an attributed graph – the edge attributes corresponding by the email topics. Preliminary work on this is reported in Priebe et al. (2010).

38.7 Discussion

There are many areas in which computational statistics can play a part in network intrusion detection and other security arenas. We have seen a few in this chapter, including modeling denial of service attacks, visualization, the analysis of streaming data applied to network data and profiling and anomaly detection.

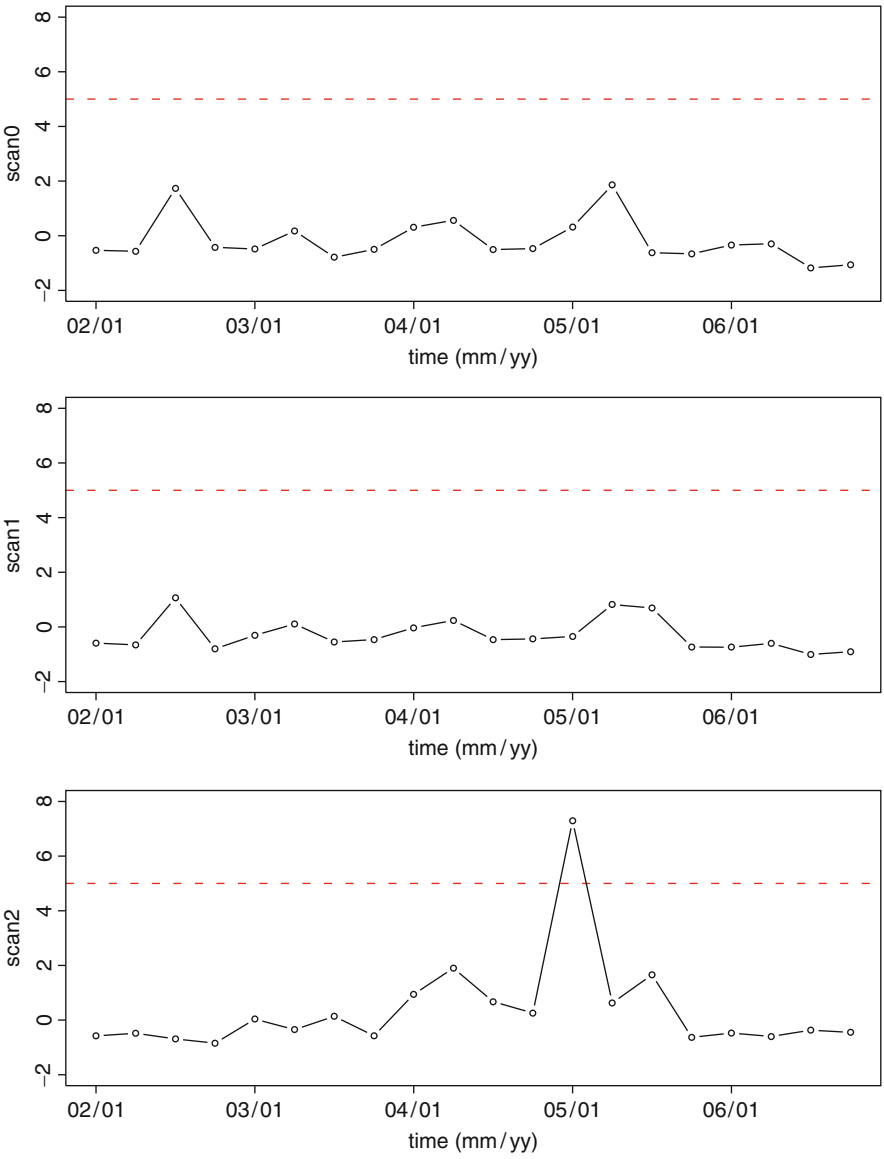


Fig. 38.14 Normalized scan statistics for the Enron graphs

The biggest problems for intrusion detection systems are the false alarm rates and the detection of novel attacks. The enormous amount of data that must be processed requires that false alarm rates must be extremely low. Typical network data consists of millions of packets an hour, and system administrators generally do not have time to track down more than a few false alarms a day. Signature based systems have the

