Sheela Ramanna
Lakhmi C. Jain
Robert J. Howlett
Editors

SMART INNOVATION, SYSTEMS AND TECHNOLOGIES ■ 13

Emerging Paradigms in Machine Learning

Springer

Sheela Ramanna, Lakhmi C. Jain,
and Robert J. Howlett (Eds.)

Emerging Paradigms in Machine Learning

Springer

Editors
Prof. Sheela Ramanna
Department of Applied Computer Science
University of Winnipeg
Winnipeg
Canada

Prof. Lakhmi C. Jain
School of Electrical and Information Engineering
University of South Australia
Mawson Lakes Campus
Adelaide
Australia

Prof. Robert J. Howlett
Executive Chair
KES International
Visiting Professor
Enterprise: Bournemouth University
United Kingdom

ISSN 2190-3018 e-ISSN 2190-3026
ISBN 978-3-642-28698-8 e-ISBN 978-3-642-28699-5
DOI 10.1007/978-3-642-28699-5
Springer Heidelberg New York Dordrecht London

Library of Congress Control Number: 2012935113

Printed on acid-free paper

Springer is part of Springer Science+Business Media (www.springer.com)

Foreword

KES International (KES) is a worldwide organisation that provides a professional community and association for researchers, practitioners and other stakeholders originally in the discipline of Knowledge Based and Intelligent Engineering Systems, and other synergetic areas. Through this, KES provides its members with opportunities for publication and beneficial interaction.

The focus of KES was originally research and technology transfer in the area of Intelligent Systems, i.e. computer-based software systems that operate in a manner analogous to the human brain, in order to perform advanced tasks. However, KES has extended its scope to encompass sustainability and renewable energy, and also the knowledge transfer, innovation and enterprise agenda.

Involving several thousand researchers, managers and engineers drawn from universities and companies world-wide, KES is in an excellent position to facilitate international research co-operation and generate synergy in the area of artificial intelligence applied to real-world 'Smart' systems and the underlying related theory.

The KES annual conference covers a broad spectrum of intelligent systems topics and attracts several hundred delegates from a range of countries round the world. KES also organises symposia on specific technical topics, for example, Agent and Multi Agent Systems, Intelligent Decision Technologies, Intelligent Interactive Multimedia Systems and Services, Sustainability in Energy and Buildings and Innovation through Knowledge Transfer. KES is responsible for two peer-reviewed journals, the International Journal of Knowledge based and Intelligent Engineering Systems, and Intelligent Decision Technologies: an International Journal.

KES supports a number of book series in partnership with major scientific publishers.

Published by Springer, 'Smart Innovative Systems and Technologies' is the KES flagship book series. The aim of the series is to make available a platform for the publication of books (in both hard copy and CRROM form) on all aspects of single and multi-disciplinary research involving smart innovative systems and technologies, in order to make the latest results available in a readily-accessible form.

The series covers systems that employ knowledge and intelligence in a broad sense. Its focus is systems having embedded knowledge and intelligence, which may be applied to the solution of world industrial, economic and environmental problems and the knowledge-transfer methodologies employed to make this happen effectively. The combination of intelligent systems tools and a broad range of applications introduces a need for a synergy of scientific and technological disciplines.

Examples of applicable areas to be covered by the series include intelligent decision support, smart robotics and mechatronics, knowledge engineering, intelligent multi-media, intelligent product design, intelligent medical systems, smart industrial products, smart alternative energy systems, and underpinning areas such as smart systems theory and practice, knowledge transfer, innovation and enterprise.

The series includes conference proceedings, edited collections, monographs, handbooks, reference books, and other relevant types of book in areas of science and technology where smart systems and technologies can offer innovative solutions.

High quality is an essential feature for all book proposals accepted for the series. It is expected that editors of all accepted volumes take responsibility for ensuring that contributions are subjected to an appropriate level of reviewing process and adhere to KES quality principles.

Professor Robert J. Howlett
Executive Chair, KES International
Visiting Professor, Enterprise: Bournemouth University
United Kingdom

Foreword

Paradigms for machine learning tend to emerge from nature studies. The penultimate example of this is in the concomitant discovery of an approach to intelligent data analysis in terms of elementary sets and a five-decade long study of nature by Z. Pawlak[1] that began in the 1950s. The rough set approach and the companion approaches provided by fuzzy set theory and the underlying contributions arising from various information granulation frameworks have proved to be fundamentally important in the discovery of paradigms for artificial intelligence and machine learning. In particular, structures from general topology[2] and proximity space theory[3] appear to be very useful.

A few examples from this volume serve to illustrate the remarkably powerful set of paradigms for machine learning in this volume. In terms of the yield from a machine learning perspective, one might consider the discovery of information granules that are a biproduct of structured thinking (Y. Yao and X. Deng), filtered sets of patterns (R.H. Estevam, M.C. Nicoletti), certain and possible rule sets induced from lower and upper approximations, respectively (J.W. Grzymała-Busse and Z.S. Hippe), interpretability and representativeness of information granules (A. Bargiela, W. Pedrycz), observational and higher level granules emerging from agent-environment interaction (A. Skowron, P. Wasilewski), box plots exhibiting the

[1] See, *e.g.*, Z. Pawlak and A. Skowron, Rough sets and intelligent data analysis, Info. Sci. 147, 2002, 1-12, and J.F. Peters, How near are Zdzisław Pawlak's paintings? Merotopic distance between regions of interest, in A. Skowron, S. Suraj, Eds., Intelligent Systems Reference Library volume dedicated to Prof. Zdzisław Pawlak, Springer, Berlin, 2012, *to appear*.

[2] See, *e.g.*, L. Polkowski, Rough Sets. Mathematical Foundations, Springer-Verlag, Berlin, 2002, and L. Polkowski, Approximate reasoning by parts: An Introduction to Rough Mereology, Springer, Heidelberg, 2011. See, also, T.Y. Lin, Topological and fuzzy rough sets, in R. Słowiński, Ed., Handbook of Applications and Advances of Rough Set Theory, Kluwer, Amsterdam, 1992, 287-304.

[3] See, *e.g.*, J.F. Peters, A. Skowron, J. Stepaniuk, Nearness of objects: Extensions of the approximation space model, Fund. Inform. 79 (3-4), 2007, 497-512.

performance of sets of classifiers and a scheme such as *input raw data→pattern detection→consequent action* (B.J. Oommen, C. Bellinger), support vector machines in classifying mass spectra (K.A. Cyran et al.), clusters created and excluded in evolving intelligent systems (A. Lemos, W. Caminhas, F. Gomide), discovery of patterns from different sources of data such as imbalances and/or noisy data (J. Stefanowski) or distributed data (H. Rybiński, D. Ryżko, P. Więch), and optimization of patterns discovery by using dynamic programming (A. Alkhalid, I. Chikalov, S. Hussain, M. Moshkov).

Basically, if we are in the pursuit of various forms of machine learning, it can be seen from the paradigms in this volume that we want to teach machines to recognize and classify patterns[4]. A beneficial side-effect of the proposed approaches to machine learning in this volume is the design of thinking machines that help us improve our circumstances (*e.g.*, navigation and obstacle avoidance by smart vehicles) and our environment (*e.g.*, control of the number of active cooling devices needed to achieve satisfactory comfort levels). In sum, taken as a whole, the proposed paradigms for machine learning also have the potential to benefit our way of life. The Editors of this impressive volume are to be congratulated on their assembly of excellent chapters on machine learning.

December 2011

James F. Peters
Andrzej Skowron

[4] This is a well-known machine learning paradigm. For details, see, *e.g.*, the chapter by B.J. Oommen and C. Cellinger in this volume.

Preface

This book presents i) fundamental topics and algorithms that form the core of machine learning (ML) research, ii) emerging paradigms in intelligent system design. In the early days of machine learning, there was considerable interest in philosophical, logical and conceptual issues[1][2][3]. Over time, research interest shifted to computational and algorithmic aspects of ML driven mainly by practical applications. Since the first Machine Learning Workshop[4] held in 1980 in Pittsburgh, PA, several disciplines such as artificial intelligence, adaptive control systems, biology, philosophy, psychological models and statistics, have contributed to the richness and diversity of research in this field.

The very multidisciplinary nature of machine learning makes it a very fascinating and popular area for research. This book attempts to capture the diversity and richness of the field of machine learning and intelligent systems. Several chapters devoted to computational learning models such as granular computing, rough sets and fuzzy sets should be of interest to students, practitioners and researchers. An account of applications of well-known learning methods in biometrics, computational stylistics, multi-agent systems, spam classification including an extremely well-written survey on Bayesian networks shed light on the strengths and weaknesses of the methods. Practical studies yielding insight into challenging problems such as learning from incomplete and imbalanced data, pattern recognition of stochastic episodic events and on-line mining of non-stationary data streams are a key part of this book.

[1] Minsky, M. (1961). Steps toward artificial intelligence. Proc. IRE 49:8-30.
[2] Sutherland, N. S. (1968). Outlines of a theory of visual pattern recognition in animals and man. Proc. Royal Soc. B 171:297-317.
[3] Nelson, R. J. (1976). On mechanical recognition. Phil. Sci. 43(1):24-52.
[4] http://www.machinelearning.org/icml.html

The Editors of this book extend their gratitude to the authors and reviewers of chapters. Without their contribution and generosity, this book would not have been possible. Prof. Sheela Ramanna has been supported by funding from the Natural Science and Engineering Research Council of Canada (NSERC).

Sheela Ramanna
Canada

Lakhmi C. Jain
Australia

Robert J. Howlett
United Kingdom

Contents

Editors

Brief Biography

Sheela Ramanna is a Full Professor and Head of the Applied Computer Science Department at the University of Winnipeg. She received a Ph.D. in Computer Science from Kansas State University, U.S.A. and a B.E. in Electrical Engineering and M.Tech.in Computer Science and Engineering from Osmania University, Hyderabad, India. She is the Managing Editor for Springer Transactions on Rough Sets Journal (TRS). She serves on the Editorial Board of TRS, Journal of Intelligent Decision Technology (IOS Press), Int. Journal of Advanced Intelligent Paradigms (Inderscience), and Journal of Agents and Multi-Agent Systems. She served as Program Co-Chair for RSKT2011, RSCTC2010 and JRS2007. Her paper on rough control co-authored with James F. Peters received the IFAC Best Paper Award in 1998. She has served on numerous program committees for international conferences and is a reviewer for several international journals. She has published numerous articles on the theory and application of computational intelligence techniques (rough and near sets, fuzzy sets and neural networks) in journals, conferences, books and edited volumes. She is a member of the Computational Intelligence Laboratory, University of Manitoba, Canada and a Visiting Professor in the Department of Computer and Information Sciences, University of Hyderabad, India. Her research interests include theory and applications of rough sets, near sets and fuzzy sets in intelligent systems.

Professor Lakhmi C. Jain is a Director/Founder of the Knowledge-Based Intelligent Engineering Systems (KES) Centre, located in the University of South Australia. He is a fellow of the Institution of Engineers Australia. His interests focus on the artificial intelligence paradigms and their applications in complex systems, art-science fusion, virtual systems, e-education, e-healthcare, unmanned air vehicles and intelligent agents.

Prof. Robert (Bob) J. Howlett PhD, MPhil, BSc(Hons) is a member of the Institution of Engineering and Technology and the British Computer Society. He is a Chartered Engineer and a Chartered Information Technology Practitioner.

Bob is the Executive Chair of the KES (Knowledge-Based and Intelligent Engineering Systems) International, an organisation dedicated to supporting and facilitating research, in three areas: intelligent systems; sustainability in energy and buildings; and innovation and knowledge transfer.

Bob is a Director of the Institute of Knowledge Transfer. He is a nationally known figure in knowledge transfer and the UK Government Knowledge Transfer Partnerships (KTP) programme. His work on knowledge transfer and innovation has been recognised by his appointment as Visiting Professor, Enterprise, at Bournemouth University. He has supervised about 20 projects transferring university expertise to companies, mainly small to medium enterprises (SMEs), and has set up many more.

He has a number of years experience of applying neural-networks, expert systems, fuzzy paradigms and other intelligent techniques to industrial problem domains e.g. sustainability; control, modelling and simulation of renewable energy systems; monitoring and control of internal combustion engines, particularly small engines for off road and power generation applications; and a range of condition monitoring and fault diagnosis problems. He led a research team, funded by grants and industrial contracts. He has published widely on the subject, and has presented invited talks, keynote addresses etc.

He is Editor-in-Chief of the International Journal of Knowledge-Based Intelligent Engineering Systems and Honorary Editor of Intelligent Decision Technologies: an International Journal. He supports a number of other journals through regional editorships and membership of advisory and review boards. He is a past and current member of the scientific committees of a number of conferences. He is on the editorial board of various book series. He has authored over 50 publications in refereed journals and conferences, and edited over 20 books. He has reviewed research project applications for the EPSRC, and internationally as an Expert Evaluator under Framework and for other EU programmes.

Chapter 1
Emerging Paradigms in Machine Learning: An Introduction

Sheela Ramanna, Lakhmi C. Jain, and Robert J. Howlett

Abstract. This chapter provides a broad overview of machine learning (ML) paradigms both emerging as well as well-established ones. These paradigms include: Bayesian Learning, Decision Trees, Granular Computing, Fuzzy and Rough Sets, Inductive Logic Programming, Reinforcement Learning, Neural Networks and Support Vector Machines. In addition, challenges in ML such as imbalanced data, perceptual computing, and pattern recognition of data which is episodic as well as temporal are also highlighted.

1.1 Introduction

Learning is acquiring new or modifying existing knowledge, behaviors, skills, values, or preferences and may involve synthesizing different types of information [6]. Learning in animals and humans has been explored by zoologists and psychologists alike. Many techniques in machine learning derive from the efforts of psychologists to make more precise their theories of animal and human learning through computational models [22]. Machine learning, a branch of artificial intelligence, is concerned with the design and development of algorithms that allow computers to evolve behaviors based on empirical data, such as from sensor data or databases. Several disciplines such as artificial intelligence [20, 12, 33, 21],

Sheela Ramanna
Professor and Chair
Dept of Applied Computer Science, University of Winnipeg, 515 Portage Avenue, Winnipeg MB R3B 2E9, Canada

Lakhmi C. Jain
School of Electrical and Information Engineering, University of South Australia, Adelaide, Mawson Lakes Campus, South Australia SA 5095, Australia

Robert J. Howlett
Professor, Enterprise: Bournemouth University
Executive Chair KES International, PO Box 2155, Shoreham-by-sea, BN43 9AF, United Kingdom

adaptive control systems [18], evolutionary models based on biology [15], statistics [35, 14], psychological models [13] to name a few have contributed to machine learning. Some of these disciplines have been explored in-depth either as foundations or applications in this book. Specifically, these approaches can be broadly categorized as follows:

Artificial Neural Networks: An Artificial Neural Network (ANN) or neural networks is a computational model inspired by the structure and functional aspects of biological neural networks [17, 33]. The greatest advantage of ANNs is their ability to be used as an arbitrary function approximation mechanism that learns from observed data and as a result have found wide use in such diverse areas as business, science and engineering. Chapter 17 discusses an application in stylistic text classification.

Bayesian Networks: Bayesian reasoning provides the basis for learning algorithms that directly manipulate probabilities, as well as a framework for analyzing the operation of other algorithms that do not explicitly manipulate probabilities [19]. A Bayesian network can thus be considered a mechanism for automatically applying Bayes' theorem [3] to complex problems. Bayesian learning have been used extensively in solving problems in artificial intelligence in general and machine learning and intelligent systems in particular. Chapter 5 investigates Bayesian Networks as a formalism for different machine learning tasks.

Decision Tree Learning: Decision Trees(DT) are one of the most popular classification algorithms currently in use in data mining and machine learning [19]. Machine learning methods largely benefit from optimization techniques in order to find an optimal model for future predictions and decisions. However, due to massive and large-scale data sets faced in real world problems, optimization becomes a challenging task. Chapter 2 introduces an algorithm for optimization of decision trees based on an extension of dynamic programming. Chapter 18 gives an application of DT in image spam classification.

Fuzzy Sets and On-Line Mining: To address the challenges of highly interpretable knowledge from streams of data, techniques where structure and functionality of a system evolves based on incoming non-stationary data streams is necessary. Evolving intelligent Systems (EIS) is seen as providing a synergy between fuzzy [40] or neuro-fuzzy techniques and online methods in machine learning [1]. Chapter 6 provides some insights into EIS and Chapter 13 presents an application of a fuzzy expert system.

Granular Computing: Granular computing entails constructing, processing and reasoning of systems at the level of information granules. Information granules can be treated as linked collections (clumps) of objects drawn together by the criteria of indiscernibility, similarity, or functionality [39]. Granular computing brings together existing formalisms of set theory (interval analysis), fuzzy sets, rough sets [27]. Since the coining of the phrase granular computing, many researchers have contributed extensively to its growth. It is in fact, an emerging paradigm for information processing and systems modelling with roots in [16]. Chapters 3, 10 and 12 are devoted to granular computing encompassing methods for information abstraction, role of interaction in perceptual processes and concept learning.

Inductive Logic Programming: The field of Inductive Logic Programming (ILP) essentially combines machine learning with logical knowledge representation. What defines and unifies much of ILP is the relational form of the input, rather than the first-order logic form of the output. This has led to a broadening of the field to relational learning and is now seen as a key area within machine learning. Logic programs provide a good representation for the generalizations required to make predictions. They are also more expressive in the form of easily understandable sentences than alternative representations that are sometimes used today to cope with data such as network and graph-based representations [37]. Chapter 8 discusses an approach to ML based in multi-agent system based on inductive logic programming.

Perceptual Computing: One of the main challenges in developing intelligent systems is to discover methods for learning and reasoning from dynamically evolving complex sensory measurements. Perception Based Computing is another emerging research area whose roots are derived from experimental psychology and especially from psychophysics [38, 11]. Papers related to models of perceptual learning where process are independent from conscious forms of learning and involve structural and/or functional changes in the primary sensory cortices can be found in [10]. Chapter 10 presents an introduction to perception based computing and discusses the role of interactions in modeling of perceptual processes.

Reinforcement Learning: A reinforcement learning system is a system that learns under the influence of reinforcement where reinforcement is feedback from the environment that assigns credit to the learning system's action [20, 34]. In other words, an autonomous agent interacts with an external environment by selecting actions and seeks the sequence of actions that maximizes its long-term performance. In particular, great strides have been made in approximating value functions, exploring efficiently, learning in the presence of multiple agents, coping with partial observability, inducing models, and reasoning hierarchically [36]. A discussion of reinforcement learning with rough sets can be found in [32, 31].

Rough Sets: Rough set theory, proposed by Pawlak [23] can be viewed as a well-established mathematical approach to vagueness. Rough set theory it is not an alternative to classical set theory but it is embedded in it. Rough set theory can be viewed as a specific implementation of Frege's idea of vagueness, i.e., imprecision is expressed by a boundary region of a set [24]. The rough set approach has led to its various generalizations [29, 30] and many interesting applications. Rough set theory is fundamentally important in research areas such as granular computing, decision analysis, intelligent systems, machine learning, pattern recognition, and knowledge discovery to name a few [25, 26]. Several chapters

(4, 9, 12 and 17) in this book are devoted to rough set research involving granular computing, machine learning and perception based computing.

Support Vector Machines: Support Vector Machines(SVMs) have been successfully extended from basic classification tasks to handle regression, operator inversion, density estimation, novelty detection, and to include other desirable traits, such as invariance under symmetries and robustness in the presence of noise [5, 35, 7, 8]. In addition to their accuracy, a key characteristic of SVMs is their mathematical tractability and geometric interpretation. This has facilitated a rapid growth of interest in SVMs not only for pattern recognition problems, but also in image analysis [9]. Chapter 15 presents an application of SVM in face-recognition problems.

Challenges in Machine Learning: One challenging topic in machine learning is learning from imbalanced data where one class is underrepresented in comparison to the remaining classes. Chapter 10 presents a study of this topic. Another important challenge is pattern recognition of Stochastically Episodic (SE) events involving intervowen classes and data that is temporal in nature. Chapter 7 presents a study of this topic and Chapter 13 discusses discovery of patterns in time-stamped data.

1.2 Chapters of the Book

This book includes 18 chapters. What follows is a brief description of each chapter.

Chapter 2 introduces an algorithm for optimization of decision trees based on an extension of dynamic programming which allows for sequential optimization relative to different cost functions, and to the study of relationships between depth and number of misclassifications of decision trees. In particular, this chapter considers $\boldsymbol{\alpha}$ -decision trees where the parameter $\boldsymbol{\alpha}$ controls the accuracy of tree. The use of the proposed technique is illustrated by experiments with some datasets from UCI ML Repository.

Chapter 3 discusses an approach to information abstraction based on a form of min/max clustering where granules are represented as hyper boxes in multidimensional feature space. This study focuses on two essential characteristics of information abstraction: the interpret ability and the representativeness of information granules. In particular, an information density-based granulation algorithm that introduces an extra stage of optimised refinement of granular prototypes is discussed. Efficacy of this algorithm is illustrated by experiments on a synthetic data set.

Chapter 4 discusses a rough-set methodology (MLEM2) for mining incomplete data i.e., data with missing attribute values, since many real-life data are incomplete. In particular, two versions (global and local) of the parallel Modified Learning from Examples Module, version 2 (MLEM2) system are presented. The parallel method induces rules from the original data set with missing attribute values considered to be lost values, attribute-concept values, or "do not care" conditions. Results of experiments on both approaches are included.

Chapter 5 empirically investigates Bayesian Networks (BN) as a sound formalism for different machine learning tasks, starting with pre-processing, followed by learning and finally contributing to post processing. This investigation includes main ideas of three important algorithms: the Naive Bayes, PC and K2 for learning BNs. In particular, the **K2**$\chi 2$ classifier that combines the K2 algorithm with the $\chi 2$ statistic test is discussed in-depth. The Efficacy of **K2**$\chi 2$ algorithm is illustrated by experiments on some datasets from UCI ML Repository.

Chapter 6 provides an overview of methods, algorithms and applications of evolving intelligent (fuzzy) systems with a particular emphasis on on-line mining of non-stationary data streams. Two models are presented: a fuzzy rule based model that incorporates a recursive clustering method and a fuzzy linear regression tree model whose topology can be continuously updated using a statistical test. The two models are evaluated in the context of time series forecasting problems.

Chapter 7 presents the state-of-the-art in relation to the pattern recognition of Stochastically Episodic (SE) events, a challenging subset of the One Class (OC) classification domain of problems, involving interwoven classes and data that is temporal in nature.

Chapter 8 introduces a novel approach to machine learning in a multi-agent system (MAS). Learning is performed both at a local level (by each agent) and at a global level (amongst agents) during reasoning. A distributed version of Inductive Logic Programming is used, which allows agents to construct new transformation rules based on knowledge and examples. Algorithms that utilize the transformation rules have been presented

Chapter 9 presents a survey of the foundations of rough non-deterministic information analysis (RNIA) for handling both complete and incomplete information. Specifically, the RNIA framework is discussed in the context of machine learning in terms of handling inexact data, question-answering and rule generation. Discussion of algorithms, tools as well as examples that aid in understanding the RNIA framework is given.

Chapter 10 presents a general scheme of interaction and the role of interactions in the modeling of perception processes. Among them are models of objects involved in perception, known as granules and interactions between them determining interactive computations on granules, issues of representation of interactions as well as (adaptive) approximate reasoning rules about interactions and interactive computations are discussed. In particular, the use of information systems for representation of actions or plans is illustrated.

Chapter 11 studies a challenging topic in machine learning, namely learning from imbalanced data where one class is under-represented in comparison with the remaining classes. The focus of this chapter is on experimentally examining the factors that are most critical for the performance of such classifiers. In particular, focused re-sampling methods, which modify the class distribution taking into account local characteristics of examples, are investigated. The experiments included preparation of a new collection of artificial data sets.

Chapter 12 examines a granular computing paradigm for concept learning. Two types of granulation, namely, partitions and coverings are discussed. Based on the rough set theory and other concept learning algorithms, two strategies for classification are investigated. The notion of a relative attribute value reduct is formally expressed, which is complementary to the widely used notion of a relative attribute reduct.

Chapter 13 presents an algorithm for finding calendar-based periodic pattern in a time-stamped data and introduces the concept of certainty factor in association with an overlapped interval. In particular, a hash based data structure for storing and managing full as well as partial periodic calendar-based patterns is used. Experimental results are provided on both real and synthetic databases.

Chapter 14 presents a study of the Mamdani methodology assumptions in the context of fuzzy expert systems to support the approximation of patients' survival length. In particular, a pattern of the s-functions is used to model constrains of fuzzy sets without their predetermined start points and endpoints. The objective of the chapter is the utilization of the Mamdani fuzzy control actions as a methodology adapted for the purpose of making the survival prognoses. The effectiveness of the Mamdani methodology is supported by a surgeon's experience.

Chapter 15 presents case-studies representing the state-of-the-art in applications of SVM in biomedical and biometric applications. These case studies include proteomic spectra analysis to find diagnostic markers. Selected cases of applications of SVM in face recognition problems are presented.

Chapter 16 includes a study of multimedia workload models and explore their applicability to surveillance systems. The workload models are especially helpful in aiding designers to determine resource requirements in a surveillance system. In particular, a novel Markov chain based formal model of multimedia workload for surveillance systems is proposed. The model is validated with real surveillance data.

Chapter 17 presents an application of Dominance-Based Rough Set Approach and Artificial Neural Networks to Computational Stylistics. Performance of both constructed classifiers for authorship attribution is given. Works by Edith Wharton and Jane Austen are used as samples for analysis.

Chapter 18 presents an application of J48 and J48 with reduced error pruning decision trees to image spam classification. In particular, image features that are effective and transcending the evolution of image spam generation. Finally, a validation by feature analysis on thirteen months of image corpus is performed.

1.3 Concluding Remarks

Machine learning paradigms are an important category of tools available to intelligent systems engineering. They can beneficially be incorporated as components of complex multi-mode intelligent systems, or may be used alone as the sole intelligent component in a system. Each paradigm has its own strengths and requirements, and is particularly suited to an individual class of problem. Each chapter of the book explores a different machine learning techniques, illustrating its use and applicability.

Acknowledgements. The first author wishes to acknowledge funding from the Natural Science and Engineering Research Council of Canada (NSERC).

References and Further Readings

1. Angelov, P., Filev, D.P., Kasabov, N.K.: Evolving Intelligent Systems: Methodology and Applications. Wiley and IEEE Series of Computational Intelligence (2010)
2. Barto, A.G., Sutton, R.S., Brouwer, P.S.: Associative Search Network: A reinforcement learning associative memory. Biological Cybernetics 40, 201–211 (1981)
3. Bayes, T., Price, M.: An Essay towards solving a Problem in the Doctrine of Chances. Philosophical Transactions of the Royal Society of London 53, 370–418 (1763)
4. Bishop, C.M.: Pattern Recognition and Machine Learning. Springer (2006) ISBN 0-387-31073-8
5. Bosen, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: Proceedings of the 5th Annual ACM Workshop on Computational Learning Theory, pp. 144–152. ACM Press, Pittsburgh (1992)
6. Concise Oxford Dictionary, 12th edn. Oxford University Press (2011), 978-0-19-960108-0
7. Cortes, C., Vapnik, V.: Support-Vector Networks. Machine Learning 20 (1995)
8. Cristianini, N., Campbell, C., Burges, C.: Kernel Methods: Current Research and Future Directions. Machine Learning 46, 5–9 (2002)
9. Datta, R., Joshi, D., Li, J., Wang, J.Z.: Image Retrieval: Ideas, Influences, and Trends of the New Age. ACM Computing Surveys 40(2), 5:3–5:60 (2008)
10. Fahle, M., Poggio, T.: Perceptual Learning. MIT Press (2002)
11. Fechner, G.: Elemente der Psychophysik, vol. 2. E.J. Bonset, Amsterdam (1860)
12. Farley, B.G., Clark, W.: Simulation of self-organzing systems by digital computer. I.R.E. Transactions on Information Theory 4, 76–84 (1954)
13. Feigenbaum, E.A.: The Simulation of Verbal Learning Behavior. In: Proceedings of the Western Joint Computer Conference, vol. 19, pp. 121–132 (1961)
14. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning, 2nd edn. Springer (2008) ISBN 0-387-95284-5
15. Holland, J.: Adaptation in Natural and Artificial Systems. The Universityof Michigan Press, Ann Arbor (1975)
16. Leibniz, G.W.: New Essays in Human Understanding. Cambridge University Press, Cambridge (1705)
17. McCulloch, W.S., Pitts, W.H.: Logical Calculus of the Ideas Immanent in Nervous Activity. Bulletin of Mathematical Biophysics 5, 115–133 (1943)

18. Mendel, J.M.: A survey of learning control systems. ISA Transactions 5, 297–303 (1966)
19. Mitchell, T.: Machine Learning. McGraw Hill (1997) ISBN 0-07-042807-7
20. Minsky, M.L.: Theory of neural-analog reinforcement learning systems and its applicationto the brain-model problem. Ph.D Dissertation, Princeton University (1954)
21. Nilsson, N.J.: Learning Machines. McGraw-Hill (1965)
22. Nilsson, N.J.: Introduction to Machine Learning, Notes. Stanford University (2010)
23. Pawlak, Z.: Rough Sets. International Journal of Computer and Information Sciences 11, 341–356 (1982)
24. Pawlak, Z., Skowron, A.: Rough sets: Some extensions. Information Sciences 177, 3–27 (2007)
25. Pawlak, Z., Skowron, A.: Rough sets: Some extensions. Information Sciences 177, 28–40 (2007)
26. Pawlak, Z., Skowron, A.: Rough sets: Some extensions. Information Sciences 177, 41–73 (2007)
27. Pedrycz, W., Skowron, A., Kreinovich, V.: Handbook of Granular Computing. Wiley (2008)
28. Peters, J.F., Skowron, A.: Transactions on Rough Sets. Springer (2004)
29. Peters, J.F.: Near sets. Special theory about nearness of objects. Fundamental Informaticae 75(1-4), 407–433 (2007)
30. Peters, J.F.: Near sets. General theory about nearness of objects. Applied Mathematical Sciences 1(53), 2609–2629 (2007)
31. Peters, J.F.: Approximation and Perception in Ethology Based Rein-forcement Learning. In: Pedrycz, W., Skowron, A., Kreinovich, V. (eds.) Handbook of Granular Computing, pp. 670–688. Wiley (2008)
32. Peters, J.F., Henry, C., Ramanna, S.: Reinforcement learning in swarms that learn. In: Proceedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT 2005), France, pp. 400–406 (2005)
33. Rosenblatt, F.: The perceptron: A perceiving and recognizing automaton. Rep. No. 85-460-1, Project PARA. Cornell Aeronautical Laboratory, Buffalo NY (1957)
34. Sutton, R.S.: Temporal Credit Assignment in Reinforcement Learning. Ph.D Dissertation. University of Massachussetts (1984)
35. Vapnik, V.: The nature of statistical learning theory. Springer (1995) ISBN 0-387-98780-0
36. Whiteson, S., Littman, M.L.: Introduction to the special issue on empirical evaluationsin reinforcement learning. Machine Learning 84, 1–6 (2011), doi:10.1007/s10994-011-5255-6
37. Muggleton, S., De Raedt, L., Poole, D., Bratko, I., Flach, P., Inoue, K., Srinivasan, A.: ILP turns 20 Biography and future challenges. Machine Learning (2011), doi:10.1007/s10994-011-5259-2
38. Weber, E.H.: De pulsu, resorptione, auditu et tactu. Anatatione-sanatomicae et physiologicae. Koehler, Leipzig (1834)
39. Zadeh, L.A.: Toward a theory of Fuzzy Information Granulation and Its Certainty in Human Reasoning and Fuzzy Logic. Fuzzy Sets and Systems 90, 111–127 (1997)
40. Zadeh, L.A.: Fuzzy Sets. Information and Control 8(3), 338–353 (1965)

Part A

Foundations

Chapter 2
Extensions of Dynamic Programming as a New Tool for Decision Tree Optimization

Abdulaziz Alkhalid, Igor Chikalov, Shahid Hussain, and Mikhail Moshkov

Abstract. The chapter is devoted to the consideration of two types of decision trees for a given decision table: α-decision trees (the parameter α controls the accuracy of tree) and decision trees (which allow arbitrary level of accuracy). We study possibilities of sequential optimization of α-decision trees relative to different cost functions such as depth, average depth, and number of nodes. For decision trees, we analyze relationships between depth and number of misclassifications. We also discuss results of computer experiments with some datasets from UCI ML Repository.

2.1 Introduction

Decision trees are widely used as predictors [8], as a way of knowledge representation [3] and as algorithms for problem solving [10]. To have more understandable decision trees we need to minimize the number of nodes in a tree. To have faster decision trees we need to minimize the depth or average depth of a tree. In many cases, we need to minimize the number of misclassifications for a tree under some restrictions on time or space complexity of the tree. If we would like to minimize the number of decision rules derived from the tree, we need to minimize the number of terminal nodes in the tree. Unfortunately, almost all problems connected with decision trees optimization are NP-hard.

Our approach to optimization of decision trees is based on an extension of dynamic programming which allows sequential optimization relative to different criteria where we can use subtables of the initial decision table as subproblems. Unfortunately, in the general decision tables could have very large number of subtables so the considered approach is applicable directly only to relatively small decision

Abdulaziz Alkhalid · Igor Chikalov · Shahid Hussain · Mikhail Moshkov
Mathematical and Computer Sciences and Engineering Division,
King Abdullah University of Science and Technology, Thuwal 23955-6900, Saudi Arabia
e-mail: {abdulaziz.alkhalid,igor.chikalov,shahid.hussain,
mikhail.moshkov}@kaust.edu.sa

tables. However, the situation can be improved if we study approximate decision trees instead of exact decision trees. We consider the number of unordered pairs of rows with different decisions $R(T)$ as uncertainty of a decision table T.

We study so called α-decision trees, $\alpha \geq 0$, that do not solve the problem of recognition of decision attached to a given row exactly but localize the given row in a subtable with uncertainty at most α. In this case it is not necessary to work with the whole set of subtables, but we finish the partitioning of a subtable if its uncertainty is at most α. The parameter α controls the computational complexity and makes the algorithm applicable for relatively large data sets.

We describe the whole set of α-decision trees for a given decision table by a directed acyclic graph $\Delta_\alpha(T)$ with subtables as nodes. We can apply to this graph a procedure of optimization relative to (for example) the number of nodes in a decision tree, and obtain a subgraph of the initial graph which represents all α-decision trees with minimum number of nodes. We can apply to this subgraph a procedure of optimization relative (for example) to the average depth, etc.

The second line of research considered in this chapter is the study of relationships between depth and number of misclassifications for decision trees. In this case, we need to avoid the consideration of restrictions on the accuracy of trees. So, instead of α-decision trees, we will study arbitrary decision trees which can finish the partition of a subtable in any time. For such trees, we can resolve problems of the following type: for a given p we need to find the minimum number of misclassifications for a decision tree among all decision trees which depth is at most p.

The chapter also contains results of computer experiments with some datasets (decision tables) from UCI Machine Learning Repository [6].

We propose algorithms for decision tree optimization based on dynamic programming. The idea is close to algorithms described in [7, 9], but authors devised it independently and made several extensions: the algorithms are capable of founding a set of optimal trees, perform sequential optimization by different criteria, and find relationships between two criteria [1, 2, 4, 5, 11].

The rest of the chapter is organized as follows. In Section 2, basic notions are discussed. In Section 3, representation of sets of decision trees and α-decision trees by directed acyclic graphs is considered. Section 4 gives a way for sequential optimization of decision trees relative to different cost functions. Section 5 describes a procedure of computing relationships between decision tree depth and number of misclassifications. Section 6 concludes the paper followed by references and an appendix describing “transformation of functions.”

2.2 Basic Notions

In the following, we consider notions of decision table, decision and α-decision tree for table, and the notion of cost function for decision trees.

2.2.1 Decision Tables and Trees

In this chapter, we consider only decision tables with discrete attributes. These tables contain neither missing values nor equal rows. Consider a *decision table T* depicted in Fig. 1.

f_1	$\dots$	f_m	d
b_{11}	$\dots$	b_{1m}	c_1
	$\dots$		$\dots$
b_{N1}	$\dots$	b_{Nm}	c_N

Fig. 1 Decision table

Here $f_1,\dots,f_m$ are names of columns (conditional attributes); $c_1,\dots,c_N$ are nonnegative integers which can be interpreted as decisions (values of the decision attribute d); b_{ij} are nonnegative integers which are interpreted as values of conditional attributes (we assume that the rows $(b_{11},\dots,b_{1m}),\dots,(b_{N1},\ \dots,b_{Nm})$ are pairwise different). We denote by $E(T)$ the set of attributes (columns of the table T), each of which contains different values. For $f_i \in E(T)$ let $E(T,f_i)$ be the set of values from the column f_i.

Let $f_{i_1},\dots,f_{i_t} \in \{f_1,\ \dots,f_m\}$ and $a_1,\dots,a_t$ be nonnegative integers. We denote by $T(f_{i_1},a_1)\dots(f_{i_t},a_t)$ the subtable of the table T, which consists of such and only such rows of T that at the intersection with columns $f_{i_1},\dots,f_{i_t}$ have numbers $a_1,\dots,a_t$ respectively. Such nonempty tables (including the table T) will be called *separable subtables* of the table T. For a subtable Θ of the table T we will denote by $R(\Theta)$ the number of unordered pairs of rows that are labeled with different decisions. Later we will interpret the value $R(\Theta)$ as *uncertainty* of the table Θ. A minimum decision value which is attached to the maximum number of rows in a nonempty subtable Θ will be called the *most common decision* for Θ.