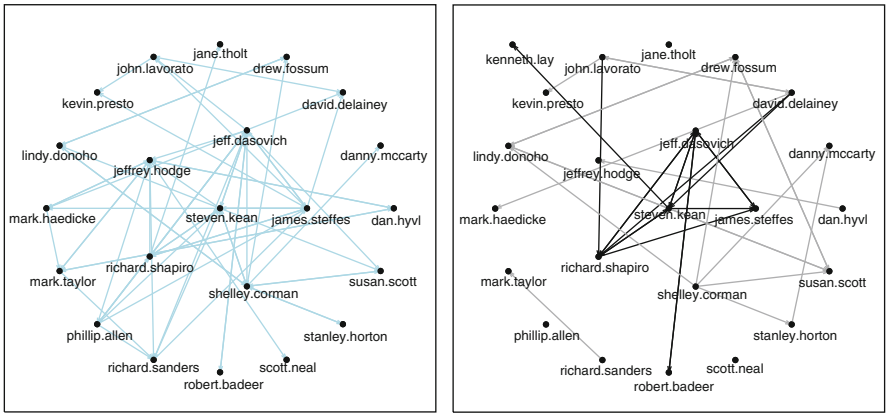


Fig. 38.15 The 2 neighborhood of the detection. In the left plot we see the graph at the detection week (*week 109*). The right plot shows the two neighborhood in the previous week (*dark edges*), with the light edges showing the edges for the rest of the vertices that appear in the two neighborhood in week 109

advantage that they rarely false alarm (assuming the signature is properly defined), but they tend to have poor performance on novel attacks. Thus it is essential that techniques be found that detect novelty that is "bad" without alarming on novelty that is benign.

One area we have not discussed is modeling attack propagation. Early work on this can be found in Kephart and White (1991, 1993). See also Wierman and Marchette (2004) for a related model. For a discussion of the slammer worm, see http://www.cs.berkeley.edu/~nweaver/sapphire/ The slammer worm was interesting because the spread was self-limiting: the worm spread so fast that the available bandwidth was reduced to the point that the worm as unable to continue to spread at its initial rate. Models for these types of worms is an interesting area of study.

References

Amoroso, E.: Intrusion Detection: An Introduction to Internet Surveillance, Correlation, Trace Back, Traps, and Response. Intrusion.net Books, Sparta, New Jersey (1999)

Anderson, D., Lunt, T.F., Javitz, H., Tamaru, A., Valdes, A.: Detecting unusual program behavior using the statistical component of the next-generation intrusion detection expert system (nides). Technical Report SRI-CSL-95-06, SRI International (1995)

anonymous: *Maximum Security*. Sams.net Publishing, Indianapolis, IN (1997)

Bace, R.G.: Intrusion Detection. MacMillan Technical Publishing, Indianapolis, IN (2000)

Benczúr, A.A., Csalogáy, K., Sarlós, T., Uher, M.: Spamrank – fully automatic link spam detection. In Proceedings of the First International Workshop on Adversarial Information Retrieval on the Web, pp. 25–38 (2005)

Bleha, S., Slivinsky, C., Hussien, B.: Computer-access security systems using keystroke dynamics. IEEE Trans. Pattern Anal. Mach. Intell. **12**(12), 1217–1222 (1990)