A *decision tree Γ over* the table T is a finite directed tree with root in which each terminal node is labeled with a decision. Each nonterminal node is labeled with a conditional attribute, and for each nonterminal node the outgoing edges are labeled with pairwise different nonnegative integers. Let v be an arbitrary node of Γ. We now define a subtable $T(v)$ of the table T. If v is the root then $T(v)=T$. Let v be a node of Γ that is not the root, nodes in the path from the root to v be labeled with attributes $f_{i_1},\dots,f_{i_t}$, and edges in this path be labeled with values $a_1,\dots,a_t$ respectively. Then $T(v)=T(f_{i_1},a_1),\dots,(f_{i_t},a_t)$.

Let Γ be a decision tree over T. We will say that Γ is a *decision tree for T* if any node v of Γ satisfies the following conditions:

- If $R(T(v))=0$ then v is a terminal node labeled with the common decision for $T(v)$.
- Otherwise, either v is a terminal node labeled with the most common decision for $T(v)$, or v is labeled with an attribute $f_i \in E(T(v))$ and, if $E(T(v),f_i)=\{a_1,\dots,a_t\}$, then t edges leave node v, and these edges are labeled with $a_1,\dots,a_t$ respectively.

Let α be a nonnegative real number such that $0 \le \alpha \le 1$. We will say that Γ is an *α-decision tree for T* if any node v of Γ satisfies the following conditions:

- If $R(T(v)) \le \alpha$ then v is a terminal node labeled with the most common decision for $T(v)$.
- Otherwise, v is labeled with an attribute $f_i \in E(T(v))$ and, if $E(T(v), f_i) = \{a_1, \ldots, a_t\}$, then t edges leave node v, and these edges are labeled with $a_1, \ldots, a_t$ respectively.

Let Γ be a decision tree for T or an α-decision tree for T, and r be a row of T. One can show that in Γ there is exactly one terminal node v such that r belongs to $T(v)$. We will say about the number attached to v as about the result of work of Γ on r.

2.2.2 Cost Functions

We will consider cost functions which are given in the following way: values of the considered cost function ψ, which are nonnegative numbers, are defined by induction on pairs (T, Γ), where T is a decision table and Γ is a decision tree for T or an α-decision tree for T. Let Γ be a decision tree represented in Fig. 2. Then $\psi(T, \Gamma) = \psi^0(T, b)$ where ψ^0 is an operator which transforms a decision table T and a nonnegative integer b into a nonnegative number. Let Γ be a decision tree depicted in Fig. 3. Then

$$\psi(T, \Gamma) = F(N(T), \psi(T(f_i, a_1), \Gamma_1), \ldots, \psi(T(f_i, a_t), \Gamma_t)).$$

Fig. 2 Trivial decision tree

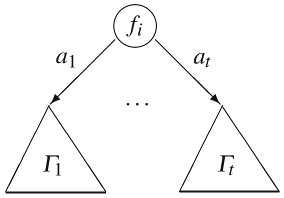


Fig. 3 Aggregated decision tree

Here $N(T)$ is the number of rows in the table T, and $F(n, \psi_1, \psi_2, \ldots)$ is an operator which transforms the considered tuple of nonnegative numbers into a nonnegative number. Note that the number of variables $\psi_1, \psi_2, \ldots$ is not bounded from above.

The considered cost function will be called *monotone* if for any natural t and any nonnegative numbers $a, c_1, \ldots, c_t, d_1, \ldots, d_t$, from inequalities $c_1 \le d_1, \ldots, c_t \le d_t$ the inequality

$$F(a, c_1, \ldots, c_t) \le F(a, d_1, \ldots, d_t)$$

follows.

The considered cost function will be called *strongly monotone* if it is monotone and for any natural t and any nonnegative numbers $a, c_1, \dots, c_t, d_1, \dots, d_t$ from inequalities $a > 0$, $c_1 \le d_1, \dots, c_t \le d_t$ and inequality $c_i < d_i$, which is true for some $i \in \{1, \dots, t\}$, the inequality $F(a, c_1, \dots, c_t) < F(a, d_1, \dots, d_t)$ follows.

Now we take a closer view of some cost functions.

Number of nodes: $\psi(T, \Gamma)$ is the number of nodes in the decision tree Γ. For this cost function $\psi^0(T, b) \equiv 1$ and $F(n, \psi_1, \psi_2, \dots, \psi_t) = 1 + \sum_{i=1}^{t} \psi_i$. This measure is strongly monotone.

Number of terminal nodes: $\psi(T, \Gamma)$ is the number of terminal nodes in the decision tree Γ. For this cost function $\psi^0(T, b) \equiv 1$ and $F(n, \psi_1, \psi_2, \dots, \psi_t) = \sum_{i=1}^{t} \psi_i$. This measure is strongly monotone.

Depth: $\psi(T, \Gamma)$ is the maximal length of a path from the root to a terminal node of Γ. For this cost function $\psi^0(T, b) \equiv 0$ and $F(n, \psi_1, \psi_2, \dots, \psi_t) = 1 + \max\{\psi_1, \dots, \psi_t\}$. This measure is monotone.

Total path length: for an arbitrary row $\bar{\delta}$ of the table T we denote by $l(\bar{\delta})$ the length of path from the root to a terminal node v of Γ such that $\bar{\delta}$ belongs to $T(v)$. Then $\psi(T, \Gamma) = \sum_{\bar{\delta}} l(\bar{\delta})$, where we take the sum on all rows $\bar{\delta}$ of the table T. For this cost function $\psi^0(T, b) \equiv 0$ and $F(n, \psi_1, \psi_2, \dots, \psi_t) = n + \sum_{i=1}^{t} \psi_i$. This measure is strongly monotone.

Note that the *average depth* of Γ is equal to the total path length divided by $N(T)$.

Number of misclassifications: $\psi(T, \Gamma)$ is the number of rows r in T for which the result of the work of decision tree Γ on r does not equal to the decision attached to the row r. For this cost function $\psi^0(T, b)$ is equal to the number of rows in T which are labeled with decisions different from b, and $F(n, \psi_1, \psi_2, \dots, \psi_t) = \sum_{i=1}^{t} \psi_i$. This measure is strongly monotone.

Later in experiments we will fix some decision table T and in notation for cost functions will omit this table. For a decision tree Γ, we denote its average depth by $h_{av}(\Gamma)$, the number of nodes by $L(\Gamma)$, the number of terminal nodes by $L_t(\Gamma)$, the depth by $h(\Gamma)$, and the number of misclassifications by $\mu(\Gamma)$.

2.3 Representation of Sets of α-Decision Trees and Decision Trees

Let α be a nonnegative real number. Consider an algorithm for construction of a graph $\Delta_\alpha(T)$, which can represent the set of all α-decision trees for the table T. Nodes of this graph are some separable subtables of the table T. During each step we process one node and mark it with symbol *. We start with the graph that consists of one node T and finish when all nodes of the graph are processed.

Let the algorithm have already performed p steps. We now describe the step number $(p+1)$. If all nodes are processed then the work of the algorithm is finished, and the resulted graph is $\Delta_\alpha(T)$. Otherwise, choose a node (table) Θ that has not been processed yet. Let b be the most common decision for Θ. If $R(\Theta) \le \alpha$, label the considered node with b, mark it with symbol * and proceed to the step number $(p+2)$. Let $R(\Theta) > \alpha$. For each $f_i \in E(\Theta)$ draw a bundle of edges from the node

Θ (this bundle of edges will be called f_i-*bundle*). Let $E(\Theta, f_i) = \{a_1, \dots, a_t\}$. Then draw t edges from Θ and label these edges with the pairs $(f_i, a_1), \dots, (f_i, a_t)$ respectively. These edges enter into the nodes $\Theta(f_i, a_1), \dots, \Theta(f_i, a_t)$. If some of the nodes $\Theta(f_i, a_1), \dots, \Theta(f_i, a_t)$ do not present in the graph then add these nodes to the graph. Mark the node Θ with symbol * and proceed to the step number $(p+2)$.

Now for each node Θ of the graph $\Delta_\alpha(T)$ we describe the set of α-decision trees corresponding to it. It is clear that $\Delta_\alpha(T)$ is a directed acyclic graph (DAG). A node of such graph will be called *terminal* if there are no edges leaving this node. We will move from terminal nodes, which are labeled with numbers, to the node T. Let Θ be a node, which is labeled with a number b. Then the only trivial α-decision tree depicted in Fig. 2 corresponds to the considered node.

Let Θ be a node (table), for which $R(\Theta) > \alpha$. There is a number of bundles of edges starting in Θ. We consider an arbitrary bundle and describe the set of α-decision trees corresponding to this bundle. Let the considered bundle be an f_i-bundle where $f_i \in E(\Theta)$ and $E(\Theta, f_i) = \{a_1, \dots, a_t\}$. Let $\Gamma_1, \dots, \Gamma_t$ be α-decision trees from the sets corresponding to the nodes $\Theta(f_i, a_1), \dots, \Theta(f_i, a_t)$. Then the α-decision tree depicted in Fig. 3 belongs to the set of α-decision trees, which correspond to this bundle. All such α-decision trees belong to the considered set, and this set does not contain any other α-decision trees. Then the set of α-decision trees corresponding to the node Θ coincides with the union of sets of α-decision trees corresponding to bundles starting in Θ. We denote by $D_\alpha(\Theta)$ the set of α-decision trees corresponding to the node Θ in the graph $\Delta_\alpha(T)$.

The following proposition shows that the graph $\Delta_\alpha(T)$ can represent all α-decision trees for the table T.

Proposition 1. *Let T be a decision table and Θ a node in the graph $\Delta_\alpha(T)$. Then the set $D_\alpha(\Theta)$ coincides with the set of all α-decision trees for the table Θ.*

Proof. We prove this proposition by induction on nodes in the graph $\Delta_\alpha(T)$. For each terminal node Θ, only one α-decision tree exists depicted in Fig. 2, and the set $D_\alpha(\Theta)$ contains only this tree. Let Θ be a nonterminal node and the statement of proposition hold for all its descendants.

Consider an arbitrary decision tree $\Gamma \in D_\alpha(\Theta)$. Obviously, Γ contains more than one node. Let the root of Γ be labeled with the attribute f_i and the edges leaving root be labeled with the numbers $a_1, \dots, a_t$. For $j = 1, \dots, t$, denote by Γ_j the decision tree connected to the root with the edge labeled with the number a_j. From definition of the set $D_\alpha(\Theta)$ it follows that f_i is contained in the set $E(\Theta)$, $E(\Theta, f_i) = \{a_1, \dots, a_t\}$ and for $j = 1, \dots, t$, the decision tree Γ_j belongs to the set $D_\alpha(\Theta(f_i, a_j))$. According to the inductive hypothesis, the tree Γ_j is an α-decision tree for the table $\Theta(f_i, a_j)$. Then the tree Γ is an α-decision tree for the table Θ.

Now we consider an arbitrary α-decision tree Γ for the table Θ. According to the definition, the root of Γ is labeled with an attribute f_i from the set $E(\Theta)$, edges leaving the root are labeled with numbers from the set $E(\Theta, f_i)$ and the subtrees whose roots are nodes, to which these edges enter, are α-decision trees for corresponding descendants of the node Θ. Then, according to the definition of the set $D_\alpha(\Theta)$ and to inductive hypothesis, the tree Γ belongs to the set $D_\alpha(\Theta)$. □

We denote $\Delta(T) = \Delta_0(T)$. Now for each node of the graph $\Delta(T)$ we describe the set of decision trees corresponding to it. We will move from terminal nodes, which are labeled with numbers, to the node T. Let Θ be a node, which is labeled with a number b. Then the only trivial decision tree depicted in Fig. 2 corresponds to the considered node.

Let Θ be a node (table), which is not terminal. There is a number of bundles of edges starting in Θ. We consider an arbitrary bundle and describe the set of decision trees corresponding to this bundle. Let the considered bundle be an f_i-bundle where $f_i \in E(\Theta)$ and $E(\Theta, f_i) = \{a_1, \ldots, a_t\}$. Let $\Gamma_1, \ldots, \Gamma_t$ be decision trees from sets corresponding to the nodes $\Theta(f_i, a_1), \ldots, \Theta(f_i, a_t)$. Then the decision tree depicted in Fig. 3 belongs to the set of decision trees, which correspond to this bundle. All such decision trees belong to the considered set, and this set does not contain any other decision trees. Then the set of decision trees corresponding to the node Θ coincides with the union of sets of decision trees corresponding to bundles starting in Θ and the set containing one decision tree depicted in Fig. 2, where b is the most common decision for Θ. We denote by $D(\Theta)$ the set of decison trees corresponding to the node Θ.

The following proposition shows that the graph $\Delta(T)$ can represent all decision trees for the table T.

Proposition 2. *Let T be a decision table and Θ a node in the graph $\Delta(T)$. Then the set $D(\Theta)$ coincides with the set of all decision trees for the table Θ.*

Proof. We prove the proposition by induction on nodes in the graph $\Delta(T)$. For each terminal node Θ, only one decision tree exists depicted in Fig. 2, and the set $D(\Theta)$ contains only this tree. Let Θ be a nonterminal node and the statement of proposition hold for all its descendants.

Consider an arbitrary decision tree $\Gamma \in D(\Theta)$. If Γ contains exactly one node labeled with the most common decision for Θ, then Γ is a decision tree for Θ. Let Γ contain more than one node, the root of Γ be labeled with an attribute f_i and the edges leaving root be labeled with the numbers $a_1, \ldots, a_t$. For $j = 1, \ldots, t$, denote by Γ_j the decision tree connected to the root with the edge labeled with the number a_j. From definition of the set $D(\Theta)$ it follows that f_i is contained in the set $E(\Theta)$, $E(\Theta, f_i) = \{a_1, \ldots, a_t\}$ and for $j = 1, \ldots, t$, the decision tree Γ_j belongs to the set $D(\Theta(f_i, a_j))$. According to the inductive hypothesis, the tree Γ_j is a decision tree for the table $\Theta(f_i, a_j)$. Then the tree Γ is a decision tree for the table Θ.

Now we consider an arbitrary decision tree Γ for the table Θ. If Γ contains exactly one node labeled with the most common decision for Θ, then $\Gamma \in D(\Theta)$. Otherwise, the root of Γ is labeled with an attribute f_i from the set $E(\Theta)$, edges leaving the root are labeled with numbers from the set $E(\Theta, f_i)$ and the subtrees whose roots are nodes, to which these edges enter, are decision trees for corresponding descendants of the node Θ. Then, according to definition of the set $D(\Theta)$ and to inductive hypothesis, the tree Γ belongs to the set $D(\Theta)$. □

2.4 Optimization of α-Decision Trees

In this section, we introduce the notion of proper subgraph of the graph $\Delta_\alpha(T)$ and give the procedure of finding a set of optimal α-decision trees relative to some functions.

2.4.1 Proper Subgraphs of Graph $\Delta_\alpha(T)$

Let us introduce the notion of proper subgraph of the graph $\Delta_\alpha(T)$. For each node of the graph $\Delta_\alpha(T)$, which is not terminal, we can remove any but not all bundles that leave the node. Denote the obtained subgraph by G. Such subgraphs will be called *proper* subgraphs of the graph $\Delta_\alpha(T)$. It is clear that all terminal nodes of G are terminal nodes of the graph $\Delta_\alpha(T)$. As it was described earlier, we can associate a set of α-decision trees to each node Θ of G. It is clear that all these α-decision trees belong to the set $D_\alpha(\Theta)$. We denote this set of α-decision trees by $D_{\alpha,G}(\Theta)$.

2.4.2 Procedure of Optimization

Let G be a proper subgraph of the graph $\Delta_\alpha(T)$, and ψ be a cost function given by operators ψ^0 and F. Below we describe a procedure, which transforms the graph G into a proper subgraph G_ψ of G. We begin from terminal nodes and move to the node T. We attach a number to each node, and possible remove some bundles of edges, which start in the considered node. Let Θ be a terminal node labeled with a number b. We attach the number $\psi^0(T,b)$ to the node Θ. Consider a node Θ, which is not terminal, and a bundle of edges, which starts in this node. Let edges be labeled with pairs $(f_i,a_1),\ldots,(f_i,a_t)$, and edges enter to nodes $\Theta(f_i,a_1),\ldots,\Theta(f_i,a_t)$, to which numbers $\psi_1,\ldots,\psi_t$ are attached already. Then we attach to the considered bundle the number $F(N(\Theta),\psi_1,\ldots,\psi_t)$.

Among numbers attached to bundles starting in Θ we choose the minimum number p and attach it to the node Θ. We remove all bundles starting in Θ to which numbers are attached that are greater than p. When all nodes will be treated we obtain a graph. Denote this graph by G_ψ. As it was done previously, for any node Θ of G_ψ we denote by $D_{\alpha,G_\psi}(\Theta)$ the set of α-decision trees associated with Θ.

Note that using the graph G_ψ it is easy to find the number of decision trees in the set $D_{\alpha,G_\psi}(\Theta)$: $|D_{\alpha,G_\psi}(\Theta)| = 1$ if Θ is a terminal node. Consider a node Θ, which is not terminal, and a bundle of edges, which start in this node and enter the nodes $\Theta_1 = \Theta(f_i,a_1),\ldots,\Theta_t = \Theta(f_i,a_t)$. We correspond to this bundle the number $|D_{\alpha,G_\psi}(\Theta_1)| \times \cdots \times |D_{\alpha,G_\psi}(\Theta_t)|$. Then $|D_{\alpha,G_\psi}(\Theta)|$ is equal to the sum of numbers corresponding to bundles starting in Θ.

Let T be a decision table and ψ a monotone cost function. Let G be a proper subgraph of $\Delta_\alpha(T)$ and Θ an arbitrary node in G. We will denote by $D^{opt}_{\alpha,\psi,G}(\Theta)$ the subset of $D_{\alpha,G}(\Theta)$ containing all α-decision trees having minimum complexity relative to ψ, i.e., $D^{opt}_{\alpha,\psi,G} = \{\hat{\Gamma} \in D_{\alpha,G}(\Theta) : \psi(\Theta,\hat{\Gamma}) = \min_{\Gamma \in D_{\alpha,G}(\Theta)} \psi(\Theta,\Gamma)\}$.

We present two theorems that describe important properties of the set $D_{\alpha,G_\psi}(\Theta)$ for the cases of monotone and strongly monotone cost function ψ.

Lemma 1. *Let T be a decision table and ψ be a monotone cost function defined by the pair of operators (ψ^0, F). Let G be a proper subgraph of $\Delta_\alpha(T)$, Θ be an arbitrary node in the graph G and p be a number assigned to the node Θ by optimization procedure. Then for each α-decision tree Γ from the set $D_{\alpha,G_\psi}(\Theta)$, the equality $\psi(\Theta,\Gamma) = p$ holds.*

Proof. We prove the lemma by induction on nodes in the graph G. For each terminal node Θ, only one α-decision tree Γ exists depicted in Fig. 2 and the statement of lemma obviously holds for Θ. Let now Θ be a nonterminal node, and the statement of lemma holds for all descendants of Θ. Consider an arbitrary α-decision tree $\Gamma \in D_{\alpha,G_\psi}(\Theta)$. Let the root of Γ be labeled with the attribute f_i and the edges leaving root be labeled with the numbers $a_1,\dots,a_t$. For $j = 1,\dots,t$ denote by Γ_j the decision tree connected to the root with edge labeled with the number a_j. Let for $j = 1,\dots,t$ the node $\Theta(f_i,a_j)$ be labeled with the number p_j. According to the inductive hypothesis, the equality $\psi(\Theta(f_i,a_j),\Gamma_j) = p_j$ holds for $j = 1,\dots,t$. According to the optimization procedure, $p = F(N(\Theta), p_1,\dots,p_t)$. From the definition of cost function it follows that $\psi(\Theta,\Gamma) = F(N(\Theta), \psi(\Theta(f_i,a_1),\Gamma_1),\dots,\psi(\Theta(f_i,a_t),\Gamma_t))$. So $\psi(\Theta,\Gamma) = p$. Since Γ is an arbitrary tree from $D_{\alpha,G_\psi}(\Theta)$, all the trees in $D_{\alpha,G_\psi}(\Theta)$ have the same cost p. □

Theorem 1. *Let T be a decision table and ψ a monotone cost function. Let G be a proper subgraph of $\Delta_\alpha(T)$ and Θ an arbitrary node in the graph G. Then $D_{\alpha,G_\psi}(\Theta) \subseteq D^{opt}_{\alpha,\psi,G}(\Theta)$.*

Proof. The proof is by induction on nodes of the graph G. Let Θ be a terminal node, and b be the most common decision for Θ. Then the set $D_{\alpha,G_\psi}(\Theta)$ contains only tree depicted in Fig. 2 and this tree, obviously, belongs to $D^{opt}_{\alpha,\psi,G}(\Theta)$. So the statement of theorem holds for the node Θ.

Let now Θ be a nonterminal node in G and the statement of theorem hold for any descendant of Θ in the graph G. Let the number p be assigned to the node Θ by optimization procedure. Lemma 1 implies that all α-decision trees in $D_{\alpha,G_\psi}(\Theta)$ have the same complexity p. Consider an arbitrary decision tree Γ from the set $D^{opt}_{\alpha,\psi,G}(\Theta)$. From definition of the set $D^{opt}_{\alpha,\psi,G}(\Theta)$ we have $\psi(\Theta,\Gamma) \le p$.

To prove the theorem we need to show that $\psi(\Theta,\Gamma) = p$. Let the root of Γ be assigned with the attribute f_i. Since Γ is an α-decision tree for Θ, f_i is contained in the set $E(\Theta)$. Let $E(\Theta, f_i) = \{a_1,\dots,a_t\}$. Then t edges leave the root labeled with numbers $a_1,\dots,a_t$. For $j = 1,\dots,t$, denote by Γ_j the subtree that is connected to the root with the edge labeled with a_j. One can show that Γ_j is contained in the set $D_{\alpha,G}(\Theta(f_i,a_j))$. Let p_j be a number assigned to the node $\Theta(f_i,a_j)$ during optimization. Since the theorem holds for the node $\Theta(f_i,a_j)$, we have $\psi(\Theta(f_i,a_j),\Gamma_j) \ge p_j$. From the description of optimization procedure it follows that $F(N(\Theta), p_1,\dots,p_t) \ge p$. Since ψ is monotone cost function, we have

$$\begin{aligned}\psi(\Theta,\Gamma) &= F(N(\Theta),\psi(\Theta(f_i,a_1),\Gamma_1),\ldots,\psi(\Theta(f_i,a_t),\Gamma_t))\\ &\geq F(N(\Theta),p_1,\ldots,p_t).\end{aligned}$$

From the two last inequalities and the inequality $\psi(\Theta,\Gamma) \leq p$ we have $\psi(\Theta,\Gamma) = p$. □

Theorem 2. *Let T be a decision table and ψ a strongly monotone cost function. Let G be a proper subgraph of $\Delta_\alpha(T)$ and Θ be an arbitrary node in the graph G. Then $D_{\alpha,G_\psi}(\Theta) = D^{opt}_{\alpha,\psi,G}(\Theta)$.*

Proof. Since ψ is strongly monotone, ψ is monotone. By Theorem 1, $D_{\alpha,G_\psi}(\Theta) \subseteq D^{opt}_{\alpha,\psi,G}(\Theta)$. Prove that for an arbitrary tree $\Gamma \in D^{opt}_{\alpha,\psi,G}(\Theta)$, the tree Γ belongs to the set $D_{\alpha,G_\psi}(\Theta)$. The induction on the nodes of G will be used. If Θ is a terminal node then only one α-decision tree exists for Θ and thus the statement of theorem holds. Let Θ be a nonterminal node and the statement of theorem hold for all descendants of Θ. Let the root of the tree Γ is assigned by the attribute f_i. Since Γ is an α-decision tree for Θ, f_i is contained in the set $E(\Theta)$. Let $E(\Theta, f_i) = \{a_1,\ldots,a_t\}$. Then t edges leaving the root are labeled with the numbers $a_1,\ldots,a_t$. For $j = 1,\ldots,t$, denote by Γ_j the subtree that is connected to the root with the edge labeled with a_j. Since ψ is a strongly monotone cost function, the tree Γ_j belongs to the set $D^{opt}_{\alpha,\psi,G}(\Theta(f_i,a_j))$. Since the statement of theorem holds for the node $\Theta(f_i,a_j)$, the tree Γ_j belongs to the set $D_{\alpha,G_\psi}(\Theta(f_i,a_j))$ for $j = 1,\ldots,t$. Consider the bundle of edges in the graph $\Delta_\alpha(T)$ that leave the node Θ and are labeled with the pairs $(f_i,a_1),\ldots,(f_i,a_t)$. Since $\Gamma \in D^{opt}_{\alpha,\psi,G}(\Theta)$ these edges were not removed by the optimization procedure. Then according to the definition of the set $D_{\alpha,G_\psi}(\Theta)$ the tree Γ belongs to this set. □

2.4.3 Possibilities of Sequential Optimization

Let the graph $\Delta_\alpha(T)$ be constructed for a decision table T. Let ψ_1, ψ_2 be strongly monotone cost functions. Apply the procedure of optimization to the graph $\Delta_\alpha(T)$ and to the cost function ψ_1. As a result we obtain a proper subgraph $(\Delta_\alpha(T))_{\psi_1}$ of the graph $\Delta_\alpha(T)$. Denote this subgraph by G_1. According to Proposition 1 and Theorem 2, the set of α-decision trees corresponding to the node T of this graph coincides with the set of all α-decision trees for the table T, which have minimum cost relative to ψ_1. Denote this set by D_1.

Apply the procedure of optimization to the graph G_1 and the cost function ψ_2. As a result we obtain the proper subgraph $(G_1)_{\psi_2}$ of the graph $\Delta_\alpha(T)$. Denote this subgraph by G_2. The set of α-decision trees corresponding to the node T of this graph coincides with the set of all α-decision trees from D_1 which have minimum cost relative to ψ_2. It is possible to continue this process of consecutive optimization relative to various criterions.

If ψ_2 is a monotone cost function then according to Theorem 1 the set of α-decision trees, corresponding to the node T of the graph G_2, is a subset of the set of all α-decision trees from D_1 which have minimum complexity relative to ψ_2.

2.4.4 Experimental Results

We made experiments with two datasets from UCI ML Repository [6]: MUSHROOM (8124 rows and 22 conditional attributes) and CARS (1728 rows and 6 conditional attributes). The dataset MUSHROOM contains 2480 missed values. We replaced each missing value with the most common value in the corresponding column. We chose relatively small datasets since we need to begin our consideration from $\alpha = 0$. In fact, instead of α, we use the parameter σ, $0 \leq \sigma < 1$, such that $\alpha = \sigma R(T)$ where $R(T)$ is the uncertainty of the considered dataset (decision table) T.

For each decision table (dataset), we made two groups of experiments. In the first group, we consider five values of σ: 0, 0.001, 0.01, 0.1 and 0.2 and four cost functions: average depth h_{av}, number of nodes L, number of terminal nodes L_t, and depth h. Each experiment contains five steps. During the step number 0 we perform initial optimization relative to each cost function independently. During the steps numbers 1, 2, 3 and 4 we perform sequential optimization relative to h_{av}, L, L_t, and h respectively. Tables 1 and 2 show the results of experiments with datasets MUSHROOM and CARS.

Table 1 Results of experiments on the decision table MUSHROOM with different values of σ ($\alpha = \sigma R(\text{MUSHROOM})$) and the order h_{av}, L, L_t, h of cost functions

optimization type	σ	h_{av}	L	L_t	h
initial	0	1.52388	21	14	3
sequential	0	1.52388	27	22	4
initial	0.001	1.51108	16	11	2
sequential	0.001	1.51108	20	17	3
initial	0.01	1.33161	11	7	2
sequential	0.01	1.33161	17	14	2
initial	0.1	1	5	3	1
sequential	0.1	1	10	9	1
initial	0.2	1	5	3	1
sequential	0.2	1	5	4	1

Table 2 Results of experiments on the decision table CARS with different values of σ ($\alpha = \sigma R(\text{CARS})$) and the order h_{av}, L, L_t, h of cost functions

optimization type	σ	h_{av}	L	L_t	h
initial	0	2.94734	396	290	6
sequential	0	2.94734	398	290	6
initial	0.001	2.19444	33	24	3
sequential	0.001	2.19444	35	25	4
initial	0.01	1.66667	12	9	2
sequential	0.01	1.66667	12	9	2
initial	0.1	1	5	4	1
sequential	0.1	1	5	4	1
initial	0.2	1	4	3	1
sequential	0.2	1	4	3	1

These tables contain the following parameters:

- σ: the parameter which defines α as $\sigma R(T)$;
- h_{av}: either the minimum average depth $h_{av}^{\sigma}(T)$ of an α-decision tree for T (initial) or the average depth of each decision tree obtained after the last step of sequential optimization (sequential), where $\alpha = \sigma R(T)$;
- L: either the minimum number of nodes $L^{\sigma}(T)$ of an α-decision tree for T (initial) or the number of nodes of each decision tree obtained after the last step of sequential optimization (sequential), where $\alpha = \sigma R(T)$;
- L_t: either the minimum number of terminal nodes $L_t^{\sigma}(T)$ of an α-decision tree for T (initial) or the number of terminal nodes of each decision tree obtained after the last step of sequential optimization (sequential), where $\alpha = \sigma R(T)$;
- h: either the minimum depth $h^{\sigma}(T)$ of an α-decision tree for T (initial) or the depth of each decision tree obtained after the last step of sequential optimization (sequential), where $\alpha = \sigma R(T)$.

Table 3 Characteristics of DAG $\Delta_{\alpha}(T)$, $\alpha = \sigma R(T)$, for the decision tables MUSHROOM and CARS

σ	MUSHROOM			CARS		
	# nodes	# edges	# trees	# nodes	# edges	# trees
0	149979	2145617	$> 10^{38}$	7007	19886	$> 10^{38}$
0.001	59802	619898	$> 10^{38}$	1727	3420	$5.26 \cdot 10^{13}$
0.01	24648	161983	$1.96 \cdot 10^{16}$	473	688	$4.99 \cdot 10^{04}$
0.1	4148	16178	750874	118	129	78
0.2	1692	5302	12331	22	21	6

Let σ be real number such that $0 \leq \sigma < 1$, T be a decision table, $\alpha = \sigma R(T)$, and Γ be an α-decision tree for T. We will say that Γ is *totally optimal for T and σ relative to the cost functions h_{av}, h, L,* and L_t if $h_{av}(\Gamma) = h_{av}^{\sigma}(T)$, $h(\Gamma) = h^{\sigma}(T)$, $L(T) = L^{\sigma}(T)$, and $L_t(\Gamma) = L_t^{\sigma}(T)$.

Table 1 shows that for any $\sigma \in \{0, 0.001, 0.01, 0.1, 0.2\}$ there is no decision tree which is totally optimal for MUSHROOM and σ relative to h_{av}, h, L, and L_t.

Table 2 shows that for any $\sigma \in \{0.01, 0.1, 0.2\}$ there exists a decision tree which is totally optimal for CARS and σ relative to h_{av}, h, L, and L_t. For $\sigma = 0$ and $\sigma = 0.001$, there are no such trees.

We found also values of the following parameters related to the directed acyclic graph $\Delta_{\alpha}(T)$:

- # nodes: the number of nodes in the directed acyclic graph $\Delta_{\alpha}(T)$;
- # edges: the number of edges in the directed acyclic graph $\Delta_{\alpha}(T)$;
- # trees: the number of α-decision trees for the table T.

Table 4 Results of sequential optimization on the decision table MUSHROOM with different orders of the cost functions

order of cost functions	h	h_{av}	L	L_t
h, h_{av}, L, L_t	3	1.65387	30	24
h, h_{av}, L_t, L	3	1.65387	30	24
h, L, h_{av}, L_t	3	2.94978	25	17
h, L, L_t, h_{av}	3	2.94978	25	17
h, L_t, h_{av}, L	3	2.94978	25	17
h, L_t, L, h_{av}	3	2.94978	25	17
h_{av}, h, L, L_t	4	1.52388	27	22
h_{av}, h, L_t, L	4	1.52388	27	22
h_{av}, L, h, L_t	4	1.52388	27	22
h_{av}, L, L_t, h	4	1.52388	27	22
h_{av}, L_t, h, L	4	1.52388	27	22
h_{av}, L_t, L, h	4	1.52388	27	22
L, h, h_{av}, L_t	4	1.95815	21	15
L, h, L_t, h_{av}	4	1.95815	21	15
L, h_{av}, h, L_t	5	1.72772	21	15
L, h_{av}, L_t, h	5	1.72772	21	15
L, L_t, h_{av}, h	5	1.72772	21	15
L, L_t, h, h_{av}	4	1.95815	21	15
L_t, h, h_{av}, L	4	3.51059	24	14
L_t, h, L, h_{av}	4	3.51059	24	14
L_t, h_{av}, h, L	5	2.86854	23	14
L_t, h_{av}, L, h	5	2.86854	23	14
L_t, L, h_{av}, h	5	2.86854	23	14
L_t, L, h, h_{av}	5	3.07632	23	14
initial optimization	3	1.52388	21	14

Table 3 shows the values of these parameters. It is possible to see that the values of # nodes, # edges and # trees for $\Delta_\alpha(T)$ decrease with the growth of $\alpha = \sigma R(T)$ (really, with growth of σ). It means that the parameter α can be used for managing algorithm complexity.

Table 3 shows also that the structure of graph $\Delta_\alpha(T)$ is usually far from a tree: the number of edges is essentially larger than the number of nodes.

In the second group of experiments, for each decision table and for $\sigma = 0$ we consider all possible orders of cost functions h, h_{av}, L and L_t. We found values of these functions for decision trees obtained after the last step of sequential optimization. Tables 4 and 5 show the results. The last row in each of these tables contains values of $h^0(T)$, $h^0_{av}(T)$, $L^0(T)$ and $L^0_t(T)$. For the decision table CARS, the obtained results depend only on the order of two functions: h_{av} and L. For the decision table MUSHROOM, the results depend on the order of all four cost functions.

Table 5 Results of sequential optimization on the decision table CARS with different orders of the cost functions

order of cost functions	h	h_{av}	L	L_t
h, h_{av}, L, L_t	6	2.94734	398	290
h, h_{av}, L_t, L	6	2.94734	398	290
h, L, h_{av}, L_t	6	2.9485	396	290
h, L, L_t, h_{av}	6	2.9485	396	290
h, L_t, h_{av}, L	6	2.94734	398	290
h, L_t, L, h_{av}	6	2.9485	396	290
h_{av}, h, L, L_t	6	2.94734	398	290
h_{av}, h, L_t, L	6	2.94734	398	290
h_{av}, L, h, L_t	6	2.94734	398	290
h_{av}, L, L_t, h	6	2.94734	398	290
h_{av}, L_t, h, L	6	2.94734	398	290
h_{av}, L_t, L, h	6	2.94734	398	290
L, h, h_{av}, L_t	6	2.9485	396	290
L, h, L_t, h_{av}	6	2.9485	396	290
L, h_{av}, h, L_t	6	2.9485	396	290
L, h_{av}, L_t, h	6	2.9485	396	290
L, L_t, h_{av}, h	6	2.9485	396	290
L, L_t, h, h_{av}	6	2.9485	396	290
L_t, h, h_{av}, L	6	2.94734	398	290
L_t, h, L, h_{av}	6	2.9485	396	290
L_t, h_{av}, h, L	6	2.94734	398	290
L_t, h_{av}, L, h	6	2.94734	398	290
L_t, L, h_{av}, h	6	2.9485	396	290
L_t, L, h, h_{av}	6	2.9485	396	290
initial optimization	6	2.94734	396	290

2.5 Relationships between Depth and Number of Misclassifications

In the following, we consider tools for the analysis of relationships between depth and number of misclassifications for decision trees for a given decision table.

2.5.1 Computing the Relationships

Let T be a decision table with N rows and m columns labeled with attributes $f_1,\dots,f_m$, and $D(T)$ be the set of all decision trees for T. For a decision tree $\Gamma \in D(T)$, we denote by $h(\Gamma)$ the depth of Γ and by $\mu(\Gamma)$ we denote the number of misclassifications for Γ for the table T. It is clear that the minimum values of h and μ on $D(T)$ are equal to zero, an upper bound on the value of h on $D(T)$ is m, and an upper bound on the value of μ on $D(T)$ is N. We denote $B_h = \{0,1,\dots,m\}$ and

$B_\mu = \{0, 1, \ldots, N\}$. We now define two functions $\mathscr{G}_T : B_h \to B_\mu$ and $\mathscr{F}_T : B_\mu \to B_h$ as follows:

$$\mathscr{G}_T(n) = \min\{\mu(\Gamma) : \Gamma \in D(T), h(\Gamma) \le n\}$$

for any $n \in B_h$, and

$$\mathscr{F}_T(n) = \min\{h(\Gamma) : \Gamma \in D(T), \mu(\Gamma) \le n\}$$

for any $n \in B_\mu$.

The function $\mathscr{G}_T$ can be represented by the tuple $(\mathscr{G}_T(0), \ldots, \mathscr{G}_T(m))$ of its values. The function $\mathscr{F}_T$ can also be represented similarly.

We now describe an algorithm which allows us to construct the function $\mathscr{G}_\Theta$ for any node (subtable) Θ from the graph $\Delta(T)$.

We begin from the terminal nodes and move to the node T. We attach a function $\mathscr{G}_\Theta$ to each node Θ of $\Delta(T) = \Delta_0(T)$.

Let Θ be a terminal node. It means that all rows of Θ are labeled with the same decision b and the decision tree Γ_b as depicted in Fig. 2 belongs to $D(\Theta)$. It is clear that $h(\Gamma_b) = 0$ and $\mu(\Gamma_b) = 0$ for the table Θ. Therefore $\mathscr{G}_\Theta(n) = 0$ for any $n \in B_h$.

Consider a nonterminal node Θ and a bundle of edges labeled with pairs $(f_i, a_1), \ldots, (f_i, a_t)$ that starts from this node. Let these edges enter into nodes $\Theta(f_i, a_1), \ldots, \Theta(f_i, a_t)$, respectively, to which functions

$$\mathscr{G}_{\Theta(f_i,a_1)}, \ldots, \mathscr{G}_{\Theta(f_i,a_t)}$$

are already attached.

We correspond to this bundle (the f_i-bundle) the function $\mathscr{G}^{f_i}_\Theta$, which for any $n \in B_h \setminus \{0\}$ is defined as follows:

$$\mathscr{G}^{f_i}_\Theta(n) = \min\{\mu(\Gamma) : \Gamma \in D(\Theta, f_i), h(\Gamma) \le n\},$$

where $D(\Theta, f_i)$ is the set of decision trees for Θ corresponding to the considered bundle. The set $D(\Theta, f_i)$ contains only such trees from $D(\Theta)$ in which the root is labeled with f_i. We can show that for any $n \in B_h \setminus \{0\}$,

$$\mathscr{G}^{f_i}_\Theta(n) = \mathscr{G}_{\Theta(f_i,a_1)}(n-1) + \cdots + \mathscr{G}_{\Theta(f_i,a_t)}(n-1).$$

It is not difficult to prove that for any $n \in B_h \setminus \{0\}$,

$$\mathscr{G}_\Theta(n) = \min\{\mathscr{G}^{f_i}_\Theta(n) : f_i \in E(\Theta)\}.$$

We know that there is only one decision tree with depth zero in $D(\Theta)$. This is the tree Γ_b as depicted in Fig. 2, where b is the most common decision for Θ. For this tree, we have $h(\Gamma_b) = 0$ and $\mu(\Gamma_b)$ for Θ is equal to the number of rows in Θ which are labeled with decisions other than b. So,

$$\mathscr{G}_\Theta(0) = N(\Theta) - N(\Theta, b),$$

where $N(\Theta)$ is the number of rows in Θ and $N(\Theta, b)$ is the number of rows in Θ which are labeled with the decision b.

Now we can use the following proposition to construct the function $\mathscr{F}_T$.

Proposition 3. *For any $n \in B_\mu$, $\mathscr{F}_T(n) = \min\{p \in B_h : \mathscr{G}_T(p) \leq n\}$.*

This statement follows immediately from Propososition 4 from the Appendix.

Note that to find the value $\mathscr{F}_T(n)$ for $n \in B_\mu$ it is enough to make $O(\log|B_h|) = O(\log m)$ operations of comparison.

2.5.2 Experimental Results

We performed several experiments on datasets (decision tables) acquired from UCI ML Repository [6]. Each dataset is represented as a table containing several input columns and an output (decision) column. In tables containing green missing values, we replaced each missing value with the most common value in the corresponding column. In some tables there were rows that contain identical values in all columns, possibly, except the decision column. In this case each group of identical rows was replaced with a single row with common values in all input columns and the most common value in the decision column. In the following, we present the experimental results and show the plots depicting relationships between the number of misclassifications and the depth of decision trees.

Figure 4 contains two plots for the decision table HOUSE-VOTES-84 (16 attributes and 279 rows). The first plot shows the relationship between the number of misclassifications and the depth (the minimum number of misclassifications among decision trees whose depth is at most the given value) and the second one shows the relationship between the depth and the number of misclassifications (the minimum depth among decision trees for which the number of misclassifications is at most the given value).

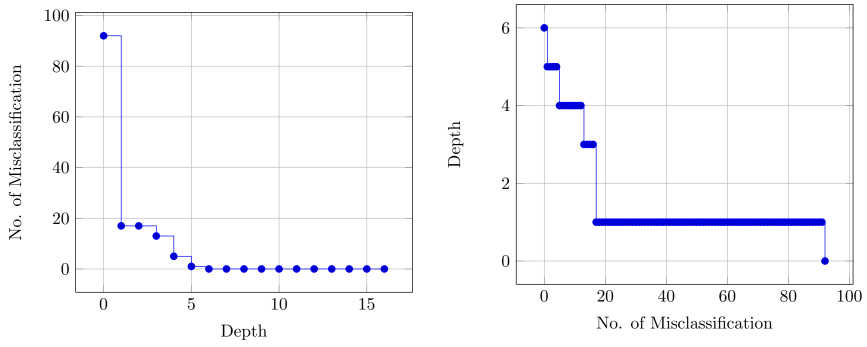


Fig. 4 Relationships between the depth and the number of misclassifications for HOUSE-VOTES-84

Figure 5 contains two plots for the decision table BREAST-CANCER (9 attributes and 266 rows). The first plot shows the relationship between the number of misclassifications and the depth and the second one shows the relationship between the depth and the number of misclassifications.

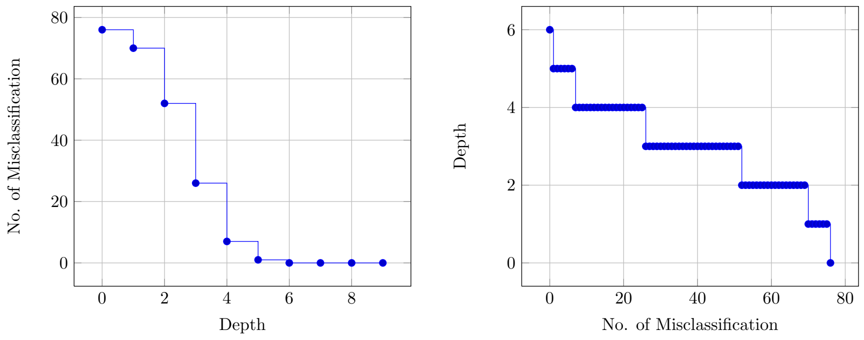


Fig. 5 Relationships between the depth and the number of misclassifications for BREAST-CANCER

Figure 6 contains two plots for the decision table CARS (6 attributes and 1727 rows). The first plot shows the relationship between the number of misclassifications and the depth and the second one shows the relationship between the depth and the number of misclassifications.

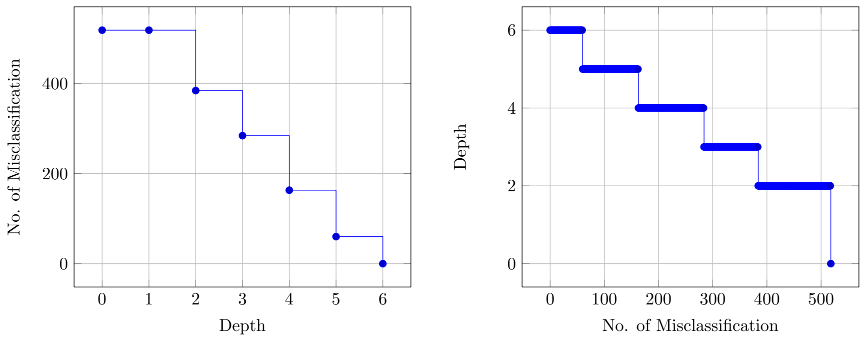


Fig. 6 Relationships between the depth and the number of misclassifications for CARS

2.6 Conclusions

The chapter is devoted to the consideration of an algorithm for sequential optimization of approximate decision trees relative to different cost functions, and to the study of relationships between depth and number of misclassifications of decision trees. Possibilities of the use of proposed techniques are illustrated by experiments with some datasets from UCI ML Repository [6]. Further studies will be connected with the extension of these tool to the decision tables containing continuous attributes.

References

1. Alkhalid, A., Chikalov, I., Moshkov, M.: On Algorithm for Building of Optimal α-Decision Trees. In: Szczuka, M., Kryszkiewicz, M., Ramanna, S., Jensen, R., Hu, Q. (eds.) RSCTC 2010. LNCS, vol. 6086, pp. 438–445. Springer, Heidelberg (2010)
2. Alkhalid, A., Chikalov, I., Moshkov, M.: A Tool for Study of Optimal Decision Trees. In: Yu, J., Greco, S., Lingras, P., Wang, G., Skowron, A. (eds.) RSKT 2010. LNCS, vol. 6401, pp. 353–360. Springer, Heidelberg (2010)
3. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Chapman and Hall, New York (1984)
4. Chikalov, I.V., Moshkov, M.J., Zelentsova, M.S.: On Optimization of Decision Trees. In: Peters, J.F., Skowron, A. (eds.) Transactions on Rough Sets IV. LNCS, vol. 3700, pp. 18–36. Springer, Heidelberg (2005)
5. Chikalov, I., Hussain, S., Moshkov, M.: Relationships Between Depth and Number of Misclassifications for Decision Trees. In: Kuznetsov, S.O., Ślęzak, D., Hepting, D.H., Mirkin, B.G. (eds.) RSFDGrC 2011. LNCS, vol. 6743, pp. 286–292. Springer, Heidelberg (2011)
6. Frank, A., Asuncion, A.: UCI Machine Learning Repository. University of California, Irvine, School of Information and Computer Sciences (2010), `http://archive.ics.uci.edu/ml`
7. Garey, M.R.: Optimal binary identification procedures. SIAM Journal on Applied Mathematics 23, 173–186 (1972)
8. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction. Springer, New York (2001)
9. Martelli, A., Montanari, U.: Optimizing decision trees through heuristically guided search. Commun. ACM 21, 1025–1039 (1978)
10. Moshkov, M.J.: Time Complexity of Decision Trees. In: Peters, J.F., Skowron, A. (eds.) Transactions on Rough Sets III. LNCS, vol. 3400, pp. 244–459. Springer, Heidelberg (2005)
11. Moshkov, M., Chikalov, I.: Consecutive optimization of decision trees concerning various complexity measures. Fundam. Inform. 61, 87–96 (2003)

Appendix. Transformation of Functions

Let f and g be two functions from a set A onto C_f and C_g respectively, where C_f and C_g are finite sets of nonnegative integers. Let $B_f = \{m_f, m_f + 1, \ldots, M_f\}$ and $B_g = \{n_g, n_g + 1, \ldots, N_g\}$ where $m_f = \min\{m : m \in C_f\}$ and $n_g = \min\{n : n \in C_g\}$. Furthermore, M_f and N_g are natural numbers such that $m \le M_f$ and $n \le N_g$ for any $m \in C_f$ and $n \in C_g$, respectively.

We define two functions $\mathscr{F} : B_g \to B_f$ and $\mathscr{G} : B_f \to B_g$ as following:

$$\mathscr{F}(n) = \min\{f(a) : a \in A, g(a) \le n\},\ \ \forall n \in B_g,$$

and

$$\mathscr{G}(m) = \min\{g(a) : a \in A, f(a) \le m\}, \ \forall m \in B_f.$$

It is clear that both $\mathscr{F}$ and $\mathscr{G}$ are nonincreasing functions.

The following proposition states that the functions $\mathscr{F}$ and $\mathscr{G}$ can be used interchangeably and we can evaluate $\mathscr{F}$ using $\mathscr{G}$ and vice versa, i.e., it is enough to know only one function to evaluate the other.

Proposition 4. *For any $n \in B_g$,*

$$\mathscr{F}(n) = \min\{m \in B_f : \mathscr{G}(m) \le n\},$$

and for any $m \in B_f$,

$$\mathscr{G}(m) = \min\{n \in B_g : \mathscr{F}(n) \le m\}.$$

Proof. Let for some $n \in B_g$,

$$\mathscr{F}(n) = m_0. \tag{1}$$

Furthermore, we assume that

$$\min\{m \in B_f : \mathscr{G}(m) \le n\} = t.$$

From (1) it follows that

(i) there exists $b \in A$ such that $g(b) \le n$ and $f(b) = m_0$;
(ii) for any $a \in A$, if $g(a) \le n$ then $f(a) \ge m_0$.

From (i) it follows that $\mathscr{G}(m_0) \le n$. This implies $t \le m_0$. Let us assume that $t < m_0$. In this case, there exits $m_1 < m_0$ for which $\mathscr{G}(m_1) \le n$. Therefore, there exists $a \in A$ such that $f(a) \le m_1$ and $g(a) \le n$, but from (ii) it follows that $f(a) \ge m_0$, which is impossible. So $t = m_0$.

Similarly we can prove the second part of the statement. □

Proposition 4 allows us to transform the function $\mathscr{G}$ given by a tuple

$$(\mathscr{G}(m_f), \mathscr{G}(m_f + 1), \ldots, \mathscr{G}(M_f))$$

into the function $\mathscr{F}$ and vice versa. We know that $\mathscr{G}(m_f) \ge \mathscr{G}(m_f + 1) \ge \cdots \ge \mathscr{G}(M_f)$. To find for a given $n \in B_g$ the minimum $m \in B_f$ such that $\mathscr{G}(m) \le n$ we can use binary search which requires $O(\log|B_f|)$ comparisons. So to find the value $\mathscr{F}(n)$ for $n \in B_g$ it is enough to make $O(\log|B_f|)$ operations of comparison.

Chapter 3
Optimised Information Abstraction in Granular Min/Max Clustering

Andrzej Bargiela and Witold Pedrycz

Abstract. The Min/Max classification and clustering has a distinct advantage of generating easily interpretable information granules - represented as hyperboxes in the multi-dimensional feature space of the data. However, while such an information abstraction lends itself to easy interpretation it leaves open the question whether the granules represent well the original data.

In this chapter we discuss an approach to optimised information abstraction, which retains the advantages of Min/Max clustering while providing a basis for building a more representative set of granules. In particular we extend the information density based granulation by including an extra stage of optimised refinement of granular prototypes. The initial granulation is accomplished by creating hyperboxes in the pattern space through the maximisation of the count of data items per unit volume of hyperboxes. The granulation is totally data driven in that it does not make any assumptions about the number or the maximum size of hyperboxes. Subsequent optimisation involves identification of granular prototypes and their refinement so as to achieve full reconstruction of the original data from the prototypes and the corresponding partition matrix.

3.1 Introductory Comments

Progression from detailed, voluminous numerical information to a more concise representation of knowledge about systems relies to a large extent on an appropriate granulation of information. While it is recognised that the granulation process degrades the accuracy of individual numerical readings the gain of achieving

Andrzej Bargiela
University of Nottingham, Nottingham, UK
e-mail: andrzej.bargiela@nottingham.ac.uk

Witold Pedrycz
University of Alberta, Edmonton, Canada
e-mail: pedrycz@ee.ualberta.ca

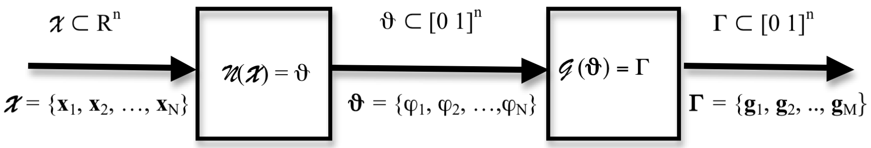
greater generality of models built on such a granulated data is a compelling argument for hierarchical structuring of information. Consequently, we observe that information granules permeate most cognitive activities of humans and help organise knowledge for the purpose of decision-making, control, simulation-modelling, prediction, etc. The mathematical frameworks for the generation of information granules have been integrated under the heading of Granular Computing [1-5, 20, 25-28, 30-32] and include most prominently fuzzy sets [12-13, 19, 25], rough sets [17, 30], probabilistic sets as well as crisp sets (interval analysis) [7, 8, 10, 15, 16]. We recognise however, that regardless of the mathematical framework, information granules are conceptual constructs that have to be mediated by the needs of a specific application. In order to reflect the requirements of the real world, i.e. to be anchored in the experimental evidence, information granules need to be easily interpretable in their specific contexts and should have inherent flexibility that allows them to be optimised in some sense [5].

Interestingly, in spite of the large diversity of approaches to information granulation, the majority of clustering techniques published in the literature produce crisp (not granular) prototypes with the corresponding numeric partition matrices. This is surprising because the very rationale of information granulation hinges on the semantical transformation of data from the precise, numeric domain to the graded, granular domain, so the natural expectation would be to have granular representatives of the granular domain.

Nevertheless, some of the previous research has taken a principled view of identifying information granules that represent the granulated data. The min-max clustering, originally proposed in [23] and subsequently developed in [9] adopt a constructive approach to "growing" granular representatives of data assuming a predefined specificity (size in the normalised feature space) of the information granules. In this study we expand on the above approach and adopt a framework of set theory and interval analysis for the granular pre-processing of data and utilise the FCM algorithm to derive the granular representatives of the original data. The granular representatives are then optimised to ensure full representation of the granulated data originating from the pre-processing with the minimum expansion of the prototypes.

Proceeding with a formal description, let us start with a collection of n-dimensional numeric data (located in $\mathbf{R}^n$), say $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$. Since we are interested in a data-driven discovery of features rather than the frequency of occurrence of a specific feature we need to ensure that the features of interest are supported by an approximately equal number of input patterns. We also require that the absolute values of readings in individual dimensions in the n-dimensional pattern space do not bias the results. The latter is easily accomplished by normalising the data, i.e. performing a transformation $\mathcal{N}(\mathcal{X}) = \vartheta$ so that $\boldsymbol{\vartheta} = \{\varphi_1, \varphi_2, \ldots,\varphi_N\} \subset [0\ 1]^n$. The second requirement can be expressed formally as follows: given the set of patterns (input set) ϑ of cardinality N and a number of features of interest (clusters) c, we require that each cluster has a support of approximately equal number of patterns. This can be achieved by granulating the original input data, so that one takes full advantage of the detailed information without biasing the subsequent clustering process. The support-balancing granulation process can be expressed formally as $\mathcal{G}(\boldsymbol{\vartheta}) = \Gamma$, where $\boldsymbol{\Gamma} = \{\mathbf{g}_1, \mathbf{g}_2, .., \mathbf{g}_M\}$ and N>M.

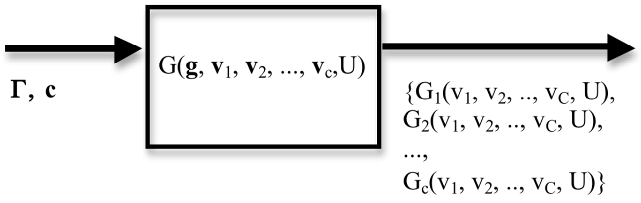
The balanced input data set **Γ** contains both numerical elements (points in $[0\ 1]^n$) and information granules (subsets of $[0\ 1]^n$). A schematic view of the formation of the balanced input set is portrayed in Figure 1.



$\mathcal{X} \subset R^n$
$\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$
$\mathcal{N}(\mathcal{X}) = \vartheta$
$\vartheta \subset [0\ 1]^n$
$\vartheta = \{\varphi_1, \varphi_2, \ldots, \varphi_N\}$
$\mathcal{G}(\vartheta) = \Gamma$
$\Gamma \subset [0\ 1]^n$
$\Gamma = \{\mathbf{g}_1, \mathbf{g}_2, .., \mathbf{g}_M\}$

Fig. 1 An overview of the transformation of the numerical data X into support-balancing granules $\tilde{G}$ G(N(*X*) = **G**

Having produced the balanced input data set **Γ** = {**g**$_1$, **g**$_2$, .., **g**$_M$} we may proceed with the construction of *c* information granules $G_1, G_2, \ldots, G_c$ representing the significant features in the input data. Depending upon the formalism of granulation used in clustering algorithm, we can arrive at G_is to be sets, fuzzy sets, rough sets, shadowed sets, etc. [28]. A schematic view of the formation of information granules is portrayed in Figure 2.



$G(\mathbf{g}, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c, U)$
Γ, c
$\{G_1(v_1, v_2, .., v_C, U),$
$G_2(v_1, v_2, .., v_C, U),$
$\ldots,$
$G_c(v_1, v_2, .., v_C, U)\}$

Fig. 2 A general view of clustering of granular data **Γ** (the granules are represented by prototypes and the partition matrix)

It is worth noting however that unlike in the standard case of clustering of numerical data we are processing here the input set Γ (which contains a mix of numerical and granular entities) thus generating prototypes $\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c$ which are themselves information granules. In the case of fuzzy sets, the clusters are characterized by the prototypes and membership functions (grades) forming the corresponding rows of the partition matrix.

The quality of clusters can be assessed via numerous cluster validity indexes [6, 22]. They could be helpful in some cases. They might also play a detrimental role by sending confusing, inconsistent messages as to the most “feasible” number of clusters. This is not surprising as each validity index originates from a certain heuristics being deemed potentially useful in focusing on some selected aspect of the structure of clusters (say, we concentrate on forming clusters as disjoint as possible). Alternatively one can think of the quality of granulation – de-granulation process in which the composition of the de-granulation and granulation mechanisms should return the result, which is as close to the original entity as possible.

The granulation mechanism returns a representation of any input data (pattern) **g** expressed in terms of membership degrees, $u_1(\mathbf{g})$, $u_2(\mathbf{g})$,…, $u_c(\mathbf{g})$. We use a concise notation $G(\mathbf{g}, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c, U)$ to underline the usage of the mechanism of granulation being applied to **g**. The de-granulation mechanism denoted by G^{-1} applies to the previously obtained result of granulation and returns a certain entity. Ideally, the obtained result should be the same as the original input **g** we have started with. In other words, we require that

$$G^{-1}(G(\mathbf{g}, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c, U))=\mathbf{g} \qquad (1)$$

It is well known (for instance, in case of quantization of continuous variables) that the above equality does not hold as we usually encounter $G^{-1}(G(\mathbf{g}, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c, U)) \approx \mathbf{g}$. Given this effect of approximate equality, we can treat a distance between the original **g** and its de-granulated version as a measure of quality of the granulation process (the quality of clusters returned by the clustering technique). In general, we may use the following index

$$\sum_{k=1}^{N} \|G^{-1}(G(\mathbf{g}_k, \mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c, U))-\mathbf{g}_k\|^2 \qquad (2)$$

that can serve as a meaningful indicator of the overall performance of the granulation mechanism. Here the symbol $\|.\|$ stands for a certain distance measure. The lower the value of Q is, the better the quality of information granules being assessed in terms of the granulation-de-granulation effect. From the perspective of the concept of information granulation, it is also not surprising that the value of the index is typically non-zero. One could anticipate that the granulation-de-granulation may produce results that are information *granules* rather than single numeric entities. The expectations could be that such information granules produced as a result of this transformation include the original input. In other words, instead of (2), we could expect a satisfaction of the following inclusion

$$\mathbf{g}_k \subset \mathbf{G}_k \qquad (3)$$

envisioning that this relationship be (hopefully) satisfied for all (or at least *most*) data $\mathbf{g}_k$. ($\mathbf{G}_k$ represents a de-granulated $\mathbf{g}_k$). The conceptual and ensuing technical question is about the origin of granularity to be used in the formation of the clusters and a way of building it in the representation of the results of clustering.

In this study we will focus on information granules $\mathbf{g}_k$ in the setting of set theory and interval analysis. The rationale behind the selection of this framework is twofold. First, interval representation of granularity leads to a clear interpretation of the results while benefiting from solid mathematical foundations of set theory. Second, the algorithmic layer of set (interval) calculus has been established for a long time and has resulted in a vast number of algorithms [15, 16]. Notwithstanding, the findings reported here can be translated and applied to other frameworks of granular computing such as fuzzy sets (the transltion hinges on the idea of representing fuzzy sets through their α-cuts [11, 19]; that is, splitting the problem into a family of interval-based granulation tasks).

This chapter is organised as follows. In Section 2, we outline areas in which information granulation plays a crucial role. Section 3, provides a detailed two-level algorithm for information granulation. Summarization of information granules (in terms of identifying granular prototypes) is discussed in Section 4. The verification of the adequacy of granular prototypes is addressed in Section 5.

3.2 Granular Information in Systems Modeling

There are a number of representative domains where information granules can emerge as a useful vehicle to represent a given problem and make problem solving more efficient [18]. The following three areas are among the most prominent applications of information granulation:

Granulation of Time Series. Time series are commonly encountered in numerous practical problems. There have been various approaches to the description of time series and their classification. They are carried out in the time domain and frequency domain. Prior to any detailed processing, time series are compressed in order to retain the most essential information and suppress details that are deemed redundant from the standpoint of further classification and processing. The essence of granulation of time series is to "discover" dominant components of the series. We may perceive these components as playing a role of basic conceptual blocks easily understood by humans and capturing the semantics of the underlying phenomenon. For instance, information granules may be formed as segments of consecutive samples of the signal. Then each segment may be labeled according to the configuration of the samples, say rapidly increasing signal, steady signal, slowly decreasing signal, etc. Alternatively, as we propose here, one may consider granulating the time series value with its gradient (and/or higher order derivatives) in individual time instances. Note that standard sampling techniques are very specific examples of granulation of time series (as we attempt to capture a segment of a signal falling under a given sampling window by a single numeric value)

Granulation of Digital Images. Digital images are two-dimensional relations. As far as understanding and processing of images is concerned, a crux there is to identify some higher level entities rather than being buried in a minute analysis completed at the level of individual pixels. Such tangible and semantically sound entities are information granules. They may arise at the level of basic homogeneous regions (in terms of brightness, color and texture) one can identify in an image. These entities are inherently hierarchical: at a higher level we may think of individual objects in the image (that are composed of the granules arising at the lower level with more specific and less abstract information granules). At the technical end, the simplest and least abstract information granules are formed by defining n-by-m blocks of pixels. At the higher level, we are concerned with various clustering techniques that help us construct abstractions out of the low-end (more detailed) information granules such as the already mentioned blocks of pixels.

Granulation of Spatial Structures. An array of current modeling pursuits occurs in the realm of distributed systems such as networks (both physical and virtual). Obvious examples of these architectures are public utility networks, telecommunication networks, social networks, supply chain networks, etc. In spite of their evident diversity, the networks share several profound commonalties. In particular, a hierarchical type of modeling is omnipresent there. Instead of analyzing the entire network, we split it into subnetworks (modules) that are loosely connected and proceed with a detailed analysis at this level. Obviously, this task is more tangible and manageable from the computational and interpretation standpoint. Each subnetwork is an information granule that is afterwards treated as a conceptual and algorithmic entity. For instance, when looking into a flow of traffic in a complex network, we partition the network into modules (call them telecommunications granules) and study all incoming and outgoing traffic from this perspective. The concept of hierarchy and information granulation is inherently associated with GIS (Geographic Information Systems) systems where we anticipate various levels of detail and control the process of concentrating on specific aspects by establishing proper levels of information granularity.

3.3 Information Density Based Granulation

In this section we focus on the algorithmic layer of the transformation of the numerical data set $\mathcal{X}$ into a balanced granular data set Γ, as illustrated in Figure 1. The normalisation stage, $\mathcal{N}(\mathcal{X}) = \vartheta$, transforming input set $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \subset R^n$ into $\vartheta = \{\varphi_1, \varphi_2, \ldots, \varphi_N\} \subset [0\ 1]^n$ is straightforward and can be expressed for individual input data as:

$$\vartheta_i = \mathbf{N}\,\mathbf{x}_i - \mathbf{o} \tag{3}$$

where $\mathbf{N} \in \mathbf{R}^{DxD}$ is a normalisation matrix

$$\mathbf{N} = \begin{bmatrix} \frac{1}{\max_i x_{i,1} - \min_i x_{i,1}} & 0 & . & 0 \\ 0 & \frac{1}{\max_i x_{i,2} - \min_i x_{i,2}} & . & 0 \\ . & . & . & . \\ 0 & 0 & . & \frac{1}{\max_i x_{i,D} - \min_i x_{i,D}} \end{bmatrix}$$

and $\mathbf{o} \in \mathbf{R}^{Dx1}$ is an offset vector constructed as

$$\mathbf{o} = \begin{bmatrix} \dfrac{\min\limits_i x_{i,1}}{\max\limits_i x_{i,1} - \min\limits_i x_{i,1}} \\ \dfrac{\min\limits_i x_{i,2}}{\max\limits_i x_{i,2} - \min\limits_i x_{i,2}} \\ \cdot \\ \dfrac{\min\limits_i x_{i,D}}{\max\limits_i x_{i,D} - \min\limits_i x_{i,D}} \end{bmatrix}$$

with max- and min- operations performed over all data items i $\in \{1, 2, \ldots, N\}$ in the D-dimensional pattern space.

The support-balancing granulation $\mathcal{G}(\vartheta) = \Gamma$ arises as a compromise between two conflicting requirements:

1. Each granule $\mathbf{g}_i$ should embrace as many elements of ϑ as possible (to be a sound representation of the underlying data). This can be expressed formally as maximising the cardinality of the $\mathbf{g}_i$ set.
2. The granule should be highly specific (its size in every dimension in the multi-dimensional pattern space should be as small as possible). The size could be measured in each dimension as a simple interval but it is convenient to represent this in a slightly more general way as function defined on such intervals (this is to avoid numerical problems with intervals of zero-length).

The algorithm implementing such a granulation has been proposed in [2] and it is briefly outlined here. The granulation is implemented as a one-pass process:

1. Initialise data structures representing cardinality and the width of individual data items (1 and 0, respectively for the point-data)
2. Calculate and store the values of "information density" (the ratio of the cardinality of the granule and the functional measurement of its size) of hypothetical granules formed by any two data items in the input data set. This forms an upper-diagonal matrix ***D*** of size NxN, where N is the cardinality of the input data set.
3. Find the maximum entry in ***D***.
4. If the maximum corresponds to an off-diagonal element (i-th and j-th coord):

- merge the two information items (identified by the i-th row and j-th column) into a single information granule, which has width defined by the maximum and minimum values of coordinates in each dimension from the two component granules; i.e.:
  - find the min of each of the coordinates of the i-th and j-th granule;
  - find the max of each of the coordinates of the i-th and j-th granule;
  - modify the i-th granule so that its size is defined by min- and max- values identified above;

- update the cardinality of the resulting granule to the sum of the cardinality counts of the component granules;
- update the i-th row and column of ***D*** with the information pertinent to the newly formed information granule and remove the j-th row and column from ***D***; i.e.:
    - copy rows 1 to j-1 from matrix ***D*** to matrix ***D1***;
    - copy rows j+1 to size(***D***) from matrix ***D*** as rows j to size(***D***)-1 in martix ***D1***;
    - copy columns 1 to j-1 from matrix ***D***1 to matrix ***D2***;
    - copy columns j+1 to size(***D***) from matrix ***D1*** as columns j to size(***D***)-1 in martix ***D2***;
    - overwrite matrix ***D*** with matrix ***D2***;
- return to 3)

5. If the maximum corresponds to a diagonal element (i=j):

- copy the granule to an output list and remove the corresponding row and column from matrix ***D***; i.e.:
    - copy rows 1 to j-1 from matrix ***D*** to matrix ***D1***;
    - copy rows j+1 to size(***D***) from matrix ***D*** as rows j to size(***D***)-1 in martix ***D1***;
    - copy columns 1 to j-1 from matrix ***D1*** to matrix ***D2***;
    - copy columns j+1 to size(***D***) from matrix ***D1*** as columns j to size(***D***)-1 in martix ***D2***;
    - overwrite matrix ***D*** with matrix ***D2***;

6. If the size of matrix ***D*** is greater than 1, return to 3), otherwise terminate.

Computational complexity of this granulation algorithm is $O(N^2)$ owing to the computations of matrix ***D*** in step 2). However, unlike the clustering techniques (such as FCM), the granulation process has an inherently local character and can be easily applied to a partitioned input data thus circumventing the high computational cost associated with large data sets. It is worth pointing out that the size of matrix ***D*** is being reduced by one row and column at each iteration thus the number of iterative steps equals N-1. Since the algorithm maintains linear computational complexity with respect of the input space dimension (not to be confused with the complexity with respect of the cardinality of the data set which is $O(N^2)$), it is particularly suitable for processing multi-dimensional data. Also, it is worth pointing that the algorithm maintains a localized view of data. As the granulation proceeds, the identified granules do not exercise further influence on data points that remain after their removal.

It is worth noting that unlike the standard Min-Max [9, 23] the above algorithm does not make any assumptions about the maximum size of granules. Granules are allowed to grow as long as their local data density keeps increasing. Furthermore, there are no assumptions about the separation of cluster centres. The formation of closely separated granules is largely avoided by the very nature of maximisation of information density, which tends to increase the size of granule if it means adding sufficiently large number of data items (another granule) without undue increase of its volume. If, on the other hand, the increase in volume would imply the reduction of information density, the granule does not expand and remains well separated from the neighboring granules. Another distinguishing feature of our algorithm is that it allows processing both point-size and hyperbox data. This is an important characteristic that allows hierarchical granulation of data. It should be noted that hierarchical granulation enables overcoming the limitations of the 'local view' of data while supporting the application of the algorithm to a partitioned input data set.

To illustrate the operation of the granulation algorithm we consider a synthetic data set shown in Figure 3. The data set comprises of 100 data items in a large cluster and 10 data items in a small cluster. The granulation algorithm produces a more balanced granular data set comprising of 16 information granules; with 14 granules representing the large cluster and 2 granules representing the small cluster. Subsequent application of the algorithm to this granular data produces a further reduction of the count of granules to 5, 3 and 2 at level-2, level-3, and level-4 granulation respectively. In the same time the ratio of data items contained in the large and the small cluster is reduced from 100/10, 14/2, 4/1, 2/1, 1/1 (Figure 4a-4d).

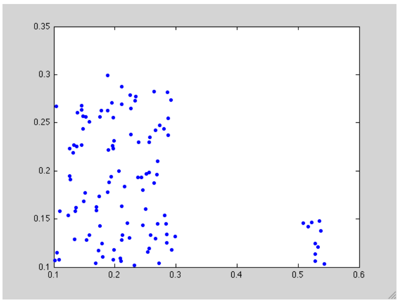


Fig. 3 Synthetic data set with two data clusters with 100 and 10 data items

It is worth noting that the number of granulation levels does not need to be defined in advance. The hierarchical granulation is simply carried out until the number of granules identified at the subsequent granulation levels does not change. Of course, in any practical application the maximum size of granules is frequently pre-defined so that the granules map well onto some linguistic entities.

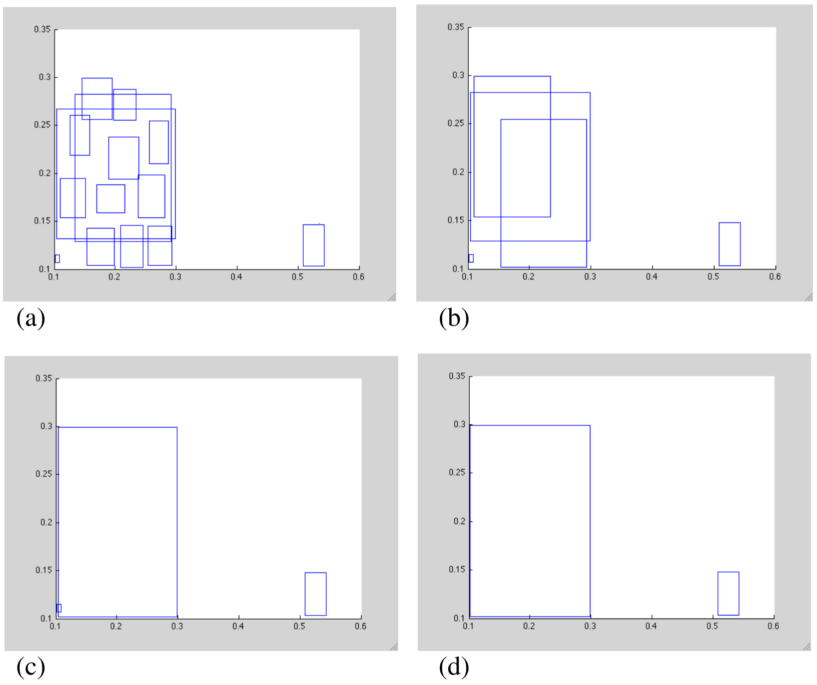


Fig. 4 (a) Information granules produced by the algorithm applied to the original data (16 granules); (b) Level-two information granules (5 granules); (c) Level-three information granules (3 granules); and (d) Level-four information granules (2 granules).

3.4 Granular Representatives of Data

The recursive application of the granulation algorithm discussed in the previous section, condense the data quite significantly. What is of fundamental interest though is whether this ‘condensing’ preserves the essential characteristics of data. We assess here the ability to preserve the essential characteristics of data by identifying a limited number of representatives of both the original numeric data **x** and the constructed information granules **g**. This is accomplished by clustering and identifying prototypes (representatives) of the original data and the granules, cf [7, 8]. In particular, we adopt a fuzzy clustering method - a well-known FCM algorithm [6]. It is instructive to recall briefly the formal description of the FCM algorithm so as to appreciate the nature of the optimisation problem represented in Figure 2. The granulation of **g** returns its representation through the collection of available information granules expressed in terms of their prototypes. More specifically, **g** is expressed in the form of the membership grades of **g** to the individual granules G_i, which form a solution to the following optimisation problem

$$\text{Min} \sum_{i=1}^{c} u_i^m(\boldsymbol{g}) \, \| \boldsymbol{g} - \mathbf{v}_i \|^2$$

subject to constraints

$$\sum_{i=1}^{c} u_i(\boldsymbol{g}) = 1 \quad u_i(\boldsymbol{g}) \in [0,1] \tag{4}$$

where "m" stands for the fuzzification coefficient, m > 1. We note however that, unlike in the original FCM formulation, the data **g** is not exclusively numeric (i.e. $\mathbf{g} \notin \mathbf{R}^n$) but involves information granules represented as hyperboxes in $\mathbf{R}^n$. In other words $\mathbf{g}=[\mathbf{g}^-, \mathbf{g}^+]$ where $\mathbf{g}^-$ are the min-value coordinates and $\mathbf{g}^+$ are the max-value coordinates of a hyperbox **g**. Since both $\mathbf{g}^-, \mathbf{g}^+ \in \mathbf{R}^n$ by concatenating the min- and max-coordinates of the hyperbox we can represent it as $\mathbf{g} \in \mathbf{R}^{2n}$. In this expanded space any numerical data **x** is represented as $\mathbf{g}=[\mathbf{x}^-, \mathbf{x}^+]$, where $\mathbf{x}=\mathbf{x}^-=\mathbf{x}^+$. The distance $\| \boldsymbol{g} - \mathbf{v}_i \|^2$ can be interpreted as the sum of distances between the minimum- and maximum-value coordinates of the respective hyperboxes, as illustrated in Figure 5.

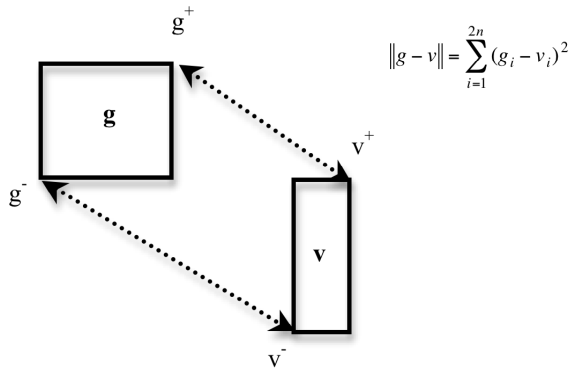
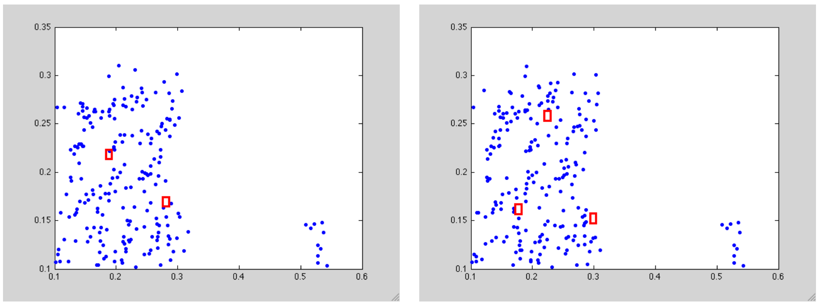


Fig. 5 Distance between hyperboxes.

Consequently the FCM solution derived in the augmented data space ($\mathbf{R}^{2n}$) reads as follows

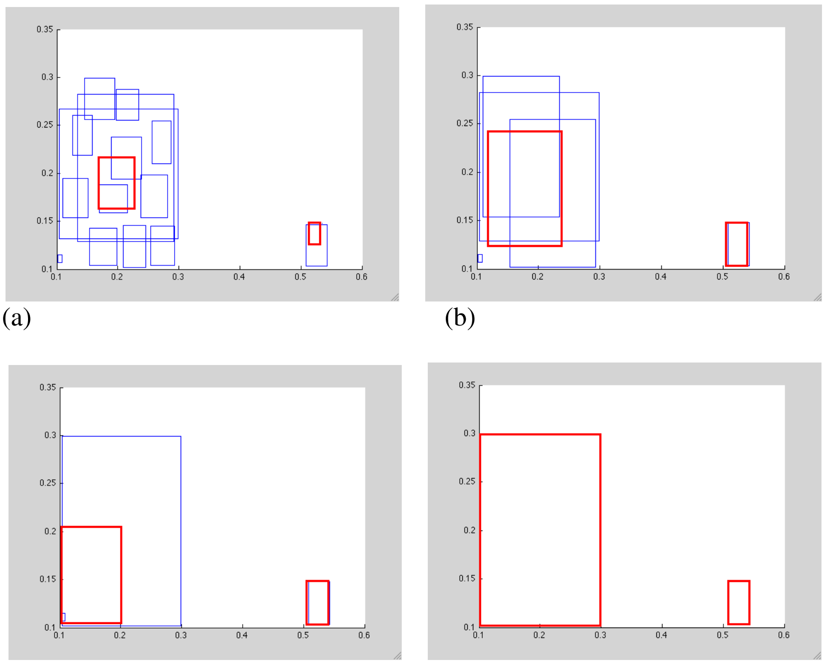
$$u_i(\boldsymbol{g}) = \frac{1}{\sum_{j=1}^{c} \left(\frac{\| \boldsymbol{g} - \mathbf{v}_i \|}{\| \boldsymbol{g} - \mathbf{v}_j \|} \right)^{2/(m-1)}} \tag{5}$$

Applying the FCM algorithm to the original and to the granulated data we identify prototypes **v** (illustrated in Figure 6 and 7) and the partition matrices u(**g**). It is clear that the prototypes evaluated for the original data are biased towards the more numerous data in the large cluster. Although we have specified 2 and 3



(a) (b)

Fig. 6 FCM prototypes found for the original numeric data; (a) two prototypes; (b) three prototypes; (the prototypes are numerical but for the sake of clarity are represented here as rectangles). Note that none of the prototypes is positioned within the small data cluster



(a) (b)

(c) (d)

Fig. 7 FCM prototypes found for the balanced, granulated data; (a) level-one; (b) level-two; (c) level-three; (d) level-four granulation; Note that for all granulation levels the small cluster of data is represented by a granule.

prototypes as a target for the FCM calculations (which should be sufficient to characterize the two clusters), the small cluster of data appears to be overwhelmed by the sheer numbers of data points in the large cluster. This is an unfortunate effect because the lack of representation of the small cluster by a local prototype is certain to lead to subsequent misclassifications of data.

By contrast, the FCM prototypes derived for the granulated data (Figure 7) capture the presence of two data clusters at all levels of data granulation by associating one prototype with each data cluster. However, the size of the granular prototypes varies depending on the level of granulation of the original data. This suggests that there is a need to develop a criterion for the selection of the appropriate level of granularity of the prototypes. The actual values of min- and max-coordinates for the various levels of data granulation are given in Table 1.

Table 1 FCM Prototypes for different levels of data granulation

Granulation Level	**Min/Max coordinates of prototypes**			
	v^-_1	v^-_2	v^+_1	v^+_2
Level 0; original numeric data (Fig. 6a)	0.1865 0.2708	0.2203 0.1580	0.1865 0.2708	0.2203 0.1580
Level 1; granulated data (Fig. 7a)	0.5127 0.1690	0.1261 0.1634	0.5309 0.2275	0.1483 0.2166
Level 2; granulated data (Fig. 7b)	0.1178 0.5046	0.1235 0.1034	0.2378 0.5392	0.2421 0.1475
Level 3; granulated data (Fig. 7c)	0.5054 0.1027	0.1034 0.1045	0.5408 0.2011	0.1483 0.2047
Level 4; granulated data (Fig. 7d)	0.5082 0.1016	0.1034 0.1020	0.5429 0.2988	0.1476 0.2995

One possible approach to the assessment of the quality of the FCM solution is to measure its ability to represent the majority of the original data, [21]. This can be accomplished by reconstructing original **g** using the prototypes **v** and the membership grades $u_i(\mathbf{g})$. It should be noted that although **g** represents the granulated data, it can also be considered as a representation of the original numeric data where each data point $\mathbf{x} \in \mathbf{R}^n$ is seen as a granule $\mathbf{g}=[\mathbf{x},\mathbf{x}] \in \mathbf{R}^{2n}$. The data reconstruction task can be formally expressed as evaluation of the vector $\hat{g}$ through a solution to the minimization problem

$$\sum_{i=1}^{c} u_i^m(\boldsymbol{g}) \, \| \hat{\boldsymbol{g}} - \mathbf{v}_i \|^2 \tag{6}$$

Because of the use of the Euclidean distance, the calculations here are straightforward yielding the result

$$\hat{\boldsymbol{g}} = \frac{\sum_{i=1}^{c} u_i^m(\boldsymbol{g})\mathbf{v}_i}{\sum_{i=1}^{c} u_i^m(\boldsymbol{g})} \tag{7}$$

3.5 Granular Refinement of Prototypes

Granular prototypes **v** obtained through the application of the FCM are dependent on the level of granulation of the original data (including the case where no data-balancing granulation has been attempted). In order to free the prototypes from this dependency, we refine the structure of the prototypes by admitting a variation of their granularity

$$V_{ij} = [\mathbf{v}_{ij}^{\min} - \sum \text{range}_j,\ \mathbf{v}_{ij}^{\max} + \sum \text{range}_j] \tag{8}$$

i=1, 2,…, c, j=1, 2, …,n and $V_{ij} \in [0\ 1]^n$. Note that each prototype increases its *granularity* to the same extent with regard to all variables. Both min-coordinates $\mathbf{v}_{ij}^{\min}$ and max-coordinates $\mathbf{v}_{ij}^{\max}$ of the prototype are transformed so as to produce a symmetrical enlargement of v_{ij} by the imposed level of granularity $\sum$.

The main limitation of this construction is that all variables are treated in the same way by assigning to all of them the same value of $\sum$. We attempt to offset some of this limitation by using a granule-specific multiplier *range* that promotes the preservation of topological similarity between the original and the modified prototypes. However, if the modified granule extends outside the unit hyperbox, we enforce the requirement that $V_{ij} \in [0\ 1]^n$ thus, in effect we generate a non symmetrical expansion of the prototype v_{ij}. We denote the resulting prototype-specific expansion factors as $\tilde{\varepsilon}_i$.

As the granularity is sought of as an important modeling resource to be prudently allocated, its distribution needs more attention. We propose here that the sum of all $\tilde{\varepsilon}_i$ is minimized. In other words we establish a performance criterion as

$$Q = \min \sum_{i=1}^{c}\sum_{j=1}^{n} \tilde{\varepsilon}_{ij} \tag{9}$$

where $\tilde{\varepsilon} = [\tilde{\varepsilon}_1, \tilde{\varepsilon}_2, \ldots, \tilde{\varepsilon}_c]$, $\tilde{\varepsilon}_i = [\tilde{\varepsilon}_{i1}, \tilde{\varepsilon}_{i2}, \ldots, \tilde{\varepsilon}_{in}]$. We require that every original data item g_k is enclosed in the reconstructed data item represented by a hyperbox $\hat{g}_k$ evaluated as in (7). We can express this formally as

$$\text{card}\{\boldsymbol{g}_k \in G^{-1}(G(\boldsymbol{g}_k, \mathbf{V}_1(\tilde{\varepsilon}_1), \mathbf{V}_2(\tilde{\varepsilon}_2), \ldots, \mathbf{V}_c(\tilde{\varepsilon}_c), U)\} = N \tag{10}$$

The constraint (10) represents the most stringent requirement on the reconstruction of data and may in some cases be replaced by a less stringent requirement, that a given proportion of data is correctly reconstructed. Evidently, the high values of $\tilde{\varepsilon}_i$ are more likely to satisfy (10) but the resulting prototypes would imply lack of specificity and might not be acceptable in many applications. In other words, the performance index (9) captures the nature of consistent and parsimonious granular representation of data through clustering: we strive to achieve a situation where all patterns when being represented by the granular prototypes are positioned within the bounds resulting through the reconstruction process (7). By noting an indirect way in which the constraint (10) depends upon the vector of variables $\tilde{\varepsilon}_i$ we resort to the use of a population–based optimisation method. One of the viable alternatives is the Particle Swarm Optimisation (PSO), see [24].

A particle swarm is a population of particles representing possible solutions located in the multidimensional search space [14, 24, 29]. Each particle explores the search space and during this search it adheres to some quite intuitively appealing guidelines navigating the search process: (a) it tries to follow its previous direction, and (b) it looks back at the best performance recorded so far both at the level of the individual particle as well as the entire population.

The algorithm exhibits some societal aspects of interaction. There is some collective search of the problem space along with some component of memory incorporated as an integral part of the search mechanism. The performance of each particle during its traversal of the search space is assessed by means of some performance index (fitness function). A position of a swarm in some search space **S** of a given dimensionality “r”, is described by some vector **z**(t)**S** where ”t” denotes consecutive discrete time moments (generation index). The method can be briefly explained as follows. The next position of the particle is governed by the following update expressions concerning the particle, **z**(t+1) and its speed, **v**(t+1)

$$z_i(t+1) = z_i(t) + v_i(t+1) \qquad \text{//update of position of the particle}$$
$$v_i(t+1) = \xi\, v_i(t) + f_{1i}(p_i - z_i(t)) + f_{2i}\,(p_{total,i} - z_i(t)) \quad \text{// update of speed of the particle} \tag{11}$$

i=1, 2, …, r where **p** denotes the best position (characterized by the lowest performance index) reported so far for this particle, $\mathbf{p}_{total}$ is the best position overall developed so far across the entire population. The two other parameters of the PSO that is f_{1i} and f_{2i} are random numbers drawn from the uniform probability distribution defined over the [0,2] interval that help build a proper combination of the components of the speed; different random numbers affect the individual coordinates of the speed.

The second expression governing the change in the velocity of the particle is particularly interesting as it captures the relationships between the particle and its history as well as the history of the overall population in terms of their performance reported so far. The current speed **v**(t) is impacted by the inertial weight (ξ) smaller than 1 whose role is to articulate some factor of resistance to change the current speed (the values of the inertia weight are kept below 1). The fitness function is given by (9). The particle consists of the coordinates of $\tilde{\varepsilon}_i$. Before it is used to evaluate the fitness function, the coordinates are normalized so that they satisfy the requirement expressed by (10).

Application of PSO to the prototypes reported in Table 1 produced optimised prototypes at each level of data granulation. The results detailed in Table 2 and Figure 8 suggest that the Level-2 granulation produces the best results in terms of interpretability and representativeness of granular prototypes of data.

Table 2 Refined prototypes for different levels of data granulation with the corresponding value of the performance index.

Granulation Level	**Coordinates of refined prototypes**				**Q**
	v^-_1	v^-_2	v^+_1	v^+_2	
Level 0; numeric data (Fig. 8a)	0 0	0 0	0.4902 0.5735	0.5241 0.4632	2.0509
Level 1; granulated data (Fig. 8b)	0.4157 0.0720	0.0291 0.0664	0.6279 0.3244	0.2453 0.3136	0.7760
Level 2; granulated data (Fig. 8c)	0.4456 0.0588	0.0444 0.0645	0.5982 0.2968	0.2065 0.3011	**0.4720**
Level 3; granulated data (Fig. 8d)	0.4064 0.0037	0.0044 0.0055	0.6398 0.3002	0.2473 0.3038	0.7538

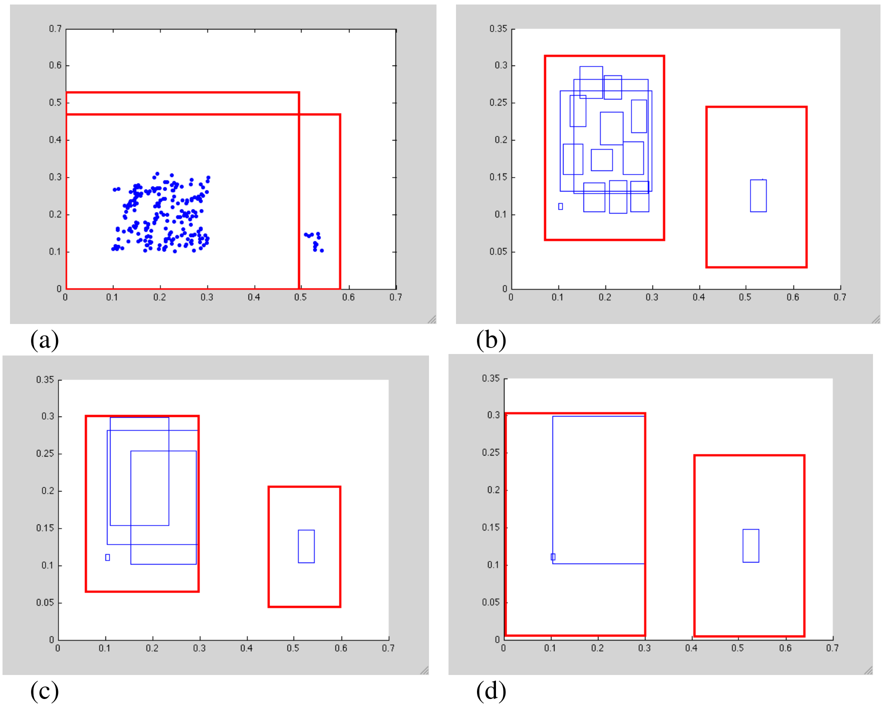


Fig. 8 FCM prototypes refined by a granular expansion; (a) Original numeric data; (b) Level-one; (c) Level-two; (d) Level-three granulated data. The prototypes satisfy the full reconstruction requirement (10).

3.6 Conclusions

Aggregation of detailed numerical information into information granules promotes a more global view of data and of the application domain from which the data is derived. However, the effectiveness of this information abstraction depends on the quality of the resulting information granules. This study focuses on two essential characteristics of information abstraction: the interpretability and the representativeness of information granules. By adopting the min/max granulation framework we generate highly interpretable hyperboxes in the pattern space while accepting the expense of having to use a greater number of hyperboxes to represent some of the more complex topologies of data clusters. The increasing levels of generalization of data lead naturally to a hierarchical structure of the proposed granulation method. The proposed evaluation and subsequent refinement of granular prototypes – representing the information granules at each level of abstraction – offers a means of maximizing the expressive power of information granules by ensuring that there is no loss of information in the granulation-degranulation process.

The study opens a large spectrum of possibilities for the refinement of the balance between the interpretability and the representativeness of information granules. In particular, some applications might be tolerant of some loss of the ability to represent fully the original data if this meant the enhancement of the specificity (small size) of the prototypes. Conversely, other applications may depend critically on the ability of the prototypes to represent all input data. The small synthetic data set discussed here is intended to promote such considerations as opposed to providing a reference solution for a specific application.

References

1. Bargiela, A., Pedrycz, W.: Granular Computing: An Introduction. Kluwer Academic Publishers, Dordrecht (2003)
2. Bargiela, A., Pedrycz, W.: Recursive information granulation: Aggregation and interpretation issues. IEEE Trans. on Syst. Man and Cybernetics 33(1), 96–112 (2003)
3. Bargiela, A., Pedrycz, W.: Granular mappings. IEEE Transactions on Systems, Man, and Cybernetics-part A 35(2), 292–297 (2005)
4. Bargiela, A., Pedrycz, W.: A model of granular data: a design problem with the Tchebyschev FCM. Soft Computing 9, 155–163 (2005)
5. Bargiela, A., Pedrycz, W.: Toward a theory of Granular Computing for human-centered information processing. IEEE Transactions on Fuzzy Systems 16(2), 320–330 (2008)
6. Bezdek, J.C.: Pattern Recognition with Fuzzy Objective Function Algorithms. Plenum Press, N. York (1981)
7. Chiu, S.: Method and software for extracting fuzzy classification rules by subtractive clustering. In: NAFIPS, pp. 461–465 (1996)
8. Cios, K., Pedrycz, W., Swiniarski, R.: Data Mining Techniques. Kluwer Academic Publishers, Boston (1998)
9. Gabrys, B., Bargiela, A.: General fuzzy min-max neural network for clustering and classification. IEEE Trans. on Neural Networks 11(3), 769–783 (2000)
10. Hata, Y., Mukaidono, M.: On some classes of fuzzy information granularity and their representations. In: ISMVL 1999, Japan, pp. 288–293 (1999)

11. Kandel, A.: Fuzzy Mathematical Techniques with Applications. Addison-Wesley, Reading, MA (1986)
12. Kacprzyk, J., Yager, R.R.: Linguistic summaries of data using fuzzy logic. Int. J. General Systems 30, 33–154 (2001)
13. Kacprzyk, J., Zadrozny: Linguistic database summaries and their protoforms: toward natural language based knowledge discovery tools. Information Sciences 173, 281–304 (2005)
14. Ling, S.H., Iu, H.H.C., Chan, K.Y., Lam, H.K., Yeung, B.C.W., Leung, F.H.: Hybrid Particle Swarm Optimization with wavelet mutation and its industrial applications. IEEE Transactions on Systems, Man, and Cybernetics, Part B 38(3), 743–763 (2008)
15. Moore, R.E.: Interval Analysis. Prentice Hall, Englewood Cliffs (1966)
16. Kreinovich, V., Lakeyev, A., Rohn, J., Kahl, P.: Computational Complexity and Feasibility of Data Processing and Interval Computations. Kluwer, Dordrecht (1998)
17. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning about Data. Kluwer Academic, Dordrecht (1991)
18. Pedrycz, W.: Computational Intelligence: An Introduction. CRC Press, Boca Raton (1997)
19. Pedrycz, W., Gomide, F.: An Introduction to Fuzzy Sets. MIT Press, Cambridge (1998)
20. Pedrycz, W., Bargiela, A.: Information granulation: A search for data structures. In: Knowledge-based Engineering Systems KES 2001, Osaka, pp. 1147–1151 (October 2001)
21. Pedrycz, W., Valente de Oliveira, J.: A development of fuzzy encoding and decoding through fuzzy clustering. IEEE Transactions on Instrumentation and Measurement 57(4), 829–837 (2008)
22. Pedrycz, W.: Knowledge-Based Fuzzy Clustering. John Wiley, N. York (2005)
23. Simpson, P.K.: Fuzzy min-max neural networks. IEEE Transactions on Neural Networks 3, 776–786 (1992)
24. Van den Bergh, F., Engelbrecht, A.P.: A study of particle swarm optimization particle trajectories. Information Sciences 176(8), 937–971 (2006)
25. Zadeh, L.A.: Fuzzy sets and information granularity. In: Gupta, M.M., Ragade, R.K., Yager, R.R. (eds.) Advances in Fuzzy Set Theory and Applications, pp. 3–18. North Holland, Amsterdam (1979)
26. Zadeh, L.A.: Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. Fuzzy Sets and Systems 90, 111–117 (1997)
27. Zadeh, L.A.: From computing with numbers to computing with words-from manipulation of measurements to manipulation of perceptions. IEEE Trans. on Circuits and Systems 45, 105–119 (1999)
28. Zadeh, L.A.: Toward a generalized theory of uncertainty (GTU) – an outline. Information Sciences 172(1-2), 1–40 (2005)
29. Zhan, Z., Zhang, J., Li, Y., Chung, H.S.H.: Adaptive Particle Swarm optimization. IEEE Trans. on Systems, Man, and Cybernetics, Part B 39(6), 1362–1381 (2009)
30. Yao, Y.Y.: Information granulation and rough set approximation. International Journal of Intelligent Systems 16(1), 87–104 (2001)
31. Yao, Y.: A unified framework of granular computing. In: Pedrycz, W., Skowron, A., Kreinovich, V. (eds.) Handbook of Granular Computing, pp. 401–410. Wiley-Interscience, New York (2008)
32. Yao, Y.Y.: Integrative levels of granularity. In: Bargiela, A., Pedrycz, W. (eds.) Human Centric Information Processing Through Granular Modelling, pp. 31–47. Springer, Berlin (2009)

Chapter 4
Mining Incomplete Data—A Rough Set Approach

Jerzy W. Grzymala-Busse and Zdzislaw S. Hippe

Abstract. Real-life data sets are frequently affected by missing attribute values. Such missing attribute values may be interpreted as lost values, "do-not care" conditions, a special case of "do-not care" conditions called restricted "do-not care" conditions, or as attribute-concept values. In general, incomplete data are described by characteristic relations, a generalization of the indiscernibility relation. A methodology of mining incomplete data is provided and illustrated by examples. In particular, two versions of the MLEM2 system, global and local, are presented. In the MLEM2 global version approximations are computed first and then the rule induction algorithm is applied. In the local version of MLEM2, rule sets are induced directly from raw data sets, incomplete and with numerical attributes. Results of experiments on both approaches are included.

4.1 Introduction

Mining incomplete data, i.e., data with missing attribute values, is important since many real-life data are incomplete. There are two basic methods to handling missing attribute values: *sequential* and *parallel* [1]. Sequential methods are based on preprocessing, before the process of knowledge acquisition, while in parallel methods both handling missing attribute values and knowledge acquisition are conducted in parallel.

Jerzy W. Grzymala-Busse
Department of Electrical Engineering and Computer Science University of Kansas, Lawrence, KS 66045, USA
e-mail: jerzy@ku.edu
and
Institute of Computer Science, Polish Academy of Sciences, 01–237 Warsaw, Poland

Zdzislaw S. Hippe
Department of Expert Systems and Artificial Intelligence,
University of Information Technology and Management, 35-225 Rzeszow, Poland
e-mail: zhippe@wsiz.rzeszow.pl

Sequential methods include techniques based on deleting cases with missing attribute values, replacing a missing attribute value by the most common value of that attribute, assigning all possible values to the missing attribute value, replacing a missing attribute value by the mean for numerical attributes, assigning to a missing attribute value the corresponding value taken from the closest fit case [2], or replacing a missing attribute value by a new value, computed from a new data set, in which the original attribute is a decision.

Parallel methods to handle missing attribute values include the MLEM2 (Modified Learning from Examples Module, version 2) rule induction algorithm in which rules are induced form the original data set, with missing attribute values considered to be lost values, attribute-concept values, or "do not care" conditions, see, e.g., [3]. MLEM2 is an option of the LERS (Learning from Examples based on Rough Sets) data mining system. The C4.5, see [4], and CART, see [5], approaches to missing attribute values are other examples of methods from the same group.

We will distinguish four types of missing attribute values. The first type of missing attribute value will be called *lost*. A missing attribute value is lost when for some case (example, object) the corresponding attribute value was mistakenly erased or forgotten to enter into the data set. The original value existed but for a variety of reasons now it is not accessible.

The next three types of missing attribute values, called *"do not care" conditions*, *restricted "do not care" conditions* and *attribute-concept values* are based on an assumption that these values were initially, when the data set was created, irrelevant. For example, in a medical setup, patients were subjected to preliminary tests. Patients whose preliminary test results were negative were diagnosed as not affected by a disease. They were perfectly well diagnosed in spite of the fact that not all tests were conducted on them. Thus, some test results are missing because these tests were redundant. In different words, a missing attribute value of this type may be potentially replaced by any value typical for that attribute. This type of a missing attribute value will be called a "do not care" condition. A special case of a "do not care" condition, called restricted "do not care" condition, has another interpretation: a restricted "do not care" condition may be replaced by any value typical for that attribute excluding lost values. Obviously, when the data set does not have any lost values, both "do not care" conditions, ordinary and restricted, are interpreted in the same way. On the other hand, we may have different expectations, for example, if a patient was diagnosed as not affected by a disease, we may want to replace the missing test (attribute) value by any typical value for that attribute but restricted to patients in the same class (concept), i.e., for other patients not affected by the disease. Such missing attribute value will be called attribute-concept value.

Incomplete decision tables in which all attribute values are lost, from the viewpoint of rough set theory, were studied for the first time in [6], where two algorithms for rule induction, modified to handle lost attribute values, were presented. This approach was studied later, e.g., in [7] and [8], where the indiscernibility relation was generalized to describe such incomplete decision tables.

On the other hand, incomplete decision tables in which all missing attribute values are "do not care" conditions, from the view point of rough set theory, were studied for the first time in [9], where a method for rule induction was introduced in which each missing attribute value was replaced by all values from the domain of the attribute. Originally, such values were replaced by all values from the entire domain of the attribute, later, by attribute values restricted to the same concept to which a case with a missing attribute value belongs. Such incomplete decision tables, with all missing attribute values being "do not care" conditions, were extensively studied in [10], [11], including extending the idea of the indiscernibility relation to describe such incomplete decision tables.

The attribute-concept approach to missing attribute values was introduced in [12], while the restricted "do not care" conditions were introduced in [13].

We will study all four interpretations of missing attribute values: lost values, "do not care" conditions , restricted "do not care" conditions and attribute-concept values. In particular, a methodology of computing characteristic sets, a generalization of elementary sets, will be presented. A characteristic relation, a generalization of the indiscernibility relation, may be determined from characteristic sets. Then we will discuss three types of approximations: singleton, subset and concept. All these approximations are reduced to standard approximations of rough set theory if the data set is complete, i.e., if all attribute values are specified.

We may use such approximations first and then induce rules using the MLEM2 rule induction algorithm. This approach is called global. On the other hand, two algorithms to compute local lower and upper coverings, using so-called local approach, will be presented as well. These two algorithms may be applied directly to raw data, with missing attribute values and some (or all) numerical attributes. The output of these algorithms are certain and possible rules. Results of experiments comparing local and global approaches show that both approaches are worth trying: sometimes lower error rate is accomplished using the local approach, sometimes using the global approach.

4.2 Blocks of Attribute-Value Pairs

For the rest of the paper we will assume that all decision values are specified, i.e., they are not missing. In addition, we will assume that lost values will be denoted by "?", "do not care" conditions by "*", restricted "do not care" conditions by "+", and attribute-concept values by "–". Additionally, we will assume that for each case at least one attribute value is specified.

We assume that the input data sets are presented in the form of a *decision table*. An example of a decision table is shown in Table 1.

Table 1 An incomplete decision table

	Attributes			Decision
Case	Temperature	Headache	Cough	Flu
1	normal	yes	yes	yes
2	*	no	*	no
3	normal	–	no	yes
4	?	yes	yes	no
5	high	yes	+	yes
6	–	no	yes	no
7	high	?	yes	yes
8	high	yes	?	no

Rows of the decision table represent *cases*, while columns are labeled by *variables*. The set of all cases will be denoted by U. In Table 1, $U = \{1, 2, 3, 4, 5, 6, 7, 8\}$. Independent variables are called *attributes* and a dependent variable is called a *decision* and is denoted by d. The set of all attributes will be denoted by A. In Table 1, $A = \{Temperature, Headache, Cough\}$.

An important tool to analyze decision tables is a *block of an attribute-value pair*. Let (a, v) be an attribute-value pair. For *complete* decision tables, i.e., decision tables in which every attribute value is known, a block of (a, v), denoted by $[(a, v)]$, is the set of all cases x for which $a(x) = v$. For incomplete decision tables the definition of a block of an attribute-value pair is modified.

– If for an attribute a there exists a case x such that $a(x) = ?$, i.e., the corresponding value is lost, then the case x should not be included in any blocks$[(a, v)]$ for all values v of attribute a,
– If for an attribute a there exists a case x such that the corresponding value is a "do not care" condition or a restricted "do not care" condition, i.e., $a(x) = *$ or $a(x) = +$, then the case x should be included in blocks $[(a, v)]$ for all specified values v of attribute a,
– If for an attribute a there exists a case x such that the corresponding value is an attribute-concept value, i.e., $a(x) = -$, then the corresponding case x should be included in blocks $[(a, v)]$ for all specified values $v \in V(x, a)$ of attribute a, where

$$V(x, a) = \{a(y) \mid a(y) \text{ is specified}, \, y \in U, \; d(y) = d(x)\}.$$

Note that these modifications of the definition of the block of attribute-value pair are consistent with the interpretation of missing attribute values: lost, "do not care" conditions, restricted "do not care" conditions, and attribute-concept values. The attribute-concept value is the most universal, since if $V(x, a) = \emptyset$, the definition of the attribute-concept value is reduced to the lost value, and if $V(x,a)$ is the set of all values of an attribute a, the attribute-concept value becomes a "do not care" condition.

For Table 1, $V(3, Headache) = \{yes\}$ and $V(6, Temperature) = \{high\}$, so the blocks of attribute-value pairs are:

[(Temperature, normal)] = {1, 2, 3},
[(Temperature, high)] = {2, 5, 6, 7, 8},
[(Headache, yes)] = {1, 3, 4, 5, 8},
[(Headache, no)] = {2, 6},
[(Cough, yes)] = {1, 2, 4, 5, 6, 7},
[(Cough, no)] = {2, 3, 5},

For a case $x \in U$, the *characteristic set* $K_B(x)$ is defined as the intersection of the sets $K(x,a)$, for all $a \in B$, where the set $K(x,a)$ is defined in the following way:

- If $a(x)$ is specified, then $K(x,a)$ is the block $[(a, a(x)]$ of attribute a and its value $a(x)$,
- If $a(x) =?$ or $a(x) = *$ then the set $K(x,a) = U$,
- If $a(x) = +$, then $K(x,a)$ is equal to the union of all blocks of (a,v), for all specified values v of attribute a,
- If $a(x) = -$, then the corresponding case x should be included in blocks $[(a,v)]$ for all known values $v \in V(x,a)$ of attribute a, where

  $$V(x,a) = \{a(y) \mid a(y) \text{ is known}, y \in U,\ d(y) = d(x)\}.$$

  If $V(x,a)$ is empty, $K(x,a) = U$.

Note that for both lost values and "do not care" conditions the corresponding set $K(x,a)$ is equal to U because the corresponding attribute a does not restrict the set $K_B(x)$: if $a(x) =?$, only the existing values need to be checked; if $a(x) = *$, the value of the attribute a is irrelevant.

For Table 1 and $B = A$,

$K_A(1) = \{1,2,3\} \cap \{1,3,4,5,8\} \cap \{1,2,4,5,6,7\} = \{1\}$,
$K_A(2) = U \cap \{2,6\} \cap U = \{2,6\}$,
$K_A(3) = \{1,2,3\} \cap \{1,3,4,5,8\} \cap \{2,3,5\} = \{3\}$,
$K_A(4) = U \cap \{1,3,4,5,8\} \cap \{1,2,4,5,6,7\} = \{1,4,5\}$,
$K_A(5) = \{2,5,6,7,8\} \cap \{1,3,4,5,8\} \cap (\{1,2,4,5,6,7\} \cup \{2,3,5\}) = \{5\}$,
$K_A(6) = \{2,5,6,7,8\} \cap \{2,6\} \cap \{1,2,4,5,6,7\} = \{2,6\}$,
$K_A(7) = \{2,5,6,7,8\} \cap U \cap \{1,2,4,5,6,7\} = \{2,5,6,7\}$,
$K_A(8) = \{2,5,6,7,8\} \cap \{1,3,4,5,8\} \cap U = \{5,8\}$.

Characteristic set $K_B(x)$ may be interpreted as the set of cases that are indistinguishable from x using all attributes from B and using a given interpretation of missing attribute values. The *characteristic relation* $R(B)$ is a relation on U defined for $x, y \in U$ as follows

$$(x,y) \in R(B) \text{ if and only if } y \in K_B(x).$$

The characteristic relation $R(B)$ is reflexive but—in general—does not need to be symmetric or transitive. Additionally, the characteristic relation $R(B)$ is known if we know characteristic sets $K_B(x)$ for all $x \in U$. In our example, $R(A)$ = {(1, 1), (2, 2), (2, 6), (3, 3), (4, 1), (4, 4), (4, 5), (5, 5), (6, 2), (6, 6), (7, 2), (7, 5), (7, 6) (7, 7), (8, 5), (8, 8)}.

For a complete decision table, the characteristic relation $R(B)$ is reduced to the indiscernibility relation [14]. Recently characteristic relations were investigated in a number of papers, see, e.g., [15, 16, 17, 18].

Definability for completely specified decision tables should be modified to fit into incomplete decision tables. For incomplete decision tables, a union of some intersections of attribute-value pair blocks, where such attributes are members of B and are distinct, will be called B-*locally definable* sets. A union of characteristic sets $K_B(x)$, where $x \in X \subseteq U$ will be called a B-*globally definable* set. Any set X that is B-globally definable is B-locally definable, the converse is not true.

For example, the set {2} is A-locally definable since {2} = [(*Temperature*, *normal*)] ∩ [(*Headache*, *no*)]. However, the set {2} is not A-globally definable. On the other hand, the set {4} = is not even locally definable since in all blocks of attribute-value pairs containing the case 4 contain also the case 1 as well.

Obviously, if a set is not B-locally definable then it cannot be expressed by rule sets using attributes from B. This is why it is so important to distinguish between B-locally definable sets and those that are not B-locally definable.

4.3 Approximations

For completely specified decision tables lower and upper approximations are defined based on the indiscernibility relation [14, 19]. Let X be any subset of the set U of all cases. The set X is called a *concept* and is usually defined as the set of all cases defined by a specific value of the decision. In general, X is not a B-definable set. However, set X may be approximated by two B-definable sets, the first one is called a *B-lower approximation* of X, denoted by $\underline{B}X$ and defined as follows

$$\{x \in U \mid [x]_B \subseteq X\},$$

where $[x]_B$ denotes an equivalence class of x with respect to the indiscernibility relation R associated with B. The second set is called a *B-upper approximation* of X, denoted by $\overline{B}X$ and defined as follows

$$\{x \in U \mid [x]_B \cap X \neq \emptyset\}.$$

The above shown way of computing lower and upper approximations, by constructing these approximations from singletons x, will be called the *first method.* The B-lower approximation of X is the greatest B-definable set, contained in X. The B-upper approximation of X is the smallest B-definable set containing X.

As it was observed in [14], for complete decision tables we may use a *second method* to define the B-lower approximation of X, by the following formula

$$\cup\{[x]_B \mid x \in U, [x]_B \subseteq X\},$$

and the B-upper approximation of x may be defined, using the second method, by

$$\cup\{[x]_B \mid x \in U, [x]_B \cap X \neq \emptyset\}.$$

Obviously, for complete decision tables both methods result in the same respective sets, i.e., corresponding lower approximations are identical, and so are upper approximations.

For incomplete decision tables lower and upper approximations may be defined in a few different ways. In this paper, we suggest three different definitions of lower and upper approximations for incomplete decision tables, following [20, 21, 22]. Again, let X be a concept, let B be a subset of the set A of all attributes, and let $R(B)$ be the characteristic relation of the incomplete decision table with characteristic sets $K_B(x)$, where $x \in U$. Our first definition uses a similar idea as in the previous articles on incomplete decision tables [7, 8, 10, 11], i.e., lower and upper approximations are sets of singletons from the universe U satisfying some properties. Thus, lower and upper approximations are defined by analogy with the above first method, by constructing both sets from singletons. We will call these approximations *singleton*. A singleton B-lower approximation of X is defined as follows:

$$\underline{B}X = \{x \in U \mid K_B(x) \subseteq X\}.$$

A singleton B-upper approximation of X is

$$\overline{B}X = \{x \in U \mid K_B(x) \cap X \neq \emptyset\}.$$

In our example of the decision table presented in Table 1 let us say that $B = A$. Then the singleton A-lower and A-upper approximations of the two concepts: {1, 3, 5, 7} and {2, 4, 6, 8} are:

$$\underline{A}\{1,3,5,7\} = \{1,3,5\},$$

$$\underline{A}\{2,4,6,8\} = \{2,6\},$$

$$\overline{A}\{1,3,5,7\} = \{1,3,4,5,7,8\},$$

$$\overline{A}\{2,4,6,8\} = \{2,4,6,7,8\}.$$

We may easily observe that the set {1, 3, 4, 5, 7, 8} = $\overline{A}\{1,3,5,7\}$ is not A-locally definable since in all blocks of attribute-value pairs cases 6 and 7 are inseparable. Similarly, $\overline{A}\{2,4,6,8\}$ is also not locally definable. Thus, as it was observed in, e.g., [20, 21, 22], singleton approximations should not be used, theoretically, for data mining and, in particular, for rule induction.

For the decision table from Table 1 both singleton lower approximations are globally definable. The following table shows that singleton lower approximations may be not definable as well.

Table 2 An incomplete decision table

	Attributes		Decision
Case	Temperature	Headache	Flu
1	normal	no	no
2	normal	*	no
3	?	no	yes
4	normal	yes	yes
5	high	yes	yes

For Table 2

[(Temperature, normal)] = {1, 2, 4},
[(Temperature, high)] = {5},
[(Headache, yes)] = {2, 4, 5},
[(Headache, no)] = {1, 2, 3},

$K_A(1) = \{1, 2, 4\} \cap \{1, 2, 3\} = \{1, 2\}$,
$K_A(2) = \{1, 2, 4\} \cap U = \{1, 2, 4\}$,
$K_A(3) = U \cap \{1, 2, 3\} = \{1, 2, 3\}$,
$K_A(4) = \{1, 2, 4\} \cap \{2, 4, 5\} = \{2, 4\}$,
$K_A(5) = \{5\} \cap \{2, 4, 5\} = \{5\}$.

The singleton lower approximation for the concept [(*Flu*, *no*)] is $\underline{A}\{1, 2\} = \{1\}$,while the set $\{1\}1$ is not even locally definable since in all attribute-value blocks containing the case 1 there exists the case 2.

The second method of defining lower and upper approximations for complete decision tables uses another idea: lower and upper approximations are unions of elementary sets, subsets of U. Therefore, we may define lower and upper approximations for incomplete decision tables by analogy with the second method, using characteristic sets instead of elementary sets. There are two ways to do this. Using the first way, a *subset* B-lower approximation of X is defined as follows:

$$\underline{B}X = \cup\{K_B(x) \mid x \in U, K_B(x) \subseteq X\}.$$

A *subset* B-upper approximation of X is

$$\overline{B}X = \cup\{K_B(x) \mid x \in U, K_B(x) \cap X \neq \emptyset\}.$$

Since any characteristic relation $R(B)$ is reflexive, for any concept X, singleton B-lower and B-upper approximations of X are subsets of the subset B-lower and B-upper approximations of X, respectively [22]. For the same decision table, presented in Table 1, the subset A-lower and A-upper approximations are

$$\underline{A}\{1,3,5,7\} = \{1,3,5\},$$

$$\underline{A}\{2,4,6,8\} = \{2,6\},$$

$$\overline{A}\{1,3,5,7\} = U,$$

$$\overline{A}\{2,4,6,8\} = \{1,2,4,5,6,7,8\}.$$

The second possibility is to modify the subset definition of lower and upper approximation by replacing the universe U from the subset definition by a concept X. A *concept* B-lower approximation of the concept X is defined as follows:

$$\underline{B}X = \cup\{K_B(x) \mid x \in X, K_B(x) \subseteq X\}.$$

Obviously, the subset B-lower approximation of X is the same set as the concept B-lower approximation of X. A *concept* B-upper approximation of the concept X is defined as follows:

$$\overline{B}X = \cup\{K_B(x) \mid x \in X, K_B(x) \cap X \neq \emptyset\} = \\ = \cup\{K_B(x) \mid x \in X\}.$$

The concept upper approximations were defined in [23] and [24] as well. The concept B-upper approximation of X is a subset of the subset B-upper approximation of X [22]. For the decision table presented in Table 1, the concept A-upper approximations are

$$\overline{A}\{1,3,5,7\} = \{1,2,3,5,6,7\},$$

$$\overline{A}\{2,4,6,8\} = \{1,2,4,5,6,8\}.$$

Note that for complete decision tables, all three definitions of lower approximations, singleton, subset and concept, coalesce to the same definition. In addition, for complete decision tables, all three definitions of upper approximations coalesce to the same definition. This is not true for incomplete decision tables, as our example shows.

4.4 Two Algorithms

Let X be any subset of the set U of all cases. Let $B \subseteq A$. In general, X is not a B-definable set, locally or globally. For a set T of attribute-value pairs, the intersection of blocks for all t from T will be denoted by $[T]$. For the rest of the paper we will assume that any set T consists of attribute-value pairs with all different attributes. A set T of attribute-value pairs, where all attributes belong to the set B, will be called a *B-complex* of X if and only if

$$\emptyset \neq [T] = \cap\{[t] \mid t \in T\} \subseteq X.$$

The *B-local lower* approximation of the concept X is defined as follows

$$\cup\{[T] \mid T \textit{ is a } B\textit{–complex of } X,\ [T] \subseteq X\}.$$

The *B-local upper* approximation of the concept X is defined as the minimal set containing X and defined in the following way

$$\cup\{[T] \mid \exists \textit{ a family } \mathcal{T} \textit{ of } B\textit{–complexes } T \textit{ of } X \textit{ with } \forall\ T \in \mathcal{T},\ [T] \cap X \neq \emptyset\}.$$

Obviously, the B-local lower approximation of X is unique and it is the largest B-locally definable set contained in X. Any B-local upper approximation of X is B-locally definable, it contains X, and is, by definition, the smallest. Note that a concept may have more than one local upper approximation [25].

Let $\mathcal{T}$ be a family of sets T of attribute-value pairs. A set $\mathcal{T}$ will be called a *local lower covering* of X if and only if the following three conditions are satisfied:

(1) $\bigcup_{T \in \mathcal{T}} [T] \subseteq X$,

(2) every $T \in \mathcal{T}$ is minimal, i.e., no proper subset T' of T exists with $[T'] \subseteq X$,

(3) $\mathcal{T}$ is minimal, i.e., for every $T \in \mathcal{T}$, $\bigcup_{S \in \mathcal{T}-\{T\}} [S] \neq \bigcup_{S \in \mathcal{T}} [S]$.

The procedure for determining a single local lower covering, based on the MLEM2 algorithm, is presented below. Notation used for numerical attributes is explained in Section 4.8.

```
Procedure for determining a single local lower covering
input: a set $X$ (a subset of $U$),
output: a single local lower covering $\mathcal{T}$ of the set $X$,
begin
    $G := X$;
    $\mathcal{T} := \emptyset$;
    $\mathcal{J} := \emptyset$;
    while $G \neq \emptyset$
        begin
            $T := \emptyset$;
            $T_s := \emptyset$;
            $T_n := \emptyset$;
            $T(G) := \{t \mid [t] \cap G \neq \emptyset\}$;
            while ($T = \emptyset$ or $[T] \not\subseteq X$) and $T(G) \neq \emptyset$
                begin
                    select a pair $t = (a_t, v_t) \in T(G)$ such that
                    $|[t] \cap G|$ is maximum; if a tie occurs, select
```

```
                        a pair $t \in T(G)$ with the smallest cardinality
                        of $[t]$; if another tie occurs, select first pair;
                        $T := T \cup \{t\}$;
                        $G := [t] \cap G$;
                        $T(G) := \{t \mid [t] \cap G \neq \emptyset\}$;
                        **if** $a_t$ is symbolic $\{$let $V_{a_t}$ be the domain of $a_t\}$
                          **then**
                              $T_s := T_s \cup \{(a_t, v) \mid v \in V_{a_t}\}$
                          **else** $\{a_t$ is numerical, let $t = (a_t, u..v)\}$
                              $T_n := T_n \cup \{(a_t, x..y) \mid$ disjoint $x..y$
                               and $u..v\} \cup \{(a_t, x..y) \mid x..y \supseteq u..v\}$;
                        $T(G) := T(G) - (T_s \cup T_n)$;
                   **end** {while};
              **if** $[T] \subseteq X$
                   **then**
                        **begin**
                             **for** each numerical attribute $a_t$ with
                             $(a_t, u..v) \in T$ **do**
                                 **while** ($T$ contains at least two
                                 different pairs $(a_t, u..v)$ and $(a_t, x..y)$
                                 with the same numerical attribute $a_t$)
                                     replace these two pairs with
                                     a new pair ($a_t$, common part of
                                     $u..v$ and $x..y$);
                             **for** each $t$ in $T$ **do**
                                 **if** $[T - \{t\}] \subseteq$ X **then** $T := T - \{t\}$;
                             $\mathcal{T} := \mathcal{T} \cup \{T\}$;
                        **end** {then}
                   **else** $\mathcal{J} := \mathcal{J} \cup \{T\}$;
              $G := X - \cup_{S \in \mathcal{T} \cup \mathcal{J}} [S]$;
         **end** {while};
         **for** each $T \in \mathcal{T}$ **do**
              **if** $\bigcup_{S \in \mathcal{T} - \{T\}} [S] = \bigcup_{S \in \mathcal{T}} [S]$ **then** $\mathcal{T} := \mathcal{T} - \{T\}$;
**end** {procedure}.
```

Note that for a local lower covering $\mathcal{T}$ of X, the set $\bigcup_{S \in \mathcal{T}} [S]$ is a subset of X and it is locally definable, however it does not need to be the A- local lower approximation of X (excluding complete decision tables).

A set $\mathcal{T}$ will be called a *local upper covering* of X if and only if the following three conditions are satisfied:

(1) $X \subseteq \bigcup_{T \in \mathcal{T}} [T]$,

(2) every T is minimal, i.e., no proper subset T' of T exists with $[T'] \subseteq \bigcup_{T \in \mathcal{T}} [T]$,

(3) $\mathcal{T}$ is minimal, i.e., for every $T \in \mathcal{T}$, $X \not\subseteq \bigcup_{S \in \mathcal{T}-\{T\}}[S]$.

The modified procedure for determining a single local upper covering is presented below.

```
Procedure for determining a single local upper covering
input: a set $X$ (a subset of $U$),
output: a single local upper covering $\mathcal{T}$ of the set $X$,
begin
        $G := X$;
        $D := X$;
        $\mathcal{T} := \emptyset$;
        while $G \neq \emptyset$
                begin
                        $T := \emptyset$;
                        $T_s := \emptyset$;
                        $T_n := \emptyset$;
                        $T(G) := \{t \mid [t] \cap G \neq \emptyset\}$;
                        while ($T = \emptyset$ or $[T] \not\subseteq D$) and $T(G) \neq \emptyset$
                                begin
                                select a pair $t = (a_t, v_t) \in T(G)$ such that
                                $|[t] \cap G|$ is maximum; if a tie occurs, select
                                a pair $t \in T(G)$ with the smallest cardinality
                                of $[t]$; if another tie occurs, select first pair;
                                $T := T \cup \{t\}$;
                                $G := [t] \cap G$;
                                $T(G) := \{t \mid [t] \cap G \neq \emptyset\}$;
                                if $a_t$ is symbolic {let $V_{a_t}$ be the domain of $a_t$}
                                   then
                                        $T_s := T_s \cup \{(a_t, v) \mid v \in V_{a_t}\}$
                                   else {$a_t$ is numerical, let $t = (a_t, u..v)$}
                                        $T_n := T_n \cup \{(a_t, x..y) \mid$ disjoint $x..y$
                                         and $u..v\} \cup \{(a_t, x..y) \mid x..y \supseteq u..v\}$;
                                $T(G) := T(G) - (T_s \cup T_n)$;
                        end {while};
                        $D := D \cup [T]$;
                        $\mathcal{T} := \mathcal{T} \cup \{T\}$;
                        $G := D - \cup_{S \in \mathcal{T}}[S]$;
        end {while};
for each $T \in \mathcal{T}$ do
        for each numerical attribute $a_t$ with $(a_t, u..v) \in T$ do
            while ($T$ contains at least two different
            pairs $(a_t, u..v)$ and $(a_t, x..y)$ with
            the same numerical attribute $a_t$)
                replace these two pairs with a new pair
                ($a_t$, common part of $(u..v)$ and $(x..y)$);
        for each $t \in T$ do
            if $[T - \{t\}] \subseteq D$ then $T := T - \{t\}$;
        for each $T \in \mathcal{T}$ do
            if $\bigcup_{S \in \mathcal{T} - \{T\}} [S] \supseteq X$ then $\mathcal{T} := \mathcal{T} - \{T\}$;
end {procedure}.
```

For a local upper covering $\mathcal{T}$ of X, the set $\bigcup_{S \in \mathcal{T}} [S]$ is a superset of X and it is locally definable, however it does not need to be the A-local upper approximation of X (excluding complete decision tables).

Comments to Table 3

1. $\{1, 3, 4, 5, 8\} \not\subseteq \{1, 3, 5\}$, search for the next t,

2. $\{1, 3, 4, 5, 8\} \cap \{1, 2, 3\} = \{1, 3\} \subseteq \{1, 3, 5\}$, so {(*Headache*, *yes*), (*Temperature*, *normal*)} is the first element T of $\mathcal{T}$,

Comments to Table 4

3. $\{2, 3, 5\} \not\subseteq \{1, 3, 5\}$, search for the next t,

4. $\{2, 3, 5\} \cap \{2, 5, 6, 7, 8\} = \{2, 5\} \not\subseteq \{1, 3, 5\}$, search for the next t,

5. $\{2, 3, 5\} \cap \{2, 5, 6, 7, 8\} \cap \{1, 3, 4, 5, 8\} = \{5\} \subseteq \{1, 3, 5\}$, so {(*Cough*, *no*), (*Temperature*, *high*), (*Headache*, *yes*)} is the second element T of $\mathcal{T}$.

Table 3 Computing a local lower covering for the concept lower approximation of [(*Flu*, *yes*)], Table 1, I

$(a, v) = t$	$[(a, v)]$	{1, 3, 5}	{1, 3, 5}
$(Temperature, normal)$	{1, 2, 3}	{1, 3}	{1, 3} ●
$(Temperature, high)$	{2, 5, 6, 7, 8}	{5}	{5}
$(Headache, yes)$	{1, 3, 4, 5, 8}	{1, 3, 5} ●	–
$(Headache, no)$	{2, 6}	–	–
$(Cough, yes)$	{1, 2, 4, 5, 6, 7}	{1, 5}	{1, 5}
$(Cough, no)$	{2, 3, 5}	{3, 5}	{3, 5}
Comments		1	2

Table 4 Computing a local lower covering for the concept lower approximation of [(*Flu*, *yes*)], Table 1, II

$(a, v) = t$	{5}	{5}	{5}
$(Temperature, normal)$	–	–	–
$(Temperature, high)$	{5}	{5} ●	–
$(Headache, yes)$	{5}	{5}	{5} ●
$(Headache, no)$	–	–	–
$(Cough, yes)$	{5}	–	–
$(Cough, no)$	{5} ●	–	–
Comments	3	4	5

4.5 Global MLEM2

Both MLEM2 modules, global and local, are parts of the data system LERS [26, 27]. LERS uses rough set theory introduced by Z. Pawlak in 1982 [14, 19].

Rules induced from the lower approximation of the concept *certainly* describe the concept, hence such rules are called *certain* [28]. On the other hand, rules induced from the upper approximation of the concept describe the concept *possibly*, so these rules are called *possible* [28].

The procedure for determining a single local lower covering may be used for rule induction using lower and upper approximations defined in Section 4.4. The corresponding MLEM2 algorithm is called *global* since the most suitable type of approximations, *concept*, is globally definable. Both certain and possible rule sets may be induced this way. If the input set X is a lower approximation of some concept, the rule set induced this way is *certain*, if the input set X is an upper approximation of the concept, the corresponding rule set is *possible*.

We will illustrate this idea with inducing the certain rule set from Table 1 using the concept lower approximation. The input set X is {1, 3, 5}, the concept lower approximation of the concept [(*Flu*, *yes*)], computed in Section 4.4. The process of tracing the procedure for determining a single local covering is presented in Tables 3 and 4. In these tables consecutive goals G are listed in the upper rows. The first such goal G is the set {1, 3, 5}. The most relevant attribute value pair t, such that $|[t] \cap G|$ is maximum, is (*Headache*, *yes*), see comment 1. The corresponding entry {1, 3, 5} from this column is marked by a bullet. As mentioned in comment 1, we need to look for the next attribute-value pair. There are three candidates for which $|[t] \cap G|$ is maximum: (*Temperature*, *normal*), (*Cough*, *yes*) and (*Cough*, *no*). For two of them, (*Temperature*, *normal*) and (*Cough*, *no*) the second criterion of selecting t, the smallest cardinality of $[t]$, there is still a tie. Therefore, the use the third criterion and we select the first pair, (the top pair): (*Temperature*, *normal*). The corresponding entry, the set {1, 3}, the intersection of {1, 2, 3} and {1, 3, 5}, is bulleted in the rightmost column of Table 3.

For Table 1, certain rules induced by global MLEM2, using concept approximations, are:

```
2, 2, 2
(Headache, yes) & (Temperature, normal) -> (Flu, yes)
2, 2, 2
(Cough, no) & (Headache, yes) -> (Flu, yes)
1, 2, 2
(Headache, no) -> (Flu, no)
```

Possible rules, induced in the same way, are:

```
2, 2, 4
(Cough, yes) & (Temperature, high) -> (Flu, yes)
1, 2, 3
(Temperature, normal) -> (Flu, yes)
1, 2, 2
(Headache, no) -> (Flu, no)
2, 1, 3
(Headache, yes) & (Cough, yes) -> (Flu, no)
2, 1, 2
(Temperature, high) & (Headache, yes) -> (Flu, no)
```

The above rules are in the LERS format (every rule is equipped with three numbers, the total number of attribute-value pairs on the left-hand side of the rule, the total number of cases correctly classified by the rule during training, and the total number of training cases matching the left-hand side of the rule), see, e.g., [3].

4.6 Local MLEM2

If we want to induce certain and possible *local* rule sets, we should use the procedures for determining single local lower and upper coverings, respectively. The input data sets are the concepts.

Let us compute certain local rules for the concept [(*Flu*, *yes*)] = {1, 3, 5, 7}. The execution of the procedure for determining a single local lower covering is traced in Tables 5 and 6. Again, the consecutive goals G are listed in the top rows of Tables 5 and 6. As follows from comments 1–7, the corresponding local lower covering is $\mathcal{T}$ = {{(*Headache*, *yes*), (*Temperature*, *normal*)}, {(*Temperature*, *high*), (*Cough*, *yes*), (*Headache*, *yes*)}}. We may compute a local lower covering for the concept [(*Flu*, *no*)] in a similar way.

Tracing of the execution of the procedure for determining a single upper covering can be presented in Tables 5 and 6 as well, the only difference is in the comment 7, this time the set T = {*Temperature*, *high*), (*Cough*, *yes*)} does not go the set $\mathcal{J}$ (which does not exist in this procedure) but to the set $\mathcal{T}$. Thus, at the end, $\mathcal{T}$ = {{(*Headache*, *yes*), (*Temperature*, *normal*)}, {(*Temperature*, *high*), (*Cough*, *yes*), (*Headache*, *yes*)}, {(*Temperature*, *high*), (*Cough*, *yes*)}} and the set D = {1, 2, 3, 5, 6, 7}. In our example, the set D is equal to the concept lower approximation of [(*Flu*, *yes*)] (in general, these two sets are different).

The first **for** loop (**for** each $T \in \mathcal{T}$) is designed to simplify each T (reduce the number of the corresponding rule conditions). The first T is {(*Headache*, *yes*), (*Temperature*, *normal*)}. Obviously, the first attribute-value pair is redundant, since

$$[(Temperature, normal)] = \{1, 2, 3\} \subseteq \{1, 2, 3, 5, 6, 7\} = D.$$

In the second T = {(*Temperature*, *high*), (*Cough*, *yes*), (*Headache*, *yes*)} the third attribute-value pair is redundant since

$$[(Temperature, high), (Cough, yes)] = \{2, 5, 6, 7\} \subseteq \{1, 2, 3, 5, 6, 7\} = D.$$

The third T = {(*Temperature*, *high*), (*Cough*, *yes*) cannot be simplified.

The second **for** loop (**for** each $T \in \mathcal{T}$) is designed to eliminate redundant Ts. The second T = {(*Temperature*, *high*), (*Cough*, *yes*)} is redundant since

$$\begin{aligned}&[(Temperature, normal)] \cup [(Temperature, high), (Cough, yes)] = \\ &= \{1, 2, 3\} \cup \{2, 5, 6, 7\} = \{1, 2, 3, 5, 6, 7\} \supseteq X = \{1, 3, 5, 7\}.\end{aligned}$$

The corresponding set of certain rules is:

```
2, 2, 2
(Headache, yes) & (Temperature, normal) -> (Flu, yes)
1, 2, 2
(Headache, no) -> (Flu, no)
```

Possible rule set, induced in the same way, is:

```
1, 2, 3
(Temperature, normal) -> (Flu, yes)
2, 2, 4
(Temperature, high) & (Cough, yes) -> (Flu, yes)
1, 2, 2
(Headache, no) -> (Flu, no)
2, 1, 2
(Headache, yes) & (Temperature, high) -> (Flu, no)
2, 1, 3
(Headache, yes) & (Cough, yes) -> (Flu, no)
```

Comments to Table 5

1. {1, 3, 4, 5, 8} $\not\subseteq$ {1, 3, 5, 7}, search for the next t,
2. {1, 3, 4, 5, 8} ∩ {1, 2, 3} = {1, 3} ⊆ {1, 3, 5, 7}, so {(*Headache*, *yes*), (*Temperature*, *normal*)} is the first element T of $\mathcal{T}$,

Table 5 Computing a single local lower covering for the concept [(*Flu*, *yes*)], Table 1, I

$(a, v) = t$	$[(a, v)]$	{1, 3, 5, 7}	{1, 3, 5}
$(Temperature, normal)$	{1, 2, 3}	{1, 3}	{1, 3} •
$(Temperature, high)$	{2, 5, 6, 7, 8}	{5, 7}	{5}
$(Headache, yes)$	{1, 3, 4, 5, 8}	{1, 3, 5} •	–
$(Headache, no)$	{2, 6}	–	–
$(Cough, yes)$	{1, 2, 4, 5, 6, 7}	{1, 5, 7}	{1, 5}
$(Cough, no)$	{2, 3, 5}	{3, 5}	{3, 5}
Comments		1	2

Comments to Table 6

3. {2, 5, 6, 7, 8} $\not\subseteq$ {1, 3, 5, 7}, search for the next t,

4. {2, 5, 6, 7, 8} ∩ {1, 2, 4, 5, 6, 7} = {2, 5, 6, 7} $\not\subseteq$ {1, 3, 5, 7}, search for the next t,

5. {2, 5, 6, 7, 8} ∩ {1, 2, 4, 5, 6, 7} ∩ {1, 3, 4, 5, 8} = {5} ⊆ {1, ,3, 5, 7}, so {(*Temperature*, *high*), (*Cough*, *yes*), (*Headache*, *yes*)} is the second element T of $\mathcal{T}$,

6. {2, 5, 6, 7, 8} $\not\subseteq$ {1, 3, 5, 7}, search for the next t,

7. {2, 5, 6, 7, 8} ∩ {1, 2, 4, 5, 6, 7} = {2, 5, 6, 7} $\not\subseteq$ {1, 3, 5, 7}, no more Ts, so {7} goes to $\mathcal{J}$.

Table 6 Computing a single local lower covering for the concept [(*Flu*, *yes*)], Table 1, II

$(a, v) = t$	{5, 7}	{5, 7}	{5, 7}	{7}	{7}
$(Temperature, normal)$	–	–	–	–	–
$(Temperature, high)$	{5, 7} •	–	–	{7} •	–
$(Headache, yes)$	{5}	{5}	{5} •	–	–
$(Headache, no)$	–	–	–	–	–
$(Cough, yes)$	{5, 7}	{5, 7} •	–	{7}	{7} •
$(Cough, no)$	{5}	{5}	–	–	–
Comments	3	4	5	6	7

4.7 Incomplete Data Sets with Numerical Attributes

The attribute *Temperature* from Table 7 is numerical. For a numerical attribute, the first step is to sort numerical attribute values. For *Temperature* the list of sorted values is 98.4, 98.8, 101.4 and 102.0. The next step is to select cutpoints. In MLEM2, the potential cutpoints are averages of consecutive values of the sorted list of all attribute values. In our example, such potential cutpoints are 98.6, 100.1, and 101.7. Thus, the potential intervals are, e.g., 98.4..100.1 and 100.1..102.0. In the current, local MLEM2 algorithm, there are two options of selecting potential cutpoints: *all cutpoints* and *selected cutpoints*.

If we use the option *all cutpoints*, for every potential cutpoint the MLEM2 algorithm creates two *primary intervals*, the first containing all numerical values smaller than the cutpoint and the second containing all numerical values greater than the cutpoint. Thus, for Table 7, the list of all primary intervals is 98.4..98.6, 98.6..102.0, 98.4..100.1, 100.1..102.0, 98.4..101.7, 101.7..102.0. The first interval, 98.4..98.6 contains just one value: 98.4, the second interval, 98.4..102.0 contains values 98.8, 101.4, and 102.0. In different words, the first interval is *represented* by the set {98.4}, and the second interval is represented by the set {98.8, 101.4, 102.0}. The all cutpoints option is the only option of the MLEM2 global version and one of two options of the MLEM2 local version. The other option of the MLEM2 local version is *selected cutpoints*.

In the all cutpoints option of MLEM2, the decision is not taken into account, while in the selected cutpoints option of MLEM2 the algorithm chooses only some selected cutpoints on the basis of the corresponding decision values. In general, if for all occurrences of the two consecutive values of the sorted list of values of a numerical attribute the decision value is the same, the corresponding cutpoint is ignored in creating primary intervals. In Table 7, there are unique values of 98.4 and 98.8, for both the decision value is the same (*yes*), so the potential cutpoint 98.6 is ignored. On the other hand, the decision values for 101.4 is *yes*, while the decision value for 102.0 is *no*, so we cannot ignore the cutpoint 101.7. Thus, the only selected cutpoint is 101.7, and the primary intervals are 98.4..101.7 and 101.7..102.0. We are following here the principle *the cutpoint will always occur on the boundary between two classes* [29], though this principle is valid only for cutpoints selected using entropy minimization.

In the MLEM2 algorithms, both versions, global and local, some operations are performed on intervals. The MLEM2 algorithm may select intervals associated with the same attribute as conditions of a rule. Such intervals are eventually *merged*. For example, let us say that MLEM2 selected two intervals of the same attribute *Temperature*, the first one is 98.4..101.7 and the second is 98.6..102.0. These two intervals will be merged into 98.6..101.7. The first interval is represented by the set {98.4, 98.8, 101.4}, the second interval is represented by {98.8, 101.4, 102.0}, the merged interval is represented by the intersection of sets represented by both intervals, i.e., by the set

Table 7 An incomplete decision table

	Attributes			Decision
Case	Temperature	Headache	Cough	Flu
1	98.4	yes	yes	yes
2	*	no	*	no
3	98.8	–	no	yes
4	?	yes	yes	no
5	101.4	yes	+	yes
6	–	no	yes	no
7	101.4	?	yes	yes
8	102.0	yes	?	no

{98.8, 101.4}. The interval 98.6..101.7 will be called a *common part* of 98.4..101.7 and 98.6..102.0.

Two intervals with the common part equal to the empty set are called *disjoint*. For example, intervals 98.4..101.7 and 101.7..102.0 are disjoint since the first interval is represented by {98.4, 98.8, 101.4} and the second interval is represented by {102.0}.

We say that an interval *includes* another interval if it is represented by a set that is a superset of the other interval. For example, 98.6..102.0 includes 100.1..102.0 since the first interval is represented by the set {98.8, 101.4, 102.0} and the second interval is represented by the set {101.4, 102.0}. We will denote it by 98.6..102.0 $\supseteq$ 100.1..102.0.

For computing characteristic sets numerical attributes are treated as symbolic. Additionally, $V(6, Temperature) = \{102.0\}$ Thus, for Table 7 the blocks of attribute-value pairs are:

[(Temperature, 98.4)] = {1, 2},
[(Temperature, 98.8)] = {2, 3},
[(Temperature, 101.4)] = {2, 5, 7},
[(Temperature, 102.0)] = {2, 6, 8},

Blocks of attribute-value pairs for the attributes *Headache* and *Cough* were computed previously. The characteristic sets for Table 7 are:

$K_A(1) = \{1, 2\} \cap \{1, 3, 4, 5, 8\} \cap \{1, 2, 4, 5, 6, 7\} = \{1\}$
$K_A(2) = U \cap \{2, 6\} \cap U = \{2, 6\}$,
$K_A(3) = \{2, 3\} \cap \{1, 3, 4, 5, 8\} \cap \{2, 3, 5\} = \{3\}$,
$K_A(4) = U \cap \{1, 3, 4, 5, 8\} \cap \{1, 2, 4, 5, 6, 7\} = \{1, 4, 5\}$,
$K_A(5) = \{2, 5, 7\} \cap \{1, 3, 4, 5, 8\} \cap (\{1, 2, 4, 5, 6, 7\} \cup \{2, 3, 5\}) = \{5\}$,
$K_A(6) = \{2, 6, 8\} \cap \{2, 6\} \cap \{1, 2, 4, 5, 6, 7\} = \{2, 6\}$,
$K_A(7) = \{2, 5, 7\} \cap U \cap \{1, 2, 4, 5, 6, 7\} = \{2, 5, 7\}$.
$K_A(8) = \{2, 6, 8\} \cap \{1, 3, 4, 5, 8\} \cup U = \{8\}$.

The characteristic relation R is {(1, 1), (2, 2), (2, 6), (3,3), (4, 1), (4, 4), (4, 5), (5, 5), (6, 2), (6, 6), (7, 2), (7, 5), (7, 7), (8, 8)}.

The singleton approximations are:

$$\underline{A}\{1,3,5,7\} = \{1,3,5\},$$

$$\underline{A}\{2,4,6,8\} = \{2,6,8\},$$

$$\overline{A}\{1,3,5,7\} = \{1,3,4,5,7\},$$

$$\overline{A}\{2,4,6,8\} = \{2,4,6,7,8\}.$$

The subset approximations are

$$\underline{A}\{1,3,5,7\} = \{1,3,5\},$$

$$\underline{A}\{2,4,6,8\} = \{2,6,8\},$$

$$\overline{A}\{1,3,5,7\} = \{1,2,3,4,5,7\},$$

$$\overline{A}\{2,4,6,8\} = \{1,2,4,5,6,7,8\}.$$

In addition, the concept approximations are

$$\underline{A}\{1,3,5,7\} = \{1,3,5\},$$

$$\underline{A}\{2,4,6,8\} = \{2,6,8\},$$

$$\overline{A}\{1,3,5,7\} = \{1,2,3,5,7\},$$

$$\overline{A}\{2,4,6,8\} = \{1,2,4,5,6,8\}.$$

Tables 8 and 9 present tracing the procedure for determining a single local lower covering for the concept [(*Flu*, *yes*)] from Table 7. Since we are applying the procedure for determining a single local covering directly to the concept, we are using the local MLEM2. Here $\mathcal{T}$ = {{(*Temperature*, 98.4..101.7), (*Headache*, *yes*)}}. Similarly, we may induce a local lower covering for the second concept and upper local coverings for both concepts. The induced local rule sets are:
certain rule set

```
2, 3, 3
(Temperature, 98.4..101.7) & (Headache, yes) -> (Flu, yes)
1, 2, 4
(Temperature, 101.7..102) -> (Flu, no)
```

possible rule set

```
1, 4, 5
(Temperature, 98.4..101.7) -> (Flu, yes)
1, 1, 2
```

(Temperature, 98.4..98.6) -> (Flu, no)
1, 3, 3
(Temperature, 101.7..102.0) -> (Flu, no)
2, 1, 3
(Headache, yes) & (Cough, yes) -> (Flu, no)

Table 8 Computing a single local lower covering for the concept [(*Flu*, *yes*)], numerical data, Table 7, I

$(a, v) = t$	$[(a, v)]$	{1, 3, 5, 7}	{1, 3, 5, 7}
$(Temperature, 98.4..98.6)$	{1, 2}	{1}	{1}
$(Temperature, 98.6..102)$	{2, 3, 5, 6, 7, 8}	{3, 5, 7}	{3, 5, 7}
$(Temperature, 98.4..100.1)$	{1, 2, 3}	{1, 3}	{1, 3}
$(Temperature, 100.1..102)$	{2, 5, 6, 7, 8}	{5, 7}	{5, 7}
$(Temperature, 98.4..101.7)$	{1, 2, 3, 5, 7}	{1, 3, 5, 7 } ●	–
$(Temperature, 101.7..102)$	{2, 6, 8}	–	–
$(Headache, yes)$	{1, 3, 4, 5, 8}	{1, 3, 5}	{1, 3, 5} ●
$(Headache, no)$	{2, 6}	–	–
$(Cough, yes)$	{1, 2, 4, 5, 6, 7}	{1, 5, 7}	{1, 5, 7}
$(Cough, no)$	{2, 3, 5}	{3, 5}	{3, 5}
Comments		1	2

Table 9 Computing a single local lower covering for the concept [(*Flu*, *yes*)], numerical data, Table 7, II

$(a, v) = t$	{7}	{ 7}	{7}
$(Temperature, 98.4..98.6)$	–	–	–
$(Temperature, 98.6..102)$	{7}	–	–
$(Temperature, 98.4..100.1)$	–	–	–
$(Temperature, 100.1..102)$	{7} ●	–	–
$(Temperature, 98.4..101.7)$	{7}	{7} ●	–
$(Temperature, 101.7..102)$	–	–	–
$(Headache, yes)$	–	–	–
$(Headache, no)$	–	–	–
$(Cough, yes)$	{7}	{7}	{7} ●
$(Cough, no)$	–	–	–
Comments	3	4	5

Comments to Table 8

1. $\{1, 2, 3, 5, 7\} \nsubseteq \{1, 3, 5, 7\}$, search for the next t,

2. $\{1, 2, 3, 5, 7\} \cap \{1, 3, 4, 5, 8\} = \{1, 3, 5\} \subseteq \{1, 3, 5, 7\}$, {(*Temperature*, 98.4..101.7), (*Headache*, *yes*)} is the first element T of $\mathcal{T}$,

Comments to Table 9

3. $\{2, 5, 6, 7, 8\} \nsubseteq \{1, 3, 5, 7\}$, search for the next t,

4. $\{2, 5, 6, 7, 8\} \cap \{1, 2, 3, 5, 7\} = \{2, 5, 7\} \nsubseteq \{1, 3, 5, 7\}$, search for the next t,

5. $\{2, 5, 6, 7, 8\} \cap \{1, 2, 3, 5, 7\} \cap \{1, 2, 4, 5, 6, 7\} = \{2, 5, 7\} \nsubseteq \{1, 3, 5, 7\}$, so $\{7\}$ goes to $\mathcal{J}$.

4.8 Experiments

We used five data sets for our experiments, see Table 10. All of these data sets, except *bankruptcy*, are well-known data accessible at the University of California at Irvine Data Depository, http://archive.ics.uci.edu/ml/. The data set *bankruptcy* was collected by E. Altman and M. Heine at the New York University, School of Business, in 1968. In all five data sets, some attribute values, about 30%, were randomly removed, or more exactly, the original attribute values were replaced by symbols of missing attribute values. Three data sets: *hepatitis*, *image segmentation* and *wine* were discretized using a discretization method based on agglomerative cluster analysis, [30].

Results of experiments are presented in Tables 11 and 12. Results of our experiments show that both versions of MLEM2, global and local, are incomparable. For two data sets (*breast cancer* and *lymphography*) global MLEM2 induces rule sets with smaller error rate, for two data sets (*image segmentation* and *wine*) local MLEM2 is better, for *hepatitis* comparing the performance ends up with a tie. For any specific data set it is a good idea to try both approaches and select the better.

Table 10 Data sets used for experiments

Data set	Number of			Type of
	cases	attributes	concepts	attributes
Breast cancer	277	9	2	symbolic
Hepatitis	155	19	2	discretized
Image segmentation	210	19	7	discretized
Lymphography	148	18	4	symbolic
Wine	178	13	3	discretized

Table 11 Error rates global MLEM2

Data set	Certain rule sets		Possible rule sets	
	Type of missing attribute values			
	?	*	?	*
Breast cancer	29.96%	27.80%	28.52%	29.60%
Hepatitis	18.06%	21.94%	18.06%	21.94%
Image segmentation	42.86%	83.33%	46.67%	43.81%
Lymphography	24.32%	33.11%	20.95%	22.97%
Wine	16.85%	52.25%	17.42%	22.47%

Table 12 Error rates local MLEM2

Data set	Certain rule sets		Possible rule sets	
	Type of missing attribute values			
	?	*	?	*
Breast cancer	28.52%	30.69%	29.96%	29.24%
Hepatitis	18.71%	20.00%	18.06%	18.71%
Image segmentation	47.62%	57.62%	47.14%	40.00%
Lymphography	23.65%	29.73%	21.62%	25.68%
Wine	15.17%	23.03%	15.17%	13.48%

4.9 Conclusions

A rough set approach to missing attribute values provide the means for different interpretations of incompleteness, depending on real-life situations. It is possible to interpret a missing attribute value as *lost values*, *"do not care" conditions*, *restricted "do not care" conditions*, and *attribute-concept values.* In most other approaches to missing attribute values such differentiation between interpretations is unavailable.

Additionally, we may use two different approaches for rule induction: *global*, in which we compute approximations first, and *local*, where everything, starting from handling missing attribute values, numerical attributes, and approximations is done during the same process of rule induction. Both approaches, global and local, are worth consideration. In practice, for a specific data set, we should select the appropriate approach experimentally.

A choice between using certain and possible rules seems to be not important.

Experimental comparison of probabilistic and rough-set approaches to mining data with missing attribute values [31] shows that there is no significant difference in performance, for any specific data set the best method to handle missing attribute values should be again selected individually.

Acknowledgements. The authors would like to thank the anonymous referee for many valuable comments.

References

1. Grzymala-Busse, J.W., Grzymala-Busse, W.J.: Handling missing attribute values. In: Maimon, O., Rokach, L. (eds.) Data Mining and Knowledge Discovery Handbook, pp. 37–57. Springer, Heidelberg (2005)
2. Grzymala-Busse, J.W., Grzymala-Busse, W.J., Goodwin, L.K.: A comparison of three closest fit approaches to missing attribute values in preterm birth data. International Journal of Intelligent Systems 17(2), 125–134 (2002)
3. Grzymala-Busse, J.W.: MLEM2: A new algorithm for rule induction from imperfect data. In: Proceedings of the 9th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems, pp. 243–250 (2002)
4. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers, San Mateo (1993)
5. Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and Regression Trees. Wadsworth & Brooks, Monterey (1984)
6. Grzymala-Busse, J.W., Wang, A.Y.: Modified algorithms LEM1 and LEM2 for rule induction from data with missing attribute values. In: Proceedings of the Fifth International Workshop on Rough Sets and Soft Computing (RSSC 1997) at the Third Joint Conference on Information Sciences (JCIS 1997), pp. 69–72 (1997)
7. Stefanowski, J., Tsoukiàs, A.: On the Extension of Rough Sets under Incomplete Information. In: Zhong, N., Skowron, A., Ohsuga, S. (eds.) RSFDGrC 1999. LNCS (LNAI), vol. 1711, pp. 73–82. Springer, Heidelberg (1999)
8. Stefanowski, J., Tsoukias, A.: Incomplete information tables and rough classification. Computational Intelligence 17(3), 545–566 (2001)
9. Grzymala-Busse, J.W.: On the unknown attribute values in learning from examples. In: Proceedings of the ISMIS 1991, 6th International Symposium on Methodologies for Intelligent Systems, pp. 368–377 (1991)
10. Kryszkiewicz, M.: Rough set approach to incomplete information systems. In: Proceedings of the Second Annual Joint Conference on Information Sciences, pp. 194–197 (1995)
11. Kryszkiewicz, M.: Rules in incomplete information systems. Information Sciences 113(3-4), 271–292 (1999)
12. Grzymala-Busse, J.W.: Three approaches to missing attribute values—a rough set perspective. In: Proceedings of the Workshop on Foundation of Data Mining, in Conjunction with the Fourth IEEE International Conference on Data Mining, pp. 55–62 (2004)

13. Grzymala-Busse, J.W.: Incomplete data and generalization of indiscernibility relation, definability, and approximations. In: Proceedings of the 10th International Conference on Rough Sets, Fuzzy Sets, Data Mining, and Granular Computing, pp. 244–253 (2005)
14. Pawlak, Z.: Rough Sets. Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Dordrecht (1991)
15. Li, T., Ruan, D., Geert, W., Song, J., Xu, Y.: A rough sets based characteristic relation approach for dynamic attribute generalization in data mining. Knowledge-Based Systems 20, 485–494 (2007)
16. Qi, Y.S., Wei, L., Sun, H.J., Song, Y.Q., Sun, Q.S.: Characteristic relations in generalized incomplete information systems. In: International Workshop on Knowledge Discovery and Data Mining, pp. 519–523 (2008)
17. Song, J., Li, T., Ruan, D.: A new decision tree construction using the cloud transform and rough sets. In: Proceedings of the Rough Sets and Knowledge Technology Conference, pp. 524–531 (2008)
18. Yang, X.B., Yang, J.Y., Wu, C., Yu, D.J.: Further investigation of characteristic relation in incomplete information systems. Systems Engineering - Theory & Practice 27(6), 155–160 (2007)
19. Pawlak, Z.: Rough sets. International Journal of Computer and Information Sciences 11, 341–356 (1982)
20. Grzymala-Busse, J.W.: Rough set strategies to data with missing attribute values. In: Workshop Notes, Foundations and New Directions of Data Mining, in Conjunction with the 3rd International Conference on Data Mining, pp. 56–63 (2003)
21. Grzymała-Busse, J.W.: Data With Missing Attribute Values: Generalization of Indiscernibility Relation and Rule Induction. In: Peters, J.F., Skowron, A., Grzymała-Busse, J.W., Kostek, B.z., Świniarski, R.W., Szczuka, M.S. (eds.) Transactions on Rough Sets I. LNCS, vol. 3100, pp. 78–95. Springer, Heidelberg (2004)
22. Grzymala-Busse, J.W.: Characteristic relations for incomplete data: A generalization of the indiscernibility relation. In: Proceedings of the Fourth International Conference on Rough Sets and Current Trends in Computing, pp. 244–253 (2004)
23. Lin, T.Y.: Topological and fuzzy rough sets. In: Slowinski, R. (ed.) Intelligent Decision Support. Handbook of Applications and Advances of the Rough Sets Theory, pp. 287–304. Kluwer Academic Publishers, Dordrecht (1992)
24. Slowinski, R., Vanderpooten, D.: A generalized definition of rough approximations based on similarity. IEEE Transactions on Knowledge and Data Engineering 12, 331–336 (2000)
25. Grzymala-Busse, J.W., Rzasa, W.: Local and Global Approximations for Incomplete Data. In: Greco, S., Hata, Y., Hirano, S., Inuiguchi, M., Miyamoto, S., Nguyen, H.S., Słowiński, R. (eds.) RSCTC 2006. LNCS (LNAI), vol. 4259, pp. 244–253. Springer, Heidelberg (2006)
26. Grzymala-Busse, J.W.: LERS—a system for learning from examples based on rough sets. In: Slowinski, R. (ed.) Intelligent Decision Support. Handbook of Applications and Advances of the Rough Set Theory, pp. 3–18. Kluwer Academic Publishers, Dordrecht (1992)
27. Grzymala-Busse, J.W.: A new version of the rule induction system LERS. Fundamenta Informaticae 31, 27–39 (1997)

28. Grzymala-Busse, J.W.: Knowledge acquisition under uncertainty—A rough set approach. Journal of Intelligent & Robotic Systems 1, 3–16 (1988)
29. Fayyad, U.M., Irani, K.B.: On the handling of continuous-valued attributes in decision tree generation. Machine Learning 8, 87–102 (1992)
30. Chmielewski, M.R., Grzymala-Busse, J.W.: Global discretization of continuous attributes as preprocessing for machine learning. International Journal of Approximate Reasoning 15(4), 319–331 (1996)
31. Grzymala-Busse, J.W.: Mining data with missing attribute values: A comparison of probabilistic and rough set approaches. In: Proceedings of the 4th International Conference on Intelligent Systems and Knowledge Engineering, pp. 153–158 (2009)

Chapter 5
Roles Played by Bayesian Networks in Machine Learning: An Empirical Investigation

Estevam R. Hruschka Jr. and Maria do Carmo Nicoletti

Abstract. Bayesian networks (BN) and Bayesian classifiers (BC) are traditional probabilistic techniques that have been successfully used by various machine learning methods to help solving a variety of problems in many different domains. BNs (and BCs) can be considered a probabilistic graphical language suitable for inducing models from data aiming at knowledge representation and reasoning about data domains. The main goal of this chapter is the empirical investigation of a few roles played by BCs in machine learning related processes namely (i) data pre-processing (feature selection and imputation), (ii) learning and (iii) post-processing (rule generation). By doing so the chapter contributes with organizing, specifying and discussing the many different ways Bayes-based concepts can successfully be employed in automatic learning.

5.1 Introduction

Since the beginning of the past decade Bayesian networks (BNs) (also known as belief networks or directed probabilistic graphical models) have been attracting a great deal of attention and have been successfully applied to solve a variety of problems in many different domains, most of them related to modeling and decision under uncertainty. They have been used in domains such as medicine (Díez *et al.* 1997) (Husmeier *et al.* 2005), molecular biology (Friedman 2004) (Sachs *et al.* 2005), genomics (Sebastiani *et al.* 2003) (Friedman *et al.* 2000) (Jansen *et al.* 2003), agricultural (Bressan *et al.* 2009) and many others. An overview of the main applications involving BNs can be seen in (Lauritzen 2003) and more recently in (Pourret *et al.* 2008).

Estevam R. Hruschka Jr. · Maria do Carmo Nicoletti
Computer Science Department, UFSCar, S. Carlos, SP, Brazil
e-mail: `estevam@dc.ufscar.br`

Maria do Carmo Nicoletti
FACCAMP, C. L. Paulista, SP, Brazil
e-mail: `carmo@dc.ufscar.br`

BNs can be considered a probabilistic graphical language for knowledge representation and reasoning. A BN (Pearl 1988) has a DAG (directed acyclic graph) structure. Each node in the graph corresponds to a discrete random variable in the domain. Edges represent conditional dependencies; an edge $Y \rightarrow X$ describes a parent-child relation, where Y is the parent and X is the child. Nodes that are not connected represent variables that are conditionally independent of each other. Each node of the BN structure is associated with a conditional probability table (CPTable) specifying the probability of each possible state of the node, given each possible combination of states of its parents.

A Bayesian classifier (BC) is a particular type of BN that aims at correctly predicting the value of a discrete class variable, given a vector of feature values. As pointed out in (Heckerman *et al.* 2000), BNs and BCs are usually employed in data mining tasks mainly because they (i) may deal with incomplete datasets straightforwardly; (ii) can learn causal relationships; (iii) may combine prior knowledge with patterns learnt from data and (iv) can help to avoid overfitting. Since Bayesian classifiers are a particular type of Bayesian networks, most of the related concepts and results are valid for both.

The main goal of this chapter is to empirically investigate possible roles played by Bayesian classifiers in three main subprocesses of machine learning processes namely: (1) data pre-processing (imputation and feature selection), (2) learning and (3) post-processing (rule generation and pruning). Although the natural order to approach machine learning (ML) subprocesses is the sequential order as stated above, in this chapter the learning of BNs and BCs will be discussed first since algorithms used for learning can be used for pre-processing as well as post-processing the data.

Besides the Introduction, the chapter is organized in six more sections. Section 2 introduces several of the underlying concepts involved in BNs and BCs, focusing on those that are relevant to some of the roles played by Bayesian models discussed in the chapter. Section 3 approaches BNs and BCs as knowledge representations and briefly presents the main ideas of three important algorithms: the Naïve Bayes (Duda and Hart 1973), PC (Spirtes *et al.* 1993) and K2 (Cooper and Herskovits 1992), used for learning BNs and BCs. Sections 4 and 5 address the use of Bayesian Classifiers for modifying the original training data available, aiming at improving its quality. The help provided by BN-based methods will specifically be investigated when the original training data patterns (a) are described by features that might be irrelevant (or superfluous) for the purpose of the learning task at hand (Section 4) and/or (b) have missing feature values, a recurrent and commonly problem found in collected data (Section 5). Section 6 describes in detail how BNs can be post-processed in order to create a set of rules (Hruschka Jr. *et al.* 2008). In Section 7 the main conclusions of the work described in the chapter are summarized.

5.2 Relevant Concepts Related to Bayesian Networks and Bayesian Classifiers

The issues discussed in this chapter are dependent on several concepts used in Bayes theory which, in turn, are heavily dependent on the probability theory. A brief review of the main relevant concepts is presented next. Most of the concepts are defined using the notation borrowed from (Friedman *et al.* 1997) and many can be revisited in Moore´s tutorials (Moore 2011). Consider:

- Lowercase letters denote specific values taken by those variables (e.g. x, y, z)
- Boldface capital letters denote sets of variables (e.g **X**, **Y**, **Z**)
- Boldface lowercase letters (e.g. **x**, **y**, **z**) denote assignments of values to the variables in sets **X**, **Y**, **Z** respectively. (Val(**X**) is used in the obvious way)
- A finite set of discrete random variables $\psi = \{X_1, X_2, \ldots, X_n\}$
- Each variable X_i may take on values from a finite set, denoted by $Val(X_i)$, $i=1,\ldots,n$.
- Capital letters will be used for variable names (e.g. X, Y, Z)

Definition 1. The probability that variable X takes the value x will be denoted P(X = x) (or P(x) when there is no risk of ambiguity). The *joint probability distribution* (JPD) over n random variables $X_1, X_2, \ldots, X_n$ encodes the probability of a particular assignment to all the variables i.e. $P(X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$ or simply $P(x_1, x_2, \ldots, x_n)$ ♦.

Definition 2. The *conditional probability* that a random variable X takes on the value x given some other random variable Y takes on the value y is written P(x|y) and is defined by eq. (1) provided that P(y) > 0 ♦.

$$P(x|y) = \frac{P(x,y)}{P(y)} \tag{1}$$

Eq. (1) can be generalized for a set of random variables $X_1, X_2, \ldots, X_n$ and $Y_1, Y_2, \ldots, Y_m$ as eq. (2), provided that $P(y_1,y_2,\ldots,y_m) > 0$.

$$P(x_1, x_2,\ldots, x_n|y_1, y_2, \ldots, y_m) = \frac{P(x_1, x_2,\ldots, x_n, y_1, y_2,\ldots, y_m)}{P(y_1, y_2,\ldots, y_m)} \tag{2}$$

Using the notation described at the beginning of this section, eq. (2) can be rewritten as eq. (3).

$$P(\mathbf{x}|\mathbf{y}) = \frac{P(\mathbf{x},\mathbf{y})}{P(\mathbf{y})} \tag{3}$$

Two sets of random variables being conditionally independent of a third set is a fundamental concept for establishing a few others concepts as well as a few procedures in a learning environment based on Bayesianism. The concept is formalized in *Definition 3.*

Definition 3. Let P be a joint probability distribution over the variables in ψ and let **X**, **Y**, **Z** be subsets of U. **X** and **Y** are said to be *conditionally independent* given **Z** noted as I(**X**,**Y**|**Z**) if for all $\mathbf{x} \in Val(\mathbf{X})$, $\mathbf{y} \in Val(\mathbf{Y})$, $\mathbf{z} \in Val(\mathbf{Z})$, $P(\mathbf{x}|\mathbf{z},\mathbf{y}) = P(\mathbf{x}|\mathbf{z})$ whenever $P(\mathbf{y},\mathbf{z}) > 0$. ♦

Bayesian networks (BN) belong to the family of probabilistic graphical models (GMs); more specifically, they are represented by a directed acyclic graph (DAG). As mentioned in (Murphy 1998), BNs enable an effective representation and computation of the JPD over a set of random variables. In a DAG the set of parents of a node X is represented by $\pi(X)$. By having the structure of an acyclic graph, it can be guaranteed that there is no node in the BN that can be its own ancestor or its own descendent. Such a condition, as mentioned in (Ben-Gal 2007), is of vital importance to the factorization of the joint probability of a collection of variables (nodes).

The specification of its DAG structure is considered the "qualitative" aspect of a BN. As it will be seen later in the chapter (Section 3.2), the concept of skeleton of a DAG (Definition 4) can be used during its construction. The specification of its "quantitative" aspect is done by specifying the conditional probability distribution at each node i.e., specifying the probability of each possible state of the node, given each possible combination of states of its parents. As pointed out in (Ben-Gal 2007), for discrete random variables, the conditional probability distribution is often represented by a table listing the local probability that the corresponding child node takes on each of its feasible values, for each combination of values of its parents. The joint distribution of a collection of variables can be determined uniquely by these local conditional probability tables (CPTables). Definition 5 gives a formal definition of BN based on the one proposed in (Friedman *et al.* 1997).

Definition 4. Let G be a DAG (directed acyclic graph). The *skeleton* of G is the undirected graph obtained from G by replacing its arcs with undirected edges ♦.

Definition 5. Consider the finite set of discrete random variables $\psi = \{X_1, X_2, \ldots, X_n\}$ where each variable X_i may take on values from a finite set. A Bayesian network for ψ is a pair $B = <G,\Theta>$. G is a directed acyclic graph (DAG) whose vertices correspond to the random variables $X_1, X_2, \ldots, X_n$ and whose arcs represent direct dependencies between the variables. A conditional dependency (which can be seen as a causal relationship) between two variables X_i and X_k defines an arc. The arc $X_k \rightarrow X_i$ describes a parent-child relation, where X_k is the parent and X_i is the child. Nodes that are not connected represent variables that are conditionally independent of each other. The graph G encodes independence assumptions: each variable X_i is independent of its nondescendants given its parents in G.

The second component of the pair i.e. Θ, represents the set of parameters that quantifies the network. It contains a parameter $\theta_{x_i|\pi_{x_i}} = P_B(x_i|\pi_{X_i})$ for each possible value x_i of X_i, and π_{x_i} of π_{X_i}, where π_{X_i} denotes the set of parents of X_i in G. A Bayesian network B defines a unique *joint probability distribution* over ψ given by eq. (4).

$$P_B(X_1,X_2,\ldots,X_n) = \prod_{i=1}^{n} P_B(X_i \mid \pi_{X_i}) = \prod_{i=1}^{n} \theta_{X_i} \mid \pi_{X_i} \qquad ♦ \ (4)$$

If a variable X_i has no parents its local probability distribution is referred to as *unconditional*, otherwise it is *conditional*. Also, if the variable represented by a node is observed, the node is said to be an *evidence node* otherwise it is said to be a *hidden node*.

A Bayesian classifier (BC) is a particular kind of BN that aims at correctly predicting the value of a discrete class variable, given the value of a vector of feature variables.

As proved in (Pearl 1988) the only nodes that have influence on the conditional probability distribution of a given node X (given the state of all the remaining nodes) are the nodes that belong to the Markov Blanket of X, an important concept formalized in Definition 6.

Definition 6. In a Bayesian network structure let λ_X represent the set of children of node X and π_X represent the set of parents of node X. The subset of nodes containing λ_X, π_X and any other parents of λ_X is called *Markov Blanket* (MB) of X. ♦

Fig. 1 shows a pictorial representation of the Markov Blanket of a variable X in a given Bayesian network.

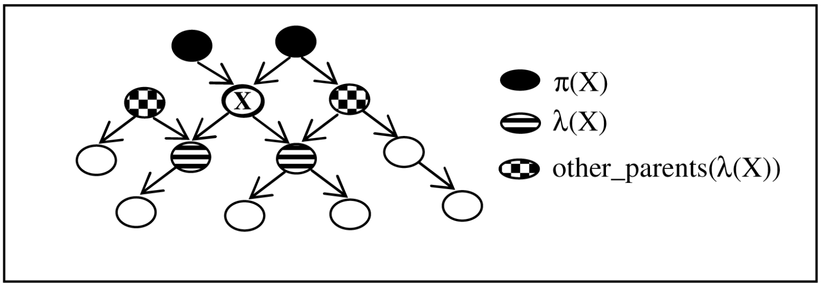


Fig. 1 MB(X) = { Z | Z ∈ π(X) or Z ∈ λ(X) or Z ∈ other_parents(λ(X))}

MB(X) contains all the nodes that shield X from the rest of the BN, i.e., the MB(X) is the only knowledge needed to predict the value of X. The concept of *moral graph* presented in Definition 7 is employed in the junction tree algorithm (Pearl 1988) which is used in belief propagation on graphical models.

Definition 7. Let G=<N1,A> be a DAG. Its counterpart *moralized graph*, G1=<N2,E> is a graph such that N1=N2 and $E = \{e \mid e = \text{undirected}(a), \text{ for all } a \in A\} \cup \{e\_new \mid e\_new = (n1,n2), n1 \neq n2, \mid \exists <n1,nk> \in A \wedge \exists <n2,nk> \in A\}$ ♦.

The corresponding moral graph of the DAG shown in Fig. 1 is shown in Fig. 2, where the three new added arcs are shown in thicker lines. A BN represents the conditional independence of a node and its predecessors, given its parents; the conditional independence test can be used for directing the construction of BNs. The concept of direction-dependent separation (*d-separation*), formally introduced in Definition 9 can be used to identify d-separated nodes in a BN.

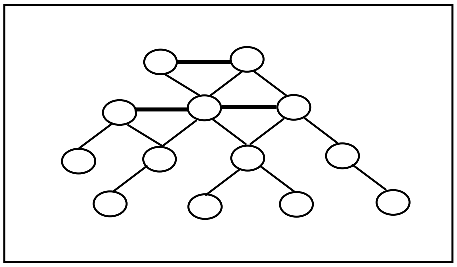


Fig. 2 Moral graph from the DAG shown in Fig. 1

Let G=<N,A> be a BN and let $X \subseteq N$, $Y \subseteq N$ and $E \subseteq N$ be three subsets of nodes. It can be proved that if every undirected path from a node in X to a node in Y is d-separated by E, then X and Y are *conditionally independent* given E. The proof that d-separated nodes are conditionally independent is elaborated and can be found in (Pearl 1988).

Definition 8. Let G=<N,A> be a BN. An *undirected path* in G is a path that does not take into account the directions of the arcs ♦.

Definition 9. (Russell and Norvig 1995) A set of nodes E *d-separates* two sets of nodes X and Y if every undirected path from a node in X to a node in Y is *blocked* given E. A path is blocked given a set of nodes E if there is a node Z on the path for which one of three conditions holds:

(1) Z is in E and Z has one arrow on the path leading in and one arrow out (chain).
(2) Z is in E and Z has both path arrows leading out.
(3) Neither Z nor any descendant of Z is in E, and both path arrows lead in to Z♦.

Fig. 3 based on (Russell and Norvig 1995) shows a pictorial representation of situations (1), (2) and (3) of Definition 9.

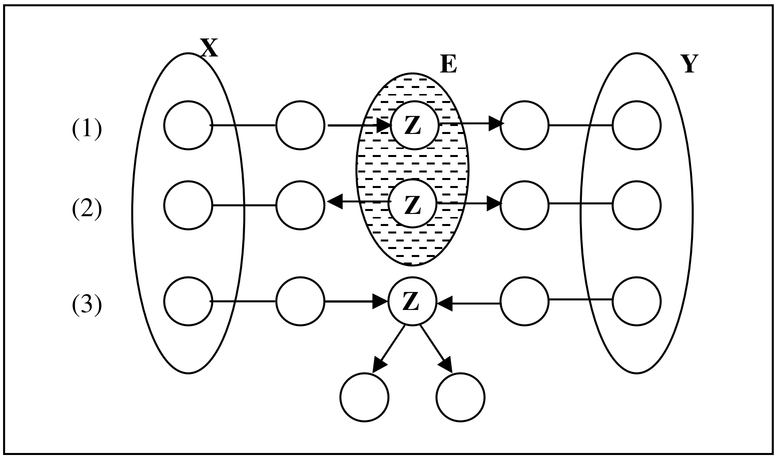


Fig. 3 Pictorial representation of the three situations a path from a node in X to a node in Y can be blocked, given the evidence E. If every path from X to Y is blocked, E d-separates X and Y

5.3 Learning Bayesian Networks and Bayesian Classifiers from Data

This section discusses the learning of Bayesian networks and Bayesian classifiers from data. Originally BNs were manually constructed by taking into account the variables involved in the problem and the causal dependencies among them. In the last years, however, with the advances of machine learning (ML) and ML techniques, several algorithms for inducing BNs from data have been proposed. Since in many practical situations all it is available is data, inductive learning algorithms play an important role in constructing BNs. As discussed in (Chickering 1996), learning Bayesian networks is NP-complete. BN learning can be divided into qualitative learning, focused on learning the DAG and quantitative learning, focused on the learning of conditional probabilities. Learning the BN structure is considered to be a more difficult problem than learning the BN parameters, unless the naïve Bayes method is employed, as discussed in Subsection 3.1.

As suggested in (Ben-Gal 2007), the BN learning problem can be stated informally as follows: given training data and prior information (e.g, expert knowledge, casual relationships), estimate the graph topology (network topology) and the parameters of the JPD in the BN (CPTables). One possible approach to the problem of inducing a BN from a training set is to use a scoring function to direct the search for an optimal BN in the space of possible BNs. Usual scoring functions are Bayesian scoring functions such as the one used in the K2 algorithm (Cooper and Herskovits 1992) presented in the Subsection 3.3 and others presented in (Heckerman *et al.* 1995). The function based on the minimal description length (MDL) principle (Lam and Bacchus 1993) (Suzuki 1993) is also commonly used.

Besides methods based on search-and-score, another approach, which conforms to constraint based learning, is based on conducting independence tests on the training data and construct the BN based on their results. Its main representative is the PC algorithm (Spirtes *et al.* 1993), discussed in Subsection 3.2.

5.3.1 The Naïve Bayes Classifier

In spite of its naivety, simplicity (its general DAG is always as displayed in Fig. 4) and relying on strong assumptions, the so called naïve Bayes classifier (NBC) is considered one of the most effective classifiers (see (Friedman *et al.* 1997 pg. 131) (Kohavi *et al.* 1997 pg. 79)). Langley and co-workers in (Langley *et al.* 1992) have shown that the NBC is competitive with one of the most successful ML system, the decision-tree inducer C4.5 (Quinlan 1993). The NBC assumes that:

- All other variables are conditionally independent of each other given the class variable.
- All other variables are directly dependent on the classification variable.

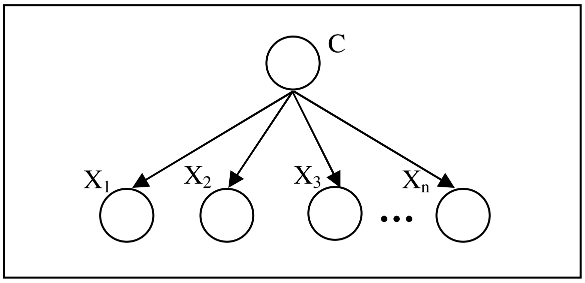


Fig. 4 The general DAG of a naïve Bayes network (classifier) where X_i's are features and C represents a class

Several NBC-based proposals attempt to achieve better performance than NBC by rewriting the assumptions. This is the case, for instance, of TAN (Tree Augmented Naïve Bayes) (Friedman and Goldszmidt 1996), SNB (Selective Naïve Bayes) (Langley and Sage 1994), BAN (Bayesian Network Augmented Naïve Bayes) (Cheng and Greiner 1999) and GBN (General Bayesian Network) (Cheng and Greiner 2001).

Since its DAG is always the same (dependent only on the number of features), the learning of a NBC consists purely in inferring, based on a given training data, the CPTables associated to each feature node, given the class label. In the classification phase, the Bayes rule is applied to compute the probability of a class label C_i given a pattern $\mathbf{X} = <X_1, X_2, \ldots, X_n>$, as show eq. (5), (6) and (7).

$$P(C_i|\mathbf{X}) = (P(\mathbf{X}|C_i) \times P(C_i))/P(\mathbf{X}) \qquad \text{Bayes rule} \qquad (5)$$

$$= P(X_1, X_2, \ldots, X_n|C_i) \times P(C_i) \qquad \text{P(}\mathbf{X}\text{) can be removed since it is the same for all class values.} \qquad (6)$$

$$= \prod_{k=1}^{n} P(X_k \mid C_i) \times P(C_i) \qquad \text{Taking into account the conditional independence assumption.} \qquad (7)$$

The probability given by eq. (7) is calculated for each class and the class with the largest posterior probability is assigned to the given pattern.

5.3.2 The PC Algorithm

The PC algorithm (Spirtes *et al.* 1993) starts the learning process from a complete, undirected graph (i.e., for every pair of nodes X and Y, $X \neq Y$, $\exists$ edge(X,Y)) and recursively deletes edges based on conditional independence tests, trying to identify the skeleton of the BN. The resulting structure can then be partially directed and further extended to represent the underlying DAG (Kalisch and Bühlmann 2007).

PC aims at a BN that represents the independence relationship among variables in a dataset and uses the conditional independence criteria $I(X_i,X_j|\mathbf{E})$ where **E** is a subset of variables, X_i and X_j are variables (a particular case of Definition 3, where the two first sets are singletons). If $I(X_i,X_j|\mathbf{E})$ is true, variable X_i is conditionally independent of X_j given **E** (which is verified using the d-separation criterion – see Definition 9). To verify whether X_i and X_j are conditionally independent given **E**, the cross entropy $CE(X_i,X_j|\mathbf{E})$ is computed, where the probabilities are their maximum likelihood estimators extracted from the data (i.e. relative frequencies). Other measures can also be used (Spirtes *et al.* 1993). The main steps of the PC algorithm are summarized in Fig. 5.

1. For each pair of variables, test for their conditional independence.
2. Based on the conditional independence results construct the skeleton (S) of the graph.
3. Identify the orientation of the edges in S.

Fig. 5 A high level description of the PC algorithm

Taking as input a list with all the independencies ($I(X_i,X_j|\mathbf{E})$) and adjacencies of each node (ADJ_{X_i}), PC first finds the graph skeleton (undirected graph) that best represents the d-separations expressed by $I(X_i,X_j|\mathbf{E})$ and then starts establishing the orientation of the edges.

As stated in (Spirtes and Meek 1995) "if the population, from which the sample input was drawn perfectly fits a DAG G, all of whose variables have been measured, and the population distribution P contains no conditional independence except those entailed by the factorization of P according to C, then in the large sample limit the PC algorithm produces the true pattern".

The variable preorder assumption can be used in the PC edge orientation step (step 3 of procedure described in Fig. 5) very successfully. To do that an ordered list (containing all the variable, class included) establishes that only variables that precede a given variable X_i may be parents of X_i. The use of a predefined order among variables can replace the search for the edge orientation.

The impact of variable orderings (VOs) on inducing efficient BCs was investigated in (Santos *et al.* 2007) using a genetic algorithm (GA) articulated to the PC algorithm, in a system named VOGA-PC. The role of the GA in the system was to search for a 'good' ordering among the variables – each individual (chromosome) in the GA population was a possible ordering. The class variable was not part of the chromosome; by default the class was always the first variable in any VO.

Each chromosome (i.e., each VO) was used in conjunction with the BC skeleton to induce a complete BC (skeleton + edge directions + CPTable). The BC was then input to a fitness function which, implementing a 10-fold cross validation process using training and testing sets, returned the average performance (Eval) of the BC. Based on performance results, the best chromosomes were then selected (tournament selection) and, using crossover and mutation operators, the next generation was built and the process repeated. Elitism of 1 was adopted i.e., in each generation the best ordering was kept and passed on to the next. The system VOGA-PC returns the best variable ordering (Best_VO) and the corresponding PC-induced BC. A more detailed description, results and analysis can be found in the previous cited reference.

5.3.3 The K2 Algorithm

The K2 algorithm (Cooper and Herskovits 1992) heuristically searches for the most probable BN structure given a dataset D containing n patterns and is based on four assumptions:

(1) Variables are discrete and all are observed (i.e., there are no hidden variables)
(2) Patterns occur independently, given a belief network model
(3) There are no patterns that have variables with missing values
(4) The density function $f(B_P|B_S)$ is uniform i.e., indifferent regarding the prior probabilities to place on a network structure B_S.

Considering the above assumptions, the algorithm looks for a Bayesian structure that best represents the patterns in D. The output of the K2 algorithm is a list of the parents of each node.

The variable preorder assumption is an important aspect of the algorithm. It is represented by an ordered list (containing all the variables, including the class) and asserts that variables can only be parents of variables that follow them in the list. The first variable in the list has no parents and that is why the head of the list is the class variable.

The network construction process uses a greedy method to search for the best structure. It begins as if every node has no parent. Then, beginning with the second variable of the ordered list, its possible parent candidates are evaluated and those that maximize the whole probability structure are added to the network. This process is repeated for each variable until the list finishes. It is done by maximizing the results of eq. (8).

$$P(B_S, D) = c\prod_{i=1}^{n}\prod_{j=1}^{q_i}\frac{(r_i-1)!}{(N_{ij}+r_i-1)!}\prod_{k=1}^{r_i}N_{ijk}! \tag{8}$$

where each discrete variable X_i ($i = 1, \ldots, n$) has r_i possible value assignments $\{v_{i_1}, v_{i_2}, \ldots, v_{i_{ri}}\}$. D is a dataset with m patterns, where each pattern contains a value assignment for each X_i ($i = 1, \ldots ,n$). Let B_S be a network structure containing just the variables X_i ($i = 1, \ldots, n$). Each variable X_i ($i = 1, \ldots, n$) in B_S has a set of parents represented by the list π_i. Let w_{ij} represents the j-th unique instantiation of π_i relative to D and suppose there are q_i such unique instantiations of π_i. Let N_{ijk} be the number of patterns in D in which X_i has value v_{ik} and π_i is instantiated as w_{ij}. Let $N_{ij} = \sum_{k=1}^{r_i} N_{ijk}$. Since by the fourth assumption previously stated the prior probabilities of all valid network structures are equal, $P(B_S)$ is a constant (c). Therefore, to maximize $P(B_S,D)$ requires finding the set of parents for each node that maximizes the second inner product of eq. (8).

With the best structure already defined, the network conditional probabilities must be determined. This is done using a Bayesian estimation of the (predefined) network structure probability. The Bayesian estimation is adopted in other learning Bayesian methods as in (Spiegelhalter and Lauritzen 1990), but there are other ways to compute this probability as shown in (Cooper and Herskovits 1992).

5.4 Bayesian Classifiers in Feature Subset Selection

This section initially defines and contextualizes the feature subset selection (FSS) problem, discussing its main characteristics and impacts on machine learning processes. It also identifies a few research works related to using Bayesian formalism for solving FSS related problems. Next it approaches the solution to the problem via a particular BN-based method, describes its main contributions and then presents a few experiments and discusses their results.

5.4.1 Considerations about the Feature Subset Selection (FSS) Problem

In many real-world problems the size of a training set can be very large in both dimensions: vertically (number of training patterns) and horizontally (number of features that describe the patterns). Large numbers in both dimensions represent problems to machine learning algorithms. Vertically large datasets are generally dealt with via a technique called sampling and the horizontally large datasets are dealt with via feature subset selection methods.

The FSS problem i.e., the selection of features that play an important role in characterizing a concept has been receiving growing attention particularly in areas such as Machine Learning and Data Mining. Research in feature selection methods has intensified in application areas where datasets are usually described by tens or hundreds of thousands of features (Guyon and Elisseeff 2003). In real-world problems, relevant features are often unknown and generally many features are used to describe the training patterns in an attempt to better represent the domain. Many of these features are either partially or completely irrelevant/redundant to the concept description. Theoretically, having more features should result in more discriminating power. However, practical experience with machine learning algorithms has shown that this is not always the case.

If the available data is suitable for machine learning, then the task of inducing the concept representation can be made easier and less time consuming by removing features that are irrelevant or redundant with respect to the concept to be learnt. In a typical situation shared by many supervised machine learning methods, given a training set which generally is described as a set of training patterns, each of them represented as a vector of feature-value pairs and an associated class, a feature selection method tries to identify features that are irrelevant or redundant for describing the concept (to be learnt) embedded in the training set.

By identifying and removing irrelevant and redundant features, these methods contribute to reducing the dimensionality of the space where concepts are represented. Machine learning and data mining techniques benefit from this since a reduction in dimensionality generally promotes the accuracy and comprehensibility of the induced concepts (Nicoletti 2007). It is common to approach the FSS problem as a heuristic search in a space defined by all possible subsets of a feature set. According to this model, Blum and Langley (1997) characterize any FSS method in terms of its stance on four basic issues that determine the nature of the heuristic search process:

1) STARTING POINT – selecting a point in the feature subset space from which to begin the search can affect the direction of the search.

1.1) All features – the search begins at the state represented by all features and successively removes them.

1.2) No features – the search begins at the state represented by no features and successively adds features.

1.3) Random – the search begins at a state represented by a set of randomly selected features.

2) SEARCH ORGANIZATION – characterizes the way the search is organized. There are two basic approaches and a few variants.

2.1) Exhaustive search – it is the simplest one, which exhaustively visits all possible states. This it not a viable alternative for most problems, since the size of the search space is 2^N, for a problem defined by N features.

2.2) Heuristic search – it is a more feasible way to conduct the search for real situations. Generally, at each space state, all the local possible moves are considered, one is selected and then a new iteration is performed.

3) EVALUATION STRATEGY – the way feature subsets are evaluated is the single biggest differentiating factor among feature selection algorithms for machine learning.

3.1) Filter – based on the general characteristics of the training set to select some features and exclude others. John, Kohavi, and Pfleger (John *et al.* 1994) call these filter methods, because they filter out irrelevant features before the induction process occurs.

3.2) Wrapper – wrapper strategies for feature selection use an inductive learning algorithm to estimate the merit of feature subsets.

3.3) Embedded – the FSS is an inherent part of the ML algorithm itself and is implemented by the learning method evaluation criteria for selecting the most relevant features (e.g. information gain criteria used by ID3 (Quinlan 1986)).

The filter approach is characterized as an independent approach – an algorithm performs the reduction (hopefully) of the number of features according to a quality metric associated to features, generally based on statistical values. Filter methods conduct the process of FSS as a pre-processing step of the original training set, based on intrinsic data characteristics (such as high information contents). They are usually based on statistic techniques and are very fast. This contributes to promoting the scalability of these methods. Fig. 6 shows a general diagram of filters.

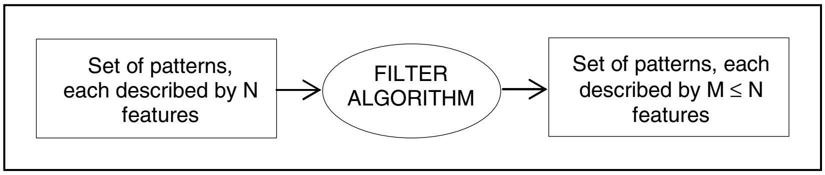

Set of patterns, each described by N features
FILTER ALGORITHM
Set of patterns, each described by M ≤ N features

Fig. 6 General scheme of a filter method for FSS

The wrapper approach works articulated to a ML method and combines a search method with a machine learning algorithm – the search is driven by the performance of the induced classifier. Fig. 7 shows a general diagram of wrappers.

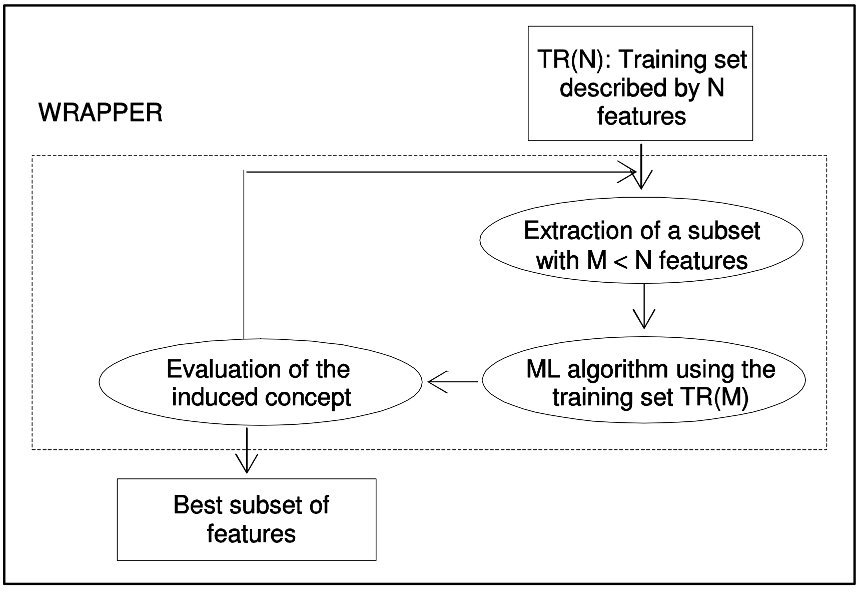

TR(N): Training set described by N features
WRAPPER
Extraction of a subset with M < N features
Evaluation of the induced concept
ML algorithm using the training set TR(M)
Best subset of features

Fig. 7 General scheme of a wrapper method for FSS

4) STOPPING CRITERION – the search for the feature subset can stop according to some pre-established criteria.

4.1) Number of feature has reached a pre-determined fixed value.

4.2) A feature selector might stop adding or removing features when none of the alternatives improves upon the merit of a current feature subset.

4.3) The algorithm might continue to revise the feature subset as long as the merit does not degrade. A further option could be to continue generating feature subsets until reaching the opposite end of the search space and then select the best.

In short, the process of finding a feature subset which allows the induction of good classifiers can be approached as a search problem in the space defined by the power set of the initial number of features. As a search problem, it can be implemented using the many available techniques, such as hill-climbing, beam search/best-first, random bit climber, Las Vegas and genetic algorithms. Reunanen in (Reunanen 2003) observes that there can be a few benefits from feature selection for learning:

- It is cheaper to measure only a subset of variables;
- Prediction accuracy might be improved through exclusion of irrelevant variables;
- The predictor to be built is usually simpler and potentially faster when less input variables are used;
- Knowing which variables are relevant can give insight into the nature of the prediction problem at hand.

There have been a few proposals to applying BN-based methods to the FSS problem, such as the hybrid method described in (Inza *et al.* 2001) and the work in (Inza *et al.* 2000). Antal and colleagues in (Antal *et al.* 2008) discuss applications of the Bayesian approach to new challenges in relevance analysis, which can be seen as a continuation of their work described in (Antal *et al.* 2006), where the generalizations of the FSS problem in a Bayesian framework based on the structural properties of BNs is formulated.

Fu and Desmarais in (Fu and Desmarais 2010) provide a review on related works on FSS based on the induction of the MB; Zeng and co-workers (Zeng *et al.* 2009) also used the concept of MB for filtering features and use the reduced feature set for learning. Brown and Tsamardinos in (Brown and Tsamardinos 2008) describe a new filter algorithm called Feature Space Markov Blanket (FSMB) which combines ideas borrowed from both, kernel and Markov Blanket based feature selection.

Authors in (Koller and Sahami 1996) have shown that the Markov Blanket criterion only removes features that are really unnecessary. They propose a heuristic approach for dealing with this problem but acknowledge that their algorithm performance can be improved by using more refined techniques, such as BNs, to choose candidate MBs. They also observe that finding an exact or an approximate MB can be a hard task. The proposal described in Subsection 4.2 is similar to the one described in (Cheng *et al.* 1997). Their main difference is that the work in (Cheng *et al.* 1997) uses a Conditional Independence Learning method and the method described in Subsection 4.2 uses a heuristic search based learning algorithm.

5.4.2 *Feature Subset Selection by Bayesian Networks – The K2χ^2 Method*

This section condenses part of the work described in (Hruschka Jr. *et al.* 2004), (Santos *et al.* 2007), (Hruschka Jr. and Ebecken 2002) where a BN-based feature selection method specifically designed for classification problems is proposed and evaluated. The method can be characterized as filter and basically (1) creates a Bayesian network from a training set and then (2) uses the Markov Blanket of the class variable as the set of relevant features for the corresponding classification problem. Fig. 8 shows the general flowchart of the method.

In order to create the Bayesian network (or classifier) from data, the variant K2χ^2 that combines the K2 algorithm (see Section 3) with the χ^2 statistic test was used. The χ^2 was employed aiming at optimizing the variable ordering to be used by the K2 algorithm, since this statistics test can be used to assess the independence of two variables (Liu and Motoda 1998). The χ^2 was used to measure the degree of dependence between the class variable and each of the variables describing the training set. The variables were then listed in descending order of their χ^2 result and the information was passed on to K2. As can be seen in (Hruschka Jr. and Ebecken 2007), the only difference between K2 and K2χ^2 is the use, by the later, of the information given by the variable ordering to induce the BC.

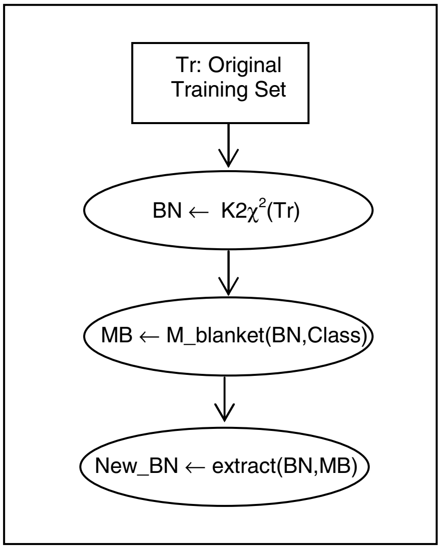


Fig. 8 Flowchart of the BN K2χ^2 inducer for feature selection (given by the Markov Blanket of the variable class)

Experiments were conducted using three knowledge domains from the UCI-Irvine Repository (Frank and Asuncion 2010), whose characteristics are presented in Table 1.

Table 1 Domain characteristics. The three domains have 2 classes. ONP/NP: original number of patterns/number of patterns, NP/C: number of patterns per class

Domain	ONP/NP	NP/C
Wisconsin Breast Cancer (WBC)	699/683	444/1 (benign) 239/2 (malignant)
Mushroom	8124/5644	1728/1 (edible) 3916/2 (inedible)
Congressional Voting Records (CVR)	435/232	124/1 (democrat) 108/2 (republican)

In the Wisconsin Breast Cancer (WBC) dataset the two classes are known to be linearly inseparable. The total number of features is 10 (including the class). The total number of patterns in the original WBC is 699; however 16 of them have missing feature values and were removed from the training data. The total number of patterns in the original Mushroom dataset is 8,124, each described by 22 features. However, 2,480 patterns whose eleventh feature was missing were removed and the remaining 5644 patterns were used in the experiments. The original Congressional Voting Records (CVR) dataset is described by 16 Boolean features and has 435 patterns, divided into 267 democrat and 168 republican. However, 203 patterns have missing values and were removed; the remaining 232 (124 democrat, 108 republican) were used in the experiments.

Aiming at identifying the influence of the variable ordering on the induced BCs, the correct classification rates (CCR) by both, K2 and K2χ^2 in the three knowledge domains, are shown in Table 2. In the table accuracy numbers refer to the average of a five-fold cross-validation process and μ and σ stand for average and standard deviation respectively. Simulations have shown that the ACCR (average correct classification rates) obtained using the training data with the original sequence of features and with the sequence given by the feature ordering were very close, leading to consistent results.

Table 2 ACCR of K2 *versus* K2χ^2. μ: average, σ: standard deviation

Dataset	Class	K2 (original variable ordering)		K2χ^2	
		μ	σ	μ	σ
WBC	1	96.84	1.66	96.61	1.58
	2	95.82	1.45	97.08	2.37
	Total	**96.48**	1.40	**96.78**	1.32
Mushroom	1	95.09	2.49	77.22	1.30
	2	5.92	3.23	87.93	1.90
	Total	**61.03**	0.72	**81.22**	0.63
CVR	1	63.33	28.49	96.0	5.65
	2	13.81	8.39	86.08	5.60
	Total	**40.17**	15.65	**91.40**	3.62

In Table 3 the selected set of features (i.e., the MB of the corresponding class feature), for each domain, is presented.

Table 3 |OF|: number of original features (class excluded), SF: selected feature set (class excluded)/|SF|

Domain	|OF|	SF/|SF|
WBC	9	{x2, x3, x4, x6, x7, x8}/6
Mushroom	22	{x3, x5, x9}/3
CVR	16	{x3, x4, x5, x12}/4

To verify the consistency of the generated networks and to provide evidence that the selected features are relevant to the model, classification tasks were performed (a) using the original features and (b) using the selected features. Results are summarized in Table 4. In all learning tasks a five-fold cross-validation process was applied.

Results from the just described FSS approach were compared against results given by classifiers induced by three different algorithms, using the original training set (all features present). The classifiers were (1) a Bayesian classifier induced by the Naïve Bayes method; (2) a decision tree induced by the C4.5 algorithm (Quinlan 1993) in its version available at the WEKA System, identified as J48 (Witten and Frank 2000); (3) a set of rules obtained by J48 PART (Witten and Frank 2000), a method that extracts rules from pruned partial decision trees (also built using the C4.5 algorithm).

Table 4 BC results (%) per domain

Class	Original Features		Selected Features	
	μ	σ	μ	σ
		WBC		
1	96.61	1.58	96.61	1.58
2	97.08	2.37	95.40	2.70
Total	**96.78**	**1.32**	**96.19**	**1.19**
		Mushroom		
1	77.22	0.70	94.61	1.03
2	87.93	0.59	96.25	1.90
Total	**81.22**	**0.41**	**95.11**	**0.63**
		CVR		
1	96.00	5.65	96.00	5.65
2	86.08	5.60	85.08	6.25
Total	**91.40**	**3.62**	**90.95**	**3.48**

As mentioned before, there are three main classes of algorithms for learning Bayesian networks. One refers to algorithms based on heuristic search, the second to algorithms based on the use the conditional independence concept and the third to algorithms that combine both previous strategies.

When using algorithms based on heuristic search, one important issue is the initial order by which features are presented to the algorithms. Algorithms use this information to determine the direction of arcs – a variable is a possible parent only of those that follow it in the ordering (Hruschka Jr. and Ebecken 2002). Conditional independence methods try to find the direction of arcs without the information given by the variable ordering. It has been reported, however, that algorithms have an improved performance when the ordering is provided (Spirtes *et al.* 1993).

Results achieved by each of the three other classifiers in the three domains are presented in tables 5, 6 and 7 respectively. In the three tables O and S stand for 'Original' (all features in the original dataset) and 'Selected' (selected features given by the MB) respectively.

The Naïve Bayes method uses all features and allows them to make contributions to the decision that are equally important and independent of one another, given the class. This leads to a simple scheme that works well in practice (Witten and Frank 2000). One can observe that the proposed method selects very predictive features, which in Mushroom and Congress provide even better ACCRs than those achieved in the dataset formed by all features. Table 5 shows the results obtained in complete datasets and in datasets described by the selected set of features with the Naïve Bayes.

Table 5 Naïve Bayes – Average Classification Rates (%).

O: original features, S: selected features (MB)

Dataset	Total	Class 1	Class 2
WBC O	96.49	95.70	97.90
WBC S	95.90	95.70	96.20
Mushroom O	97.36	99.60	93.80
Mushroom S	99.22	100.00	98.00
CVR O	91.40	88.70	94.40
CVR S	93.10	91.10	95.40

The J48 algorithm is the WEKA´s implementation of the popular C4.5 (Quinlan 1993). In fact, J48 is a C4.5 improved version, called revision 8 (Witten and Frank 2000). Table 6 shows the simulation results using J48. The table also shows the set of features present in the best induced classifier obtained in the five-fold cross validation process. It can be noticed that all features selected by the Bayesian approach were employed both in the Mushroom and WBC domains; in the CVR domain, however, the feature selection process was not so important, since only one feature is necessary to classify all examples. Besides, in the simulations the selection method was consistent in the context of J48, and consequently with the information gain criterion.

Table 6 J48 – Average Correct Classification Rates (%).

O: original features, S: selected features (MB)

Dataset	Features in the best classifier induced	Total	Class 1	Class 2
WBC O	{x1,x2,x3,x4,x5,x6,x7}	95.17	96.40	92.90
WBC S	{x2,x3,x4,x6,x7,x8}	95.61	95.00	96.70
Mushroom O	{x5,x18,x17,x8,x4,x20}	100.00	100.00	100.00
Mushroom S	{x3,x5,x9}	99.86	99.80	100.00
CVR O	{x4}	95.26	95.20	95.40
CVR S	{x4}	96.98	95.20	99.10

The J48 PART extracts rules from pruned partial decision trees built using C4.5; it combines the divide-and-conquer strategy for decision trees learning with the separate-and-conquer for rule learning (Witten and Frank 2000). To make a single rule, a pruned decision tree is built based on the current set of patterns and then the path to the leaf with the largest coverage is made into a rule and the tree is discarded. Table 7 shows the obtained results with J48 PART. Again, all the selected features were employed both in Mushroom and in WBC datasets, whereas in CVR only one of the selected features was enough to classify all examples. The results are consistent with those in Table 6, indicating that the proposed method is a good option as a FSS method, allowing the extraction of simple rules with very good ACCRs.

Table 7 J48 PART – Average Correct Classification Rates (%).

O: original features, S: selected features (MB)

Dataset	Features in the best classifier induced	Total	Class 1	Class 2
WBC O	{x1,x2,x3,x4,x5,x6,x7,x8}	96.05	97.50	93.30
WBC S	{x2,x3,x4,x6,x7,x8}	95.31	95.90	94.10
Mushroom O	{x5,x8,x11,x17,x18,x20,x21}	100.00	100.00	100.00
Mushroom S	{x3,x5,x9}	99.86	99.80	100.00
CVR O	{x2,x3,x4,x9,x11}	94.40	95.20	93.50
CVR S (4)	{x4}	96.98	95.20	99.10

The main simulation results are condensed in Table 8, where the total ACCR values as well as the number of features employed per domain per classifier are presented. In general, the ACCRs obtained using as training data the datasets described by the original features were very close to those obtained using datasets described by the selected features. It is noticeable the significant improvements in relation to the number of employed features. In the WBC, 66.67% of the original features were enough to obtain high classification rates. A similar effect was observed in the Mushroom, where only 13.64% of the original number of features was selected. It is noticeable that the Bayesian network accuracy improved significantly. Finally, in the CVR, by using a specific 25% of the original number of features the classification rate still was high. Another important aspect is that the ACCR values obtained by the classifiers (using original or selected subset of features) are comparable to the best ones found in the literature. Duch and colleagues in (Duch *et al.* 2000), for instance, describe classification results obtained by 15 methods. Their results, in the Mushroom dataset, vary from 91% to 100%, while in the WBC they vary from 92.7% to 99%. The CVR dataset information file reports accuracies that vary from 90% to 95% (Schllimmer 1987).

Table 8 Main simulation results: Average Correct Classification Rates (%).

O: original set of features, S: selected set of features

Classifier	WBC		Mushroom		CVR	
	O (9)	S (6)	O (22)	S (3)	O (16)	S (4)
BN	96.78	96.19	81.22	95.11	91.40	90.95
Naïve Bayes	96.49	95.90	97.36	99.22	91.40	93.10
J48	95.17	95.61	100.00	99.86	95.26	96.98
J48 PART	96.05	95.31	100.00	99.86	94.40	96.98

5.5 Bayesian Classifiers in Imputation Processes

This section initially defines and contextualizes imputation processes, discussing its main characteristics, uses and impacts on machine learning processes. Next it describes the proposal of a BN-based imputation process and its main contributions.

5.5.1 Considerations about Imputation Processes

The absence of information is common in real-world databases and it can occur due to a number of reasons, such as malfunctioning measurement equipment, changes in experimental design during data collection, collation of several similar but not identical datasets, refusal of some respondents to answer certain questions in surveys, etc. Such missing data are usually problematic. Therefore, several approaches have been proposed to deal with them as can be seen in (Rubin 1976), (Rubin 1987), (Little and Rubin 1987), (Pyle 1999) and (Schafer 2000). A simple approach to deal with missing values ignores patterns and/or features containing missing values; the loss of data however can be considerable and reduced datasets may lead to biased statistical analyses. Alternatively, some approaches for data analysis (e.g. (Breiman *et al.* 1983), (Quinlan 1993)) can be tolerant to missing values. Finally, a significant number of machine learning methods work only with complete datasets. For these methods, approaches aimed at filling in missing values are particularly relevant.

The task of filling in missing data is often referred to as missing values substitution or imputation and it can be performed in a number of ways such as by the widely used naïve mean/mode method. The substitution of missing values by the mean/mode can eventually lead to reasonable results. This procedure, however, assumes that all missing values represent the same value, possibly leading to considerable distortions. The mean/mode method underestimates the population variance and does not take into account the relationships between features, which are usually relevant to the process of missing values replacement. Moreover, machine learning methods usually explore relationships between features and, thus, it is critical to preserve them, as far as possible, when replacing missing values (Pyle 1999). In this sense, imputation is aimed at carefully substituting missing values, trying to avoid the insertion of bias in the dataset. If imputation is performed in a suitable way, higher quality data might becomes available and results from machine learning tasks can be improved.

Techniques to deal with missing values have already been studied for many years (e.g. see (Anderson 1946), (Preece 1971), (Dempster *et al.* 1977), (Rubin 1977), (Rubin 1987), (Schafer 2000)). Although most of these techniques have been applied to survey data analysis, they can also be useful for machine learning applications. Therefore, before focusing on the specific imputation method described in (Hruschka Jr. *et al.* 2007), a brief survey on imputation methods is presented next.

5.5.2 Commonly Used Imputation Methods

The expectation-maximisation (EM) algorithm (Dempster *et al.* 1977), (Redner and Walker 1984), (Wu 1983), (Ghahramami and Jordan 1995), (Jordan and Xu 1996), (Bilmes 1997) has been widely used for imputation. EM assumes that missing data (Y) are governed by a distribution $f(Y|X,\theta)$, where X (data without missing values) and the parameters θ (mean and variance) are fixed. The EM algorithm is based on the likelihood function, and it fills in the missing data based on an initial estimate of θ. Then, it re-estimates θ based on the complete and filled data, iterating until the estimates converge.

Depending on the complexity of the density function that describes the dataset, the convergence may be slow (Little and Rubin 1987). In addition, the computations performed by EM are dependent on the assumption of a particular density function and its parameters.

Multiple imputation (MI) (Rubin 1977) has been widely used for multivariate analysis, and it consists in using more than one value to fill in the gaps in the sample (e.g. the mean of probable values). MI can provide good results, but the involved computational cost is considerably higher when compared to single imputations (Rubin 1987).

Data augmentation (DA) (Tanner and Wong 1987) can be informally described as the process in which observed data Y (whose distribution depends on the parameters θ) is augmented by the quantity Z (using a Monte Carlo sampling strategy). Based on the MI idea, multiple values for Z can be generated using the $p(Z|Y)$ distribution and then obtaining $p(\theta|Y)$ as the average of $p(\theta|Y,Z)$ over the imputed Zs. In theory, this method provides a way to improve the inference in small samples, a situation where EM has pitfalls.

Considering decision trees, some practical results about ignoring patterns with missing values can be found in Quinlan (1986) and White (1987). Another approach involves replacing missing values with the most frequent value (Kononenko *et al.* 1984). In the probability method (Quinlan 1989), (Lobo and Noneao 2000), a decision tree is constructed to substitute missing values of each feature, using the information contained in the other features. The dynamic path generation (Quinlan 1986) and the lazy decision tree approach (Friedman *et al.* 1996) do not generate the whole tree, but only the most promising path instead.

Several imputation methods based on nearest neighbours can be found in the literature. They basically select patterns with feature values similar to the pattern of interest to impute missing values. For instance, see (Troyanskaya *et al.* 2001), (Batista and Monard 2003) and (Hruschka *et al.* 2003).

According to Schaffer and Graham (2002), maximum likelihood methods (e.g. EM and Bayesian algorithms) represent the state of the art for imputation. Considering high-dimensional datasets, BNs are usually more efficient than methods based on the EM algorithm (Zio *et al.* 2004). Zio and co-workers describe the use of BNs for imputing missing values, arguing that two relevant advantages of using BNs as imputation models are the possibility of preserving statistical relationships between variables, and dealing with high-dimensional datasets.

5.5.3 Imputation by Bayesian Networks and the K2Iχ2 Method

There have been a few attempts towards imputation processes articulated to Bayesian networks, such as the proposal described in (Kong *et al.* 1994). Zio and colleagues in (Zio *et al.* 2004) discuss the use of BNs for imputation aiming at dealing with the problem of the consistency of imputed values: preservation of statistical relationships between variables (statistical consistency) and preservation of logical constraints in data (logical consistency).

In order to tackle the missing value problem in classification tasks, the K2Iχ^2 method was proposed to impute (substitute) missing values based on Bayesian networks as described in (Hruschka Jr. *et al.* 2007).

K2Iχ^2 relies on the construction of a BN to infer the most suitable values to fill in the gaps produced by missing values. K2χ^2 learning algorithm (as described in Section 4.2) is applied to construct a BN to be used as a prediction model to substitute the missing values. Instead of generating one BN for each feature with missing values, as described in (Hruschka Jr. and Ebecken 2002), K2Iχ^2 builds a single BN to infer the best values to substitute the missing ones in all features. The unrestricted BN is therefore used considering all variables as potential predictors.

The imputation process performed by K2Iχ^2 can be summarised by the following steps: (i) generate a single clean (i.e., without missing values) training dataset C; (ii) build an unrestricted BN' using C and (iii) use BN' to infer the best values to replace the missing ones.

In (Hruschka Jr. *et al.* 2007) K2Iχ^2 is evaluated in the context of both prediction and classification tasks, and its performance is compared with those obtained by classical imputation methods (EM, Data Augmentation, Decision Trees, and Mean/Mode). The simulations were performed on four UCI (Frank and Asuncion 2010) datasets (Congressional Voting Records, Mushroom, Wisconsin Breast Cancer and Adult), which are benchmarks for data mining methods. Missing values were simulated by the elimination of some known values. Thus, it was possible to compare original values with imputed ones, evaluating the prediction capability of the imputation methods. In addition, a methodology to estimate the bias inserted by imputation methods in classification tasks is proposed. Four classifiers (One Rule, Naïve Bayes, J48 and J48 PART) were used to evaluate the five imputation methods in classification scenarios. Computing times consumed to perform imputations were also reported for each imputation method. Simulation results in terms of prediction, classification and computing times allow to perform several analyses, leading to interesting conclusions. K2Iχ^2 has shown to be competitive against classical imputation methods and achieved good results when variable relationships as well as the imputation bias were taken into account. More discussions on imputation bias and BN imputation methods can be found in (Hruschka *et al.* 2009).

5.6 Post-processing a Bayesian Classifier into a Set of Rules

This section describes in detail how BNs can be post-processed in order to create a set of rules. The main motivation for post-processing a BC into a set of rules is for the sake of understandability. Many automatic learning tasks are used in real data domains which, generally, are described by a large number of features; in such domains the induced classifiers tend to be large and complex and usually, hard to be completely understood by humans.

It is a fact that knowledge represented by BCs is not as comprehensible as knowledge represented by some other forms such that of classification rules. This can be a drawback in areas where the understandability of the representation plays a major role. Reasoning with rules is comprehensible, provides explanations, and may be validated by human inspection. It can also increase the user's confidence in the system and eventually can help to discover important relationships among variables.

The BayesRule method described in this section, originally proposed in (Hruschka Jr. *et al.* 2008), translates a learnt BC into a set of classification rules, a much more suitable knowledge representation for promoting understandability. The experiments show that the reduced set of rules extracted from a BC can be reasonably condensed and still maintain the original BC classification accuracy.

Subsection 6.1 addresses the description of the BayesRule algorithm. The experimental results, described in Section 6.2, show that it is possible to extract a reduced number of simple rules from a BC and, thus, circumvent the dimensionality problem without the use of complex procedures of optimization and pruning.

5.6.1 Translating a Bayesian Classifier into a Reduced Set of Rules – The BayesRule Algorithm

As discussed in Section 2 and proved in (Pearl 1988), the only nodes that have influence on the conditional probability distribution of a given node X (given the state of all the remaining nodes of a BC) are the nodes that belong to the Markov Blanket of X. Thus, after learning a BC from data, the Markov Blanket of the class node identifies, among all nodes that define the BC those that influence on the class node. In the BayesRule method the MB concept is used to reduce both the number as well as the complexity (in relation to the number of variable tests in the antecedent) of classification rules. When generating the set of propositional classification rules, the only variables taken into account are those in the MB of the class variable.

As standard propositional if-then classification rule is the simplest and the most comprehensible way to represent classification knowledge it has been adopted by BayesRule. The BayesRule method is based on the intuition that the best explanation for a piece of evidence is the most probable state of the world, given the evidence. This approach is called maximum a posteriori (MAP) approach (see (Henrion and Druzdzel 1990) and (Pearl 1988) for details).

MAP is a standard approach to parameter estimation and inference in statistics. When concerning classification tasks, many algorithms consider some candidate classes (or hypothesis) $\{C_1, C_2, \ldots, C_j\}$ and try to identify the one that best fits a given background (BK) knowledge. The choice of the best class is often based on the most probable class C_J ($J = 1, \ldots, j$) given the BK. Any such maximally probable class is called Maximum a Posteriori (MAP) class (C_{MAP}). As proved in (Mitchell 1997), a BC, as well as any other classifier based on the Bayes Theorem, can be used to calculate the posterior probability of each candidate hypothesis as follows:

$$C_{MAP} = C_{J_k} \text{ such that } P(BK \mid C_{J_k})P(C_{J_k}) = \max_{J \in \{1,\ldots,j\}} P(BK \mid C_J)P(C_J) \qquad (9)$$

In the particular situation where all classes C_J are equally probable *a priori* (i.e. $P(C_m) = P(C_n)$, $\forall$ m, n $\in \{1, 2, \ldots, j\}$, $m \neq n$), the term $P(C_J)$ can be removed from eq. (9) and C_{MAP} is renamed as Maximum Likelihood (ML) class (C_{ML}).

When using the MAP approach to extract rules from a BC, one rule is created for each possible value of the involved variables, a computationally expensive procedure. This happens mainly because it is very common the presence of hundreds or thousands of variables in probabilistic models (Druzdzel 1996). In most cases, however, many variables may only be relevant for some types of reasoning; very rarely all of them will be relevant in the reasoning process associated to one single query, for instance. Therefore, it is essential to focus only on the relevant part of the model (i.e. the class variable and the variables that belong to its Markov Blanket) when translating it into a set of rules. In this sense, the proposed BayesRule method uses the Markov Blanket concept to select the variables that will be in the antecedent of rules. Thus both, the number and the complexity of rules are minimized along with the rule extraction process. The variable selection strategy, however, does not guarantee a minimal rule set.

The extracted rule set can still undergo a pruning step which will remove from the rule set the rules containing superfluous conditions.

In accordance with the MAP approach, a BC evidence propagation algorithm must be used to propagate the variable values aiming at inferring the class value. Let A be a set of variables that are instantiated and B a set of their corresponding values. An evidence propagation algorithm determines $P(V_{i,J_i} |A,B)$ for all values V_{i,J_i} of all variables in the network except those in A. For singly-connected (only one edge is allowed between two nodes) BNs there are simple and efficient evidence propagation algorithms; when the BN structure is a multiply-connected graph, however, the evidence propagation process is, in the worst case, NP-hard (Cooper 1990).

It is agreed that the most popular exact BN inference algorithm is Lauritzen and Spiegelhalter's clique-tree propagation algorithm (Lauritzen and Spiegelhalter 1988). Their algorithm is based on Pearl's polytree propagation algorithm (Pearl 1988). However, considering that Pearl's algorithm can only be used when the BN structure is a polytree, the clique-tree propagation algorithm first transforms a multiply-connected network into a clique tree by clustering the triangulated moral graph of the underlying undirected graph, then performs message propagation over the clique tree using the classic polytree algorithm.

Pearl's polytree algorithm exploits the fact that a polytree is a singly-connected structure and consequently, there is only one path between two nodes; this special characteristic allows only one choice for transmitting the evidence in the BN, i.e., there is no risk of redundant propagation. The polytree propagation algorithm may be summarized as follows: each node (variable) exchanges messages with its parents and its children. Messages sent from parents to children are called π-messages and messages sent from children to parents are called λ-messages. In a BN, for each new evidence impacting on a node X, this node must update its own CPTable and propagate the new values to all its parents nodes $\pi_{(X)}$ and to all its children nodes $\lambda_{(X)}$. Once a node U (parent of node X) receives the new probability values from X, it must update its own CPTable and propagate the updated values to all its parents nodes $\pi_{(U)}$ and to all (except X) its children nodes $\lambda_{(U)}$. Once a node Y (child of node X) receives the new probability values from X, the same updating process must be repeated. In this sense, Y must update its own CPTable, and propagate its new probability values to all (except X) its parents nodes $\pi_{(Y)}$ and to all its children nodes $\lambda_{(Y)}$. According to the updating procedure, the new evidence impacted on X will be propagated to all nodes in the BN without redundancy.

As previously described, when a node receives new evidence, or a message from its parents or children, it must update its own CPTable. This is done simply by multiplying its probability estimation matrix (CPTable) by the probability estimation vector received from its parents (or children).

The BayeRule implements the evidence propagation algorithm proposed by Lauritzen and Spiegelhalter because it is an efficient algorithm even when applied to BNs consisting of very large number of variables. Fig. 9 presents a simplified version of the pseudocode of the BayesRule method; the procedure expects as input a Bayesian Classifier with N nodes and assumes by default that the class variable is X_1.

```
procedure BayesRule;
input     BC: Bayesian Classifier with N nodes
          $X_1$: Class variable
output: RSR {Reduced Set of Rules}

begin

 1.  RSR ← ∅ {reduced set of rules is empty}
 2.  CMB ← MB($X_1$) {Markov Blanket of $X_1$ (class variable)}
 3.  M ← |CMB|
 4.  Rename the variables in CMB as $X_2$, $X_3$,…, $X_{M+1}$
 5.  for i ← 2 to M+1 do
 6.  begin
 7.   $V_i$ ← set of possible values of variable $X_i$
 8.   $j_i$ ← |$V_i$|
 9.  end
 10. RI ← 1   {rule index}
 11. for $k_2$ ← 1 to $j_2$ do
 12.  for $k_3$ ← 1 to $j_3$ do
 13.   ………………
 14.    for $k_{M+1}$ ← 1 to $j_{M+1}$ do
 15.     begin
 16.      Rule_antecedent ← $X_2 = v_{2_{k_2}}$ and $X_3 = v_{3_{k_3}}$ and … and $X_{M+1} = v_{N_{k_{M+1}}}$
 17.        • propagate Rule_antecedent throughout BC and determine
              the class value Val_Class and certainty factor F
 18.           • define rule $R_{RI}$ as: if Rule_antecedent then $X_1$ =
          Val_Class(F%)
 19.      RSR ← RSR ∪ {$R_{RI}$}
 20.      RI ← RI + 1
 21.     end
 22. RSR ← remove_irrelevant_rules(RSR)
end
```

Fig. 9 Pseudocode of the BayesRule algorithm

The BayesRule procedure for extracting classification rules from BCs is quite simple. As mentioned before, the BC structure provides a simple and efficient mechanism (Markov Blanket) to reduce the number and the complexity of the rule set. The procedure described in Fig. 9 creates probabilistic rules as:

If < *condition* > then < *class* > with certainty F

where *condition* is the antecedent part of the rule, *class* the consequent and F is a percentage value.

Let V_1, V_2,… , V_n, C be the sets of variable values for X_1, X_2,…, X_n and C, respectively. Also, let $|V_i| = j_i$, i = 1,… , n and |C| = j. A probabilistic if-then rule can be characterized as:

$$\text{If } X_1 \text{ is } V_{1,J_1} \text{ and } \ldots \text{ and } X_n \text{ is } V_{n,J_n} \text{ then C is } C_J \text{ (F\%)}$$

where $J_i \in \{1,\ldots,j_i\}$, i = 1, … , n and $J \in \{1,\ldots,j\}$.

By using the BayesRule method, the number of variables involved in the antecedent of a rule is reduced since the method only considers the Markov Blanket of the class variable C. Considering a particular situation where the Markov Blanket of the class variable C is the set $\{X_1,\ldots, X_k\}$, the *a posteriori* probability of class $C = C_j$ given the values of the variables in the Markov Blanket of class C for a particular instantiation of indexes J_i, i = 1, …, k is

$$P(C = C_j \mid V_{1,J_1}, \ldots, V_{k,J_k}) = \max_{J \in \{1,\cdots,j\}}\{P(C = C_J \mid V_{1,J_1}, \ldots, V_{k,J_k})\}$$

where each $P(C = C_J \mid V_{1,J_1}, \ldots, V_{k,J_k})$ is calculated using eq.(4).

The confidence degree associated to a rule can be defined using inferential results. In doing so, the probability given to the inferred class may be used as a confidence value and it is embedded in the inference algorithm. The rule coverage can be obtained from the numerical parameters (CPTables) already stored in the BC, and consequently no extra computation is needed for defining it.

As an example of how the BayesRule procedure works, consider a BN with 5 nodes and 5 arcs, as depicted in Fig. 10 (borrowed from (Cooper 1984)), referred to as Example_BN. Consider also that all nodes (variables) are binary (present/absent) and the class variable is CA.

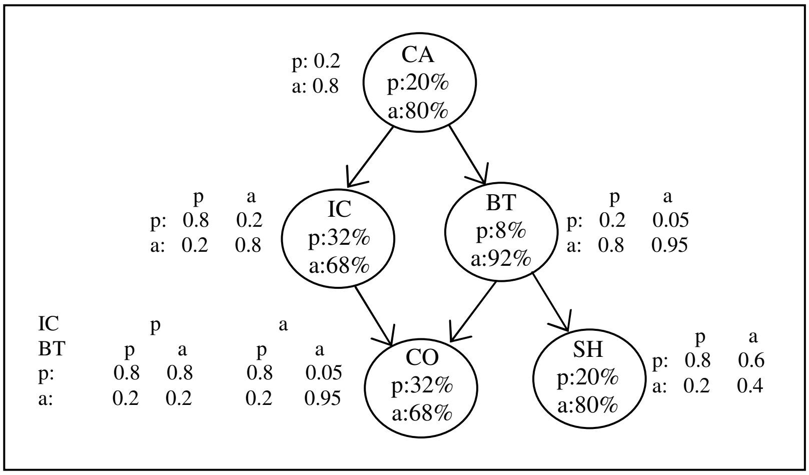


Fig. 10 Example_BN: an example for explaining how the rule extraction process conducted by BayesRule works

As can easily be seen in Fig. 10, MB(CA) = {IC, BT}. Since the step 4 of BayesRule procedure renames variables, CA, IC and BT as X_1, X_2 and X_3 respectively. Considering that the two variables (X_2 and X_3) defining the antecedent part of the rules are binary, four rules will be created. The Lauritzen and Spigelhalter propagation algorithm (Lauritzen and Spigelhalter 1988) was used to determine the value of X_1 for the possible combinations of X_2 and X_3. In the example the four extracted rules shown in Fig. 11 define the final RSR (reduced set of rules) created by BayesRule.

R_1: if X_2 = present and X_3 = present then X_1 = present (80%)
R_2: if X_2 = present and X_3 = absent then X_1 = absent (54%)
R_3: if X_2 = absent and X_3 = present then X_1 = absent (80%)
R_4: if X_2 = absent and X_3 = absent then X_1 = absent (95%)

Fig. 11 Reduced Set of Rules (RSR) extracted by BayesRule from the BN shown in Fig. 10

It is worth remembering that the number of rules has a significant impact on the system accuracy as well as on its understandability as a motivation for the introduction of the last step of the BayesRule procedure. While a high number of rules may improve classification accuracy, it may also disrupt understandability. It is also well known that rules with too many conditions in their antecedent parts are more difficult to understand than those with a lesser number of conditions. Taking into account both issues, the last step of BayesRule prunes the RSR particularly when the set has a large number of long rules. When having a small RSR, however, the pruning step may not be applied. As described in line 22 of the BayesRule algorithm, a pruning step may be applied to the final RSR. Considering the RSR extracted from Example_BN (Fig. 10) the pruning step would be skipped (as the RSR has only four short rules). Nevertheless, to illustrate the *remove_irrelevant_rules(RSR)* procedure consider applying it to the RSR shown in Fig. 11. A careful look at rules R_3 and R_4 reveals that when X_2 = absent, X_1 is always classified as absent (X_3 value has no influence in the class definition in such a situation). Thus, the *remove_irrelevant_rules(RSR)* replaces rules R_3 and R_4 by a new rule defined as: "if X_2 = absent then X_1 = absent (96%)". As rules R_3 and R_4 were removed, the new rule is named R_3 and a more reduced set of rules is generated as depicted in Fig. 12.