DeVault, K., Tucey, N., Marchette, D.: Analyzing process table and window title data for user identification in a windows environment. Technical Report NSWCDD/TR-03/122, Naval Surface Warfare Center (2003)
Early, J.P., Brodley, C.E.: Behavioral authentication of server flows. In The 19th Annual Computer Security Applications Conference, pp. 49–55 (2003)
Escamilla, T.: Intrusion Detection: Network Security Beyond the Firewall. Wiley, New York (1998)
Forrest, S., Hofmeyr, S.A.: Immunology as information processing. In: Segel, L.A., Cohen, I. (eds.) Design Prinicples for the Immune System and Other Distributed Autonomous Systems, Santa Fe Institute Studies in the Sciences of Complexity. Oxford University Press, Oxford, UK, 361–387, (2000) Also available at www.cs.unm.edu/∼forrest/ism_papers.htm.
Forrest, S., Perelson, A.S., Allen, L., Cherukuri, R.: Self-nonself discrimination in a computer. In 1994 IEEE Symposium on Research in Security and Privacy, pp. 202–212. Los Alamitos, CA (1994); Also available at www.cs.unm.edu/∼forrest/isa_papers.htm.
Forrest, S., Hofmeyr, S.A., Somayaji, A.: Computer immunology. Comm. ACM **40**, 88–96 (1997)
Giles, K., Marchette, D.J., Priebe, C.E.: A backscatter characterization of denial-of-service attacks. In Proceedings of the Joint Statistical Meetings, CDROM (2003)
Glaz, J., Naus, J., Wallenstein, S.: Scan Statistics. Springer, New York (2010)
Karonski, M., Singer, K., Scheinerman, E.: Random intersection graphs: the subgraph problem. Combinator. Probab. Comput. **8**, 131–159 (1999)
Kephart, J.O., White, S.R.: Directed-graph epidemiological models of computer viruses. In Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy, pp. 343–359 (1991)
Kephart, J.O., White, S.R.: Measuring and modeling computer virus prevalence. In Proceedings of the IEEE Computer Society Symposium on Research in Security and Privacy, pp. 2–15 (1993)
Lin, D.-T.: Computer-access authentication with neural network based keystroke identity verification. In International Conference on Neural Networks, pp. 174–178 (1997)
Marchette, D.J.: Computer Intrusion Detection and Network Monitoring: A Statistical Viewpoint. Springer, New York (2001)
Marchette, D.J.: A study of denial of service attacks on the internet. In Proceedings of the Army Conference on Applied Statistics (2002); 41–40, available at http://www.armyconference.org/ACAS00-02/Master02.pdf
Marchette, D.J.: Passive detection of denial of service attacks on the internet. In: Chen, W. (eds.) Statistical Methods in Computer Security. Marcel Dekker; 183–211.
Marchette, D.J.: Profiling users by their network activity. In Proceedings of the Joint Statistical Meetings 219–228 (2003).
Marchette, D.J.: Random Graphs for Statistical Pattern Recognition. Wiley, New York (2004)
Maxion, R.A.: Masquerade detection using enriched command lines. In International conference on dependable systems and networks(DNS-03). IEEE Computer Society Press, Washington, DC (2003)
Maxion, R.A., Townsend, T.N.: Masquerade detection using truncated command lines. In International conference on dependable systems and networks(DNS-02). IEEE Computer Society Press, Washington, DC (2002)
Moore, D., Voelker, G.M., Savage, S.: Infering Internet denial-of-service activity. In Proceedings of the 2001 USENIX Security Symposium, pp. 9–22 (2001). Available on the web at www.usenix.org/publications/library/proceedings/sec01/moore.html USENIX Security '01.
Northcutt, S., Novak, J., McLaclan, D.: Network Intrusion Detection. An Analyst's Handbook. New Riders, Indianapolis, IN (2001)
Obaidat, M.S., Sadoun, B.: Verification of computer users using keystroke dynamics. IEEE Trans. Syst. Man Cybernetics **27**(2), 261–269 (1997)
Oshima, S., Nakshima, T., Nishikido, Y.: Extraction of characteristics of anomaly accessed IP packets using chi-square method. In Complex, Intelligent and Softawre Intensive Systems, CISIS '09, pp. 287–292 (2009)
Priebe, C.E.: Adaptive mixture density estimation. J. Am. Stat. Assoc. **89**, 796–806 (1994)