R_1: if X_2 = present and X_3 = present then X_1 = present (80%)
R_2: if X_2 = present and X_3 = absent then X_1 = absent (54%)
R_3: if X_2 = absent then X_1 = absent (96%)

Fig. 12 Reduced Set of Rules (RSR) extracted by BayesRule using a naïve pruning strategy from the BN shown in Fig. 10

In addition, two important issues should be taken into consideration in the pruning process. The first one is that the probability estimation (96%) of the new R_3 rule was obtained running the Lauritzen and Spiegelhalter clique tree algorithm (considering X_2 = absent as the only evidence to be propagated). The second issue is that this pruning strategy is very simple and more elaborated techniques should be investigated in further implementations. The experiments described in the next subsection show that in addition to produce a reduced set of rules, BayesRule can still maintain a good classification performance.

5.6.2 Using BayesRule - Experiments and Results

In order to empirically validate BayesRule, a few experiments were conducted. Considering that BayesRule expects as input a Bayesian Network and a class variable the experiments were based on well-known BNs. The two main advantages of working with a well-known BN are (1) as it is possible to know *a priori* the real dataset probability distribution and its characteristics, the results obtained in the experiments can be analyzed in a more consistent and reliable way; (2) it is possible to inspect the behavior of BayesRule in specific situations of interest.

Five well-known BNs namely Alarm (Beinlich *et al.* 1989), Asia (Lauritzen and Spiegelhalter 1988), Credit (Druzdzel 1996), Engine Fuel System (Engine) (Druzdzel 1996), Win95pts (Horvitz *et al.* 1998) and two artificially created BNs, referred to as Synthetic 1 (Syn_1) and Synthetic 2 (Syn_2), were employed in the experiments. Table 9 summarizes the dataset characteristics.

Table 9 Dataset description where AT: number of features plus class, IN: number of patterns and Cl: number of classes

	Alarm	Asia	Credit	Engine	Win95pts	Syn_1	Syn_ 2
AT	38	8	12	9	76	32	32
IN	10^4	10^4	10^4	10^4	10^4	10^4	10^4
Cl	See Table 10	2	2	2	See Table 10	2	2

Table 10 Alarm and Win95pts variable class names and their respective domain sizes

Alarm		Win95pts	
Class name	|Class|	Class name	|Class|
Anaphylaxis	2	Repeatable Problem	2
Intubation	3	Driver File Status	2
KinkedTube	2		
Disconnect	2		
Hypovolemia	2		
InsuffAnesth	2		
LVFailure	2		
PulmEmboulus	2		

The Alarm BN, a network for monitoring patients in intensive care, is based on expert knowledge and was originally described in (Beinlich *et al.* 1989). It is defined by 37 variables and 46 arcs and represents 8 diagnostic variables, 16 measurements, and 13 intermediate variables that connect diagnostic problems to findings. The diagnostic variables have no predecessors and are assumed to be mutually independent *a priori*. These variables represent the presence, absence or the severity of a particular disease. The measurement variables represent quantitative information available when a patient is being monitored. The intermediate variables cannot be measured and, thus, are inferred using the available information. In the conducted experiments, the Alarm BN was used as a BC. The eight Alarm diagnostic variables *Hypovolemia, LVFailure, Anaphylaxis, Insufficient Anesthesia, PulmEmboulus, Intubation, KikedTube* and *Disconnect* were used for classification purposes (i.e., were considered one at a time, as the class variable). The experiments also had the purpose of exploring the BC ability to try different class variables using the same model (instead of building a customized model for each variable). Fig. 13 shows the structure of the Alarm BN. Due to the reduced dimensions of this figure, variable names have been replaced by numbers following the convention described in (Cooper and Herskovits 1992). The variables originally named *Hypovolemia, LVFailure, Anaphylaxis, Insufficient Anesthesia, PulmEmboulus, Intubation, KikedTube* and *Disconnect* are represented by numbers 17, 18, 19, 20, 21, 22, 23 and 24 respectively.

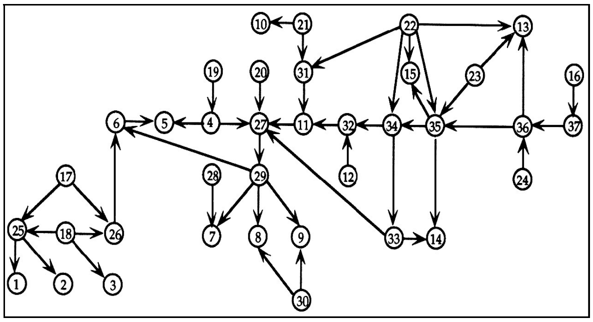


Fig. 13 Bayesian Network structure representing the Alarm problem

As briefly mentioned in Section 6.1, the MAP approach to extract classification rules from a BC takes into account the whole BC structure (variables and arcs). In the Alarm network this will produce over 2^{36} rules, each one with an antecedent part containing 36 variable tests. The BayesRule method, however, by using the Markov Blanket of the class feature, minimizes the number of rules as well as the number of variable tests in the antecedent part of each rule. Table 10 shows the variables used as class variables in the experiments conducted using the Alarm domain. To extract classification rules for each Alarm class variable, the BayesRule procedure was run eight times (one for each class) using the same Alarm network as input.

The Asia BN is a simple graphical model having 8 nodes and 8 arcs. It is commonly used in the literature to illustrate basic concepts of Bayesian networks in diagnosis and learning problems. It was first mentioned in (Lauritzen and Spiegelhalter 1988) and the name Asia came from the fact that, in this BN, there is a node (considered the class variable in the experiments) which models whether an individual has recently visited Asia, which is considered to be a risk factor in tuberculosis. Fig. 14 depicts the Asia network structure.

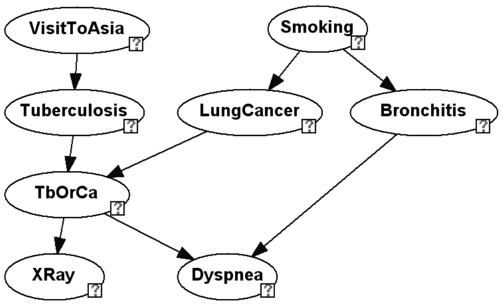


Fig. 14 Bayesian network structure representing the Asia domain

As described in the GeNIe[1] software (Druzdzel 1999), the Credit BN is a simple network for assessing credit worthiness of an individual. The node *CreditWorthiness* is of interest to the user and as such should be assigned as the class variable. All parentless nodes are described by uniform distributions; this is a weakness of the model, although it can be compensated by the fact that most of the time all the corresponding variables will be observed and the network will compute the probability distribution over credit worthiness correctly. Fig. 15 shows the Credit BN structure.

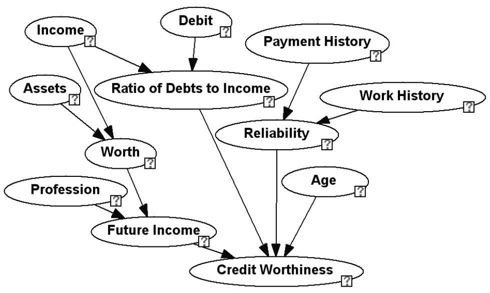


Fig. 15 Bayesian network structure representing the Credit domain

Also in GeNIe software, the Engine Fuel System (Engine) BN describes a simple diagnostic domain of a vehicle fuel system. It has 9 nodes and was created to verify whether the "Fuel Filters and Bypass Valves" are defective or not. Fig. 16 shows the BN structure.

The Win95pts BN was created to be used as an expert system for printer troubleshooting in Windows 95. It was developed at the Microsoft Research Center and was part of the Lumiere Project (Horvitz *et al.* 1998) at Microsoft Research that was initiated in 1993 with the goal of developing methods and an architecture for reasoning about the goals and needs of software users as they

[1] GeNIe modeling environment developed by the Decision Systems Lab. of the University of Pittsburgh (http://www.sis.pitt.edu/~dsl).

work with software. At the heart of the Lumiere are Bayesian models that capture the uncertain relationships between the goals and needs of a user and observations about a program state, sequences of actions over time, and words in a user's query (when such a query has been made). Fig. 17 depicts the Win95pts BN structure.

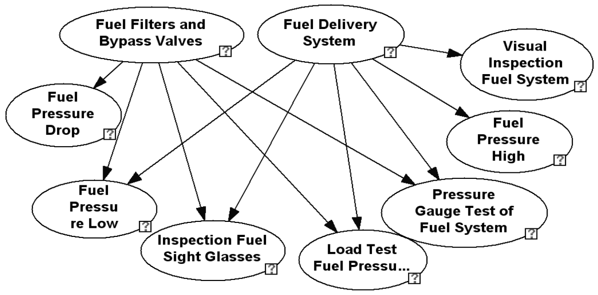

Fuel Filters and Bypass Valves
Fuel Delivery System
Visual Inspection Fuel System
Fuel Pressure Drop
Fuel Pressure High
Fuel Pressu re Low
Pressure Gauge Test of Fuel System
Inspection Fuel Sight Glasses
Load Test Fuel Pressu...

Fig. 16 Bayesian network structure representing the Engine Fuel System domain

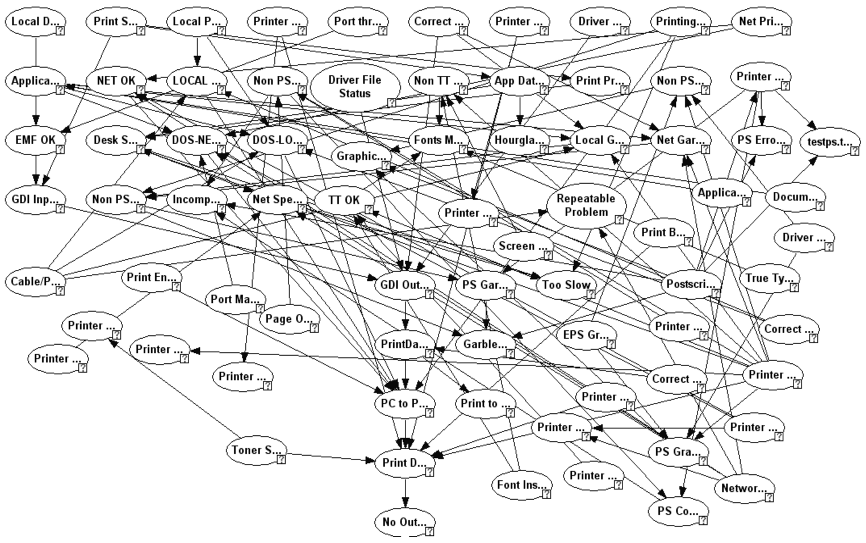


Fig. 17 Bayesian network structure representing the Win95pts domain

Table 10 shows the variables used as class variables in the experiments conducted using the Win95pts domain. Two variables (one at a time) were used as class variables, namely the "Repeatable Problem" and the "Driver File Status".

As with Alarm BN, the Win95pts BN was not used to classify a single variable. Using the Win95pts BN, however, only two variables (one at a time) were considered as class, namely the "Repeatable Problem" and the "Driver File Status" variables.

Two artificial domains named Synthetic 1 (Syn_1) and Synthetic 2 (Syn_2) were simulated in order to verify the behavior of BayesRule. Such simulations were performed manually by building BNs to encode a joint probability distribution over a set of random variables and, thus, reproducing hypothetical circumstances. Two synthetic BNs also named Synthetic 1 (Syn_1) and Synthetic 2 (Syn_2) were built and are depicted in Fig. 18.

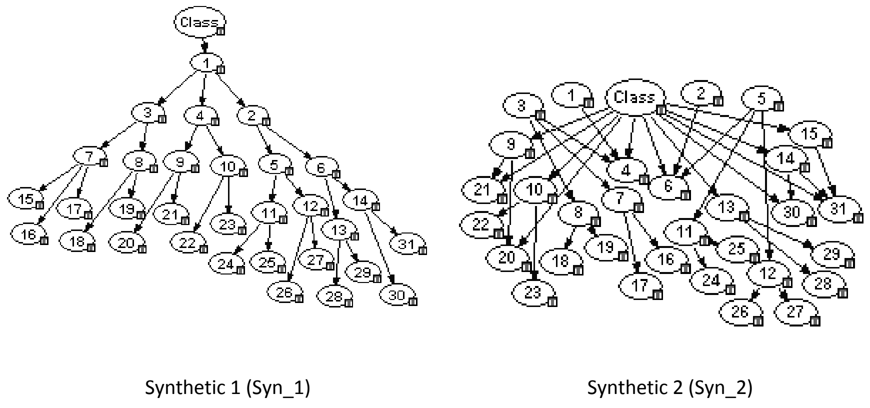


Fig. 18 Bayesian networks representing Synthetic 1 and Synthetic 2 domains

Syn_1 BN represents a domain with 32 variables where only one variable directly influences the class variable (the MB of the class variable has only one variable) and all variables have at most one parent. Therefore, the Syn_1 structure represents a polytree which is a suitable structure to verify the behavior of a classifier in problems where variables have simple interdependencies relationships (Pearl 1988). The BN was created to simulate a situation that favors BayesRule. Considering that the BN has 32 binary variables and only one is present in MB of the class, BayesRule should generate only two rules, each having a single variable test in its antecedent part.

Syn_2 BN describes a domain having 32 variables with 14 variables directly influencing the class variable. In this BN, each variable has 3 parents at most and this fact allows the establishing of more complex interdependency relationships among variables than the polytree structures. Therefore, Syn_2 is a lesser restrictive model than Syn_1. This BN was created in an attempt to simulate a situation that does not favor BayesRule. Considering that the BN has 32 binary variables and that 14 of them are in the MB of the class, BayesRule should generate 2^{14} rules having 14 variable tests in their antecedents. This illustrates a situation where the use of BayesRule is not recommended. In scenarios like that BayesRule should be used with a pruning mechanism (such as confidence and coverage). For measuring the accuracy of the set of generated rules from each BN, a testing set containing 10,000 patterns was created using the GeNIe software.

For a more robust comparative analysis, besides presenting the classification results (ACCRs) obtained using BayesRule, this section also shows the performance of the traditional decision tree based C4.5 algorithm (Quinlan 1993) (using the WEKA data mining environment (Witten and Frank 2000)) in the same domains; the obtained C4.5 trees were translated into sets of classification rules. It is important to remind that an unique BN can be used to extract classification rules having any of involved variables as consequent. Thus, when using BayesRule, a single BN was induced for each dataset. When extracting classification rules from decision trees, however, one particular decision tree will give rise to rules having one specific variable as consequent. Therefore, for the Alarm domain 8 different decision trees (one for each class variable) were induced and for the Win95pts domain, two different decision trees were induced. The performance results of BayesRule and C4.5 are presented in Table 11.

Table 11 Results obtained using BayesRule and C4.5 without pruning. A: Alarm, W:Win95pts and ACCR: Average Correct Classification Rate

Domain		Number of rules		Max. number of variable tests per rule		ACCR(%)	
		BayesRule	C4.5	BayesRule	C4.5	BayesRule	C4.5
A	Hypovolemia (17)	18	151	3	24	98.36	98.96
	LVFailure (18)	36	101	4	16	99.02	99.42
	Anaphylaxis (19)	3	35	1	10	98.97	99.02
	Insufficient Anesthesia (20)	54	795	4	22	88.01	90.68
	PulmEmboulus (21)	18	61	3	14	99.54	99.67
	Intubation (22)	8223	180	8	15	98.61	99.13
	KinkedTube (23)	192	114	4	15	99.19	99.42
	Disconnect (24)	16	89	2	25	98.98	99.23
Asia		2	1	1	1	98.98	98.98
Credit		24	3016	4	11	72.51	66.90
EngineFuelSystem		64	10	6	5	99.94	99.94
W	Repeatable Problem	4	96	2	14	98.22	98.42
	Driver File Status	2	47	1	28	98.23	98.23
Syn_1		2	550	1	18	89.16	81.86
Syn_2		16384	363	14	17	88.12	87.82

The ACCR values in Table 11 were obtained in a 10-fold cross-validation strategy and all the datasets used by both BayesRule and C4.5 were the same. Analyzing the results shown in Table 11 it is possible to observe that the ACCR values produced using either BayesRule or C4.5 set of rules are very similar. The only significant difference occurred in the Credit domain where BayesRule produced a more accurate rule set.

Focusing on the number of rules, however, it can be seen that results produced by BayesRule and C4.5 are not so similar. In ten out of fifteen classification experiments, BayesRule outperformed C4.5. For the class variable *Intubation* (22), BayesRule generated 8,223 rules having at most 8 variable tests each, while the decision tree based approach generated 180 rules having at most 15 variable tests each. The considerably high number of rules generated by BayesRule for the class variable 22 is not surprising. As the Markov Blanket of *Intubation* (22) has eight variables, the number of generated rules tends to be large. Thus, BayesRule may not be convenient when extracting classification rules for a variable having a large MB. The same situation happened with the Synthetic 2 domain, which confirmed this expected behavior. Based on these facts the following rule of thumb is suggested: a variable X may be not suitable for undergoing a 'translation into rules' process (using BayesRule) if $|MB(X)| \geq 6$.

In an attempt to improve the results obtained with variable 22 (from Alarm) and with the Synthetic 2 domain, a simple pruning strategy was implemented, that of removing rules containing superfluous variable tests. The C4.5 algorithm also implements a pruning procedure. Results using the pruned rule sets obtained with both algorithms are shown in Table 12. With pruning, BayesRule and C4.5 results have improved, since the number of rules has dropped in most of the experiments.

Table 12 Summary of the results obtained using BayesRule and C4.5 with pruning. A: Alarm, W:Win95pts and ACCR: Average Correct Classification Rate

Domain		Number of rules		Max. number of variable tests per rule		ACCR(%)	
		BayesRule	C4.5	BayesRule	C4.5	BayesRule	C4.5
A	Hypovolemia (17)	12	121	3	17	98.36	98.96
	LVFailure (18)	32	79	3	10	99.02	99.42
	Anaphylaxis (19)	1	29	0	7	98.97	99.02
	Insufficient Anesthesia (20)	34	678	4	22	88.01	90.68
	PulmEmboulus (21)	9	45	3	8	99.54	99.67
	Intubation (22)	1905	164	8	14	98.61	99.13
	KinkedTube (23)	84	83	4	9	99.19	99.42
	Disconnect (24)	13	79	2	20	98.98	99.23
Asia		2	1	1	1	98.98	98.98
Credit		20	114	4	11	72.51	72.39
EngineFuelSystem		34	10	6	5	99.94	99.94
W	Repeatable Problem	4	44	1	14	98.22	98.33
	Driver File Status	2	17	1	12	98.23	98.99
Syn_1		2	2	14	1	89.16	89.16
Syn_2		8056	363	14	17	88.12	87.92

The results displayed in Table 12 show that BayesRule generated considerably smaller rule sets than the C4.5 in nine out of fifteen experiments. Even after pruning the resulting rule set for the Alarm, variable 22 remains with 1,905 rules, and for Syn_2, with 8,056 rules. In both cases the number of rules is still considerably high when compared to the number of rules produced by C4.5. There is no enough evidence to state that one method is better than the other. One may conclude, however, that BayesRule is a consistent way of extracting relevant classification rules from a BN; specifically in the conducted experiments, it generated smaller rule sets, when compared to those generated by the C4.5. The use of the MB concept was crucial for simplifying the rule set, while maintaining accuracy. Taking into account the MB of the class variable, the maximum number of generated rules was substantially reduced.

It is important to mention that the number of variables in the Markov Blanket of the class variable is not the only criteria to identify the number of rules to be generated. This situation is illustrated in the Alarm domain with variable 20 and variable 24 as classes. Notice that, although |MB(20)|=4 and |MB(24)| = 2, BayesRule produced a rule set with 12 and 16 rules respectively. This is a consequence of the number of possible values each of these variables can have. Variables that have a greater number of possible associated values generate a higher number of combinations and, consequently, a greater number of rules. Results show that there are two main factors influencing the number of rules generated by BayesRule. One is the Markov Blanket of the class variable and the other, the number of possible values each variable in the Markov Blanket of the class variable has.

In spite of the motivating results, there are still some issues that should be further investigated. There is a possibility that a more elaborated pruning procedure, involving the concepts of confidence and coverage of rules, could improve the results. Another relevant aspect to be explored is related to the fact that the rules extracted from BCs may be in a causal context (Pearl 2000).

5.7 Conclusion

The main goal of this chapter was to show that BN is a sound formalism that has a broad use in many different machine leaning tasks, starting with pre-processing, followed by learning and finally contributing in post processing. Although the chapter describes a few methods related to the three main ML tasks, it is important to mention that (a) there are many approaches that have only been cited and (b) several other roles that Bayesian networks can play have not been addressed. In addition, all the discussed algorithms are based on BNs having only discrete variables. When continuous variables are employed BN variations such as Gaussian BNs (Neapolitan 2003) should be used.

One of the advantages of using the BN approach in all the three ML tasks is to maintain the same inductive bias throughout the whole sequence of ML steps i.e., preparing the data, learning and pruning. Although in particular domains there is a chance of this not being a particularly convenient feature, we believe that, in general, by maintaining the same bias, the whole three-step process can be more efficient.

Due to their similarities to BNs, other probabilistic graphical models such as Markov Networks (Pearl 1988) and Conditional Random Fields (Lafferty *et al.* 2001) can also play the same roles discussed in this paper in spite of not being able to represent causal relationships among variables as BN does.

Acknowledgments. This research was partially supported by Brazilian Research Agencies FAPESP (Fundação de Amparo à Pesquisa do Estado de São Paulo-Brazil) and CNPq (Conselho Nacional de Desenvolvimento Científico e Tecnológico-Brazil). We wish to express our gratitude to Dr. Sheela Ramanna for her insightful comments and suggestions concerning our manuscript.

References

Abellán, J., Gómez-Olmedo, M., Moral, S.: Some variations on the PC algorithm. In: Proc. of The 3rd European Workshop on Probabilistic Graphical Models (PGM 2006), Prague, pp. 1–8 (2006)

Anderson, R.L.: Missing plot techniques. Biometrics 2, 41–47 (1946)

Antal, P., Hullám, G., Gézsi, A., Millinghoffer, A.: Learning complex Bayesian network features for classification. In: Proc. of The 3rd European Workshop on Probabilistic Graphical Models, pp. 9–16 (2006)

Antal, P., Millinghoffer, A., Hullam, G., Szalai, C., Falus, A.: A Bayesian view of challenges in feature selection: multilevel analysis, feature aggregation, multiple targets, redundancy and interaction. In: Journal of Machine Learning Research: Workshop and Conference Proceedings, vol. 4, pp. 74–89 (2008)

Batista, G.E.A.P., Monard, M.C.: An analysis of four missing data treatment methods for supervised learning. Applied Artificial Intelligence 17(5-6), 519–534 (2003)

Beinlich, I., Suermondt, H.J., Chavez, R.M., Cooper, G.F.: The ALARM monitoring system: a case study with two probabilistic inference techniques for belief networks. In: Proc. of the 2nd European Conference on Artificial Intelligence in Medicine, London, UK, vol. 38, pp. 247–256 (1989)

Ben-Gal, I.: Bayesian networks. In: Ruggeri, F., Faltin, F., Kenett, R. (eds.) Encyclopedia of Statistics in Quality & Reliability. Wiley & Sons (2007)

Bilmes, J.: A gentle tutorial on the EM algorithm and its application to parameter estimation for Gaussian mixture and hidden Markov models. Technical Report, University of Berkeley, ICSI-TR-97-021 (1997)

Blum, A.L., Langley, P.: Selection of relevant features and examples in machine learning. Artificial Intelligence, 245–271 (1997)

Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: CART: Classification and Regression Trees. Chapman & Hall, Wadsworth (1983)

Bressan, G.M., Oliveira, V.A., Hruschka Jr., E.R., Nicoletti, M.C.: Using Bayesian networks with rule extraction to infer the risk of weed infestation in a corn-crop. Engineering Applications of Artificial Intelligence 22, 579–592 (2009)

Brown, L.E., Tsamardinos, I.: Markov blanket-based variable selection in feature space. Technical Report DSL TR-08-01, Department of Biomedical Informatics, Vanderbilt University (2008)

Chajewska, U., Halpern, J.Y.: Defining explanation in probabilistic systems. In: Proc. of Conference of Uncertainty in Artificial Intelligence, Providence, RI, pp. 62–71 (1997)

Cheng, J., Bell, D.A., Liu, W.: Learning belief networks from data: an information theory based approach. In: Proc. of The 6th ACM International Conference on Information and Knowledge Management, pp. 325–331 (1997)

Cheng, J., Greiner, R.: Comparing Bayesian network classifiers. In: Proc. of The 15th Conference on Uncertainty in Artificial Intelligence, pp. 101–107 (1999)

Cheng, J., Greiner, R.: Learning Bayesian Belief Network Classifiers: Algorithms and System. In: Stroulia, E., Matwin, S. (eds.) Canadian AI 2001. LNCS (LNAI), vol. 2056, pp. 141–151. Springer, Heidelberg (2001)

Cheng, J., Greiner, R., Kelly, J., Bell, D., Liu, W.: Learning Bayesian networks from data: an information-theory based approach. Artificial Intelligence 137(1), 43–90 (2002)

Chickering, D.M.: Learning Bayesian networks is NP-complete. In: Fisher, D., Lenz, A. (eds.) Learning from Data: Artificial Intelligence and Statistics V, pp. 121–130. Springer (1996)

Chickering, D.M.: Optimal structure identification with greedy search. Journal of Machine Learning Research 3, 507–554 (2002)

Cooper, G.F.: The computational complexity of probabilistic inference using Bayesian belief networks (research note). Artificial Intelligence 42(2-3), 393–405 (1990)

Cooper, G., Herskovitz, E.: A Bayesian method for the induction of probabilistic networks from data. Machine Learning 9, 309–347 (1992)

Cooper, G.F.: NESTOR: A computer-based medical diagnostic aid that integrates causal and probabilistic knowledge. PhD thesis, Medical Information Sciences, Stanford University, Stanford, CA (1984)

Dempster, A.P., Laird, N.M., Rubin, D.B.: Maximum likelihood from incomplete data via the EM algorithm. Journal of the Royal Statistical Society B 39, 1–39 (1977)

Díez, F.J., Mira, J., Iturralde, E., Zubillaga, S.: Diaval, a Bayesian expert system for echocardiography. Artificial Intelligence in Medicine 10(1), 59–73 (1997)

Duda, R.O., Hart, P.E.: Pattern classification and scene analysis. John Wiley & Sons (1973)

Druzdzel, M.J.: Qualitative verbal explanations in Bayesian belief networks. Artificial Intelligence and Simulation of Behaviour Quarterly 94, 43–54 (1996)

Druzdzel, M.J.: SMILE: Structural modeling, inference, and learning engine and GeNIe: A development environment for graphical decision-theoretic models. In: Proc. of the 16th National Conference on Artificial Intelligence, Orlando, FL, pp. 902–903 (1999)

Duch, W., Adamczak, R., Grabczewski, K.: A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. IEEE Transactions on Neural Networks 11(2), 1–31 (2000)

Fast, A., Jensen, D.: Constraint relaxation for learning the structure of Bayesian networks. Technical Report 09-18, Computer Science Department, University of Massachusetts, Amherst (2009)

Fayyad, U.M., Shapiro, G.P., Smyth, P.: From data mining to knowledge discovery: an overview. In: Fayyad, et al. (eds.) Advances in Knowledge Discovery and Data Mining, pp. 1–37. MIT Press (1996)

Frank, A., Asuncion, A.: UCI Machine Learning Repository. School of Information and Computer Science. University of California, Irvine (2010), `http://archive.ics.uci.edu/ml`

Friedman, N., Linial, M., Nachman, I., Pe'er, D.: Using Bayesian network to analyze expression data. Journal of Computational Biology 7, 601–620 (2000)

Friedman, N.: Inferring cellular networks using probabilistic graphical models. Science 303, 799–805 (2004)

Friedman, N., Geiger, D., Goldszmidt, M.: Bayesian network classifiers. Machine Learning 29, 131–163 (1997)

Friedman, N., Goldszmidt, M.: Building classifiers using Bayesian networks. In: Proc. of the AAAI 1996, vol. 2, pp. 1277–1284 (1996)

Friedman, H.F., Kohavi, R., Yun, Y.: Lazy decision trees. In: Proc. of the 13th National Conference on Artificial Intelligence, pp. 717–724. AAAI Press/MIT Press, Cambridge, MA (1996)

Fu, F.S., Demarais, M.C.: Markov blanket based feature selection: a review of past decade. In: Proc. of the World Congress on Engineering (WCE 2010), London, UK, pp. 321–328 (2010)

Ghahramami, Z., Jordan, M.: Learning from incomplete data. Technical Report AI Lab Memo no. 1509, CBCL paper no. 108. MIT AI Lab. (1995)

Guo, H., Hsu, W.: A survey on algorithms for real-time Bayesian network inference. In: Proc. of The AAAI-02/KDD-02/UAI-02 Joint Workshop on Real-Time Decision Support and Diagnosis Systems, Edmonton, Alberta, Canada (2002)

Guyon, I., Elisseeff, A.: An introduction to variable and feature selection. Journal of Machine Learning Research 3, 1157–1182 (2003)

Heckerman, D.: Bayesian networks for data mining. Data Mining and Knowledge Discovery Journal 1(1), 79–119 (1997)

Heckerman, D., Geiger, D.: Learning Bayesian networks: a uni. cation for discrete and Gaussian domains. In: Proc. 11th Conference on Uncertainty in Artificial Intelligence (UAI 1995), pp. 274–284 (1995)

Heckerman, D., Chickering, D.M., Meek, C., Rounthwaite, R., Kadie, C.: Dependency networks for inference, collaborative filtering, and data visualization. Journal of Machine Learning Research 1(1), 49–75 (2000)

Henrion, M., Druzdzel, M.J.: Qualitative propagation and scenario-based approaches to explanation of probabilistic reasoning. In: Proc. of 6th Conference on Uncertainty in Artificial Intelligence, Cambridge, MA, pp. 17–32 (1990)

Horvitz, E., Breese, J., Heckerman, D., Hovel, D., Rommelse, K.: The Lumiere project: Bayesian user modeling for inferring the goals and needs of software users. In: Proc. of the 14th Conference on Uncertainty in Artificial Intelligence, Madison, WI, pp. 256–265. Morgan Kaufmann, San Francisco (1998)

Hruschka Jr., E.R., Nicoletti, M.C., Oliveira, V., Bressan, G.: BayesRule: a Markov-blanket based procedure for extracting a set of probabilistic rules from Bayesian classifiers. Int. Journal of Hybrid Intelligent Systems 76(2), 83–96 (2008)

Hruschka, E.R., Garcia, A., Hruschka Jr., E.R., Ebecken, N.F.F.: On the influence of imputation in classification: practical issues. Journal of Experimental and Theoretical Artificial Intelligence 21, 43–58 (2009)

Hruschka Jr., E.R., Hruschka, E.R., Ebecken, N.F.F.: Bayesian networks for imputation in classification problems. Journal of Intelligent Information Systems 29, 231–252 (2007)

Hruschka Jr., E.R., Hruschka, E.R., Ebecken, N.F.F.: Feature Selection by Bayesian Networks. In: Tawfik, A.Y., Goodwin, S.D. (eds.) Canadian AI 2004. LNCS (LNAI), vol. 3060, pp. 370–379. Springer, Heidelberg (2004)

Hruschka Jr., E.R., Ebecken, N.F.F.: Missing values prediction with K2. Intelligent Data Analysis Journal (IDA) 6(6), 557–566 (2002)

Hruschka Jr., E.R., Ebecken, N.F.F.: Ordering attributes for missing values prediction and data classification. In: Data Mining III - Management Information Systems Series, 6th edn., WIT Press, Southampton (2002)

Hruschka, E.R., Hruschka Jr., E.R., Ebecken, N.F.F.: Evaluating a Nearest-Neighbor Method to Substitute Continuous Missing Values. In: Gedeon, T(T.) D., Fung, L.C.C. (eds.) AI 2003. LNCS (LNAI), vol. 2903, pp. 723–734. Springer, Heidelberg (2003)

Husmeier, D., Dybowski, R., Roberts, S. (eds.): Probabilistic modeling in bioinformatics and medical informatics. Springer, London (2005)

Inza, I., Larrañaga, P., Etxeberia, R., Sierra, B.: Feature subset selection by Bayesian networks based optimization. Artificial Intelligence 123(1-2), 157–184 (2000)

Inza, I., Larrañaga, P., Sierra, B.: Feature subset selection by Bayesian networks: a comparison with genetic and sequential algorithms. International Journal of Approximate Reasoning 27, 143–164 (2001)

Jansen, R., et al.: A Bayesian network approach for predicting protein-protein interactions from genomic data. Science 302, 449–453 (2003)

John, G., Kohavi, R., Pfleger, K.: Irrelevant features and the subset selection problem. In: Proc. of the 11th International Conference on Machine Learning, pp. 121–129 (1994)

Jordan, M., Xu, L.: Convergence results for the EM approach to mixtures of experts architectures. Neural Networks 8, 1409–1431 (1996)

Kalisch, M., Bühlmann, P.: Estimating high-dimensional directed acyclic graphs with the PC-algorithm. Journal of Machine Learning Research 8, 613–636 (2007)

Kohavi, R., Becker, B., Sommerfield, D.: Improving simple Bayes. In: van Someren, M., Widmer, G. (eds.) Poster papers of the ECML 1997, pp. 78–87. Charles University, Prague (1997)

Koller, D., Sahami, M.: Toward optimal feature selection. In: Proc. of the 13th International Conference on Machine Learning, pp. 284–292 (1996)

Kong, A., Liu, J.S., Wong, W.H.: Sequential imputations and Bayesian missing data problems. Journal of the American Statistical Association 89(425), 278–288 (1994)

Kononenko, I., Bratko, I., Roskar, E.: Experiments in automatic learning of medical diagnostic rules. Technical Report, Jozef Stefan Institute, Ljubjana (1984)

Lacave, C., Díez, F.: A review of explanation methods for Bayesian networks. The Knowledge Engineering Review 17(2), 107–127 (2002)

Lafferty, J., McCallum, A., Pereira, F.: Conditional random fields: probabilistic models for segmenting and labeling sequence data. In: Proc. 18th International Conference on Machine Learning, pp. 282–289. Morgan Kaufmann, San Francisco (2001)

Lam, W., Bacchus, E.: Using causal information and local measures to learn Bayesian networks. In: Proceedings of 9th Conference on Uncertainty in Artificial Intelligence, Washington, DC, pp. 243–250 (1993)

Langley, P., Iba, W., Thompson, K.: An analysis of Bayesian classifiers. In: Proc. of the AAAI 1992, pp. 223–228 (1992)

Langley, P., Sage, S.: Induction of selective Bayesian classifiers. In: Proc. of the 10th Conference on Uncertainty in Artificial Intelligence, pp. 399–406. Morgan Kaufmann Publishers, Seattle (1994)

Lauritzen, S.L.: Some modern applications of graphical models. In: Green, P.J., Hjort, N.L., Richardson, S. (eds.) Highly Structured Stochastic Systems. Oxford University Press (2003)

Lauritzen, S., Spiegelhalter, D.: Local computations with probabilities on graphical structures and their application to expert systems. Journal of the Royal Statistical Society B 50, 157–224 (1988)

Little, R., Rubin, D.B.: Statistical analysis with missing data. John Wiley & Sons, New York (1987)

Liu, H., Motoda, H.: Feature selection for knowledge discovery and data mining. Kluwer Academic (1998)

Lobo, O.O., Noneao, M.: Ordered estimation of missing values for propositional learning. Journal of the Japanese Society for Artificial Intelligence 15(1), 162–168 (2000)

Madden, M.G.: Evaluation of the performance of the Markov blanket Bayesian classifier algorithm. Technical Report No. NUIG-IT-011002, NUI Galway, Ireland (2002)

Mitchell, T.: Machine learning. The McGraw-Hill Companies, Inc. (1997)

Moore, A.: Data Mining Tutorials (2011), `http://www.autonlab.org/tutorials/`

Murphy, K.: A brief introduction to graphical models and Bayesian networks (1998), `http://www.cs.ubc.ca/~murphyk/Bayes/bnintro.html`

Neapolitan, R.E.: Learning Bayesian networks. Prentice Hall (2003)

Nicoletti, M.C.: The feature subset selection problem in machine learning – Talk presented at The Seventh International Conference on Intelligent Systems Design and Applications, Rio de Janeiro, Brazil (2007) (unpublished)

Pearl, J.: Probabilistic reasoning in intelligent systems: networks of plausible inference. Morgan Kaufmann Publishers, San Mateo (1988)

Pearl, J.: Causality: models, reasoning, and inference. Cambridge University Press (2000)

Pourret, O., Nai, P., Marcot, B.: Bayesian networks: a practical guide to applications. Wiley, Chichester (2008)

Preece, A.D.: Iterative procedures for missing values in Experiments. Technometrics 13, 743–753 (1971)

Pyle, D.: Data preparation for data mining. Academic Press, San Diego (1999)

Quinlan, J.R.: C4.5 program for machine learning. Morgan Kaufmann, San Francisco (1993)

Quinlan, J.R.: Induction of decision trees