Priebe, C.E., Conroy, J.M., Marchette, D.J., Park, Y.: Scan statistics on enron graphs. Comput. Math. Organ. Theor. **11**, 229–247 (2005)

Priebe, C.E., Park, Y., Marchette, D.J., Conroy, J.M., Grothendieck, J., Gorin, A.L.: Statistical inference on attributed random graphs: Fusion of graph features and content: An experiment on time series of enron graphs. Comput. Stat. Data Anal. **54**, 1766–1776 (2010)

Proctor, P.E.: The Practical Intrusion Detection Handbook. Prentice-Hall, Englewood Cliffs, NJ (2001)

Robinson, J.A., Liang, V.M., Chambers, J.A.M., MacKenzie, C.L.: Computer user verification using login string keystroke dynamics. IEEE Trans. Syst. Man Cybernetics **28**(2), 236–241 (1998)

Sasaki, M., Shinnou, H.: Spam detection using text clustering. International Conference on Cyberworlds. **0**, 316–319 (2005); http://doi.ieeecomputersociety.org/10.1109/CW.2005.83.

Schonlau, M., DuMouchel, W., Ju, W.-H., Karr, A.F., Theus, M., Vardi, Y.: Computer intrusion: Detecting masquerades. Stat. Sci. **16**, 58–74 (2001)

Song, D.X., Wagner, D., Tian, X.: Timing analysis of keystrokes and timing attacks on SSH. In Proceedings of the 10th USENIX Security Symposium (2001); http://www.usenix.org/publications/library/proceedings/sec01/song.html.

Stevens, W.R.: TCP/IP Illustrated, Volume 1: The Protocols. Addison-Wesley, Reading, MA (1994)

Tan, K.M.C., Maxion, R.A.: "Why 6?" defining the operational limits of stide, an anomaly-based intrusion detector. In IEEE Symposium on Security and Privacy. IEEE Computer Society Press, Washington, DC (2002)

Wegman, E.J., Davies, H.I.: Remarks on some recursive estimators of a probability density. Ann. Stat. **7**, 316–327 (1979)

Wegman, E.J., Dorfman, A.: Visualizing cereal world. Technical Report TR 178, George Mason University, Center for Computational Statistics (2001)

Wegman, E.J., Marchette, D.J.: On some techniques for streaming data: a case study of Internet packet headers. JCGS (2003), **12**(4), 893–914.

Wierman, J.C., Marchette, D.J.: Modeling computer virus prevalence with a susceptible-infected-susceptible model with reintroduction. Computational Statistics and Data Analysis (2004), **45**(1), 3–23.

Wilhelm, A.F.X., Wegman, E.J., Symanzik, J.: Visual clustering and classification: The oronsay particle size data set revisited. Comput. Stat. 109–146 (1999)

Wright, C.V., Ballard, L., Monrose, F., Masson, G.M.: Language identification of encrypted voip traffic: Alejandra y roberto or alice and bob? In SS'07: Proceedings of 16th USENIX Security Symposium on USENIX Security Symposium, pp. 1–12, Berkeley, CA, USA (2007); USENIX Association.

Wulf, W.A., Jones, A.K.: Reflections on cybersecurity. Science **326**, 943–944 (2009)

Yamato, H.: Sequential estimation of a continuous probability density function and the mode. Bulletin Math. Stat. **14**, 1–12 (1971)

Ye, N., Emran, S.M., Chen, Q., Vilbert, S.: Multivariate statistical analysis of audit trails for host-based intrusion detection. IEEE Trans. Comput. **51**, 810–820 (2002)

Zhou, B., Shi, Q., Merabti, M.: Intrusion detection in pervasive networks based on a chi-square statistic test. In Computer Software and Applications Conference, COMPSAC '06, vol. 2, pp. 203–208 (2006)

Index

J.E. Gentle et al. (eds.), *Handbook of Computational Statistics*, Springer Handbooks of Computational Statistics, DOI 10.1007/978-3-642-21551-3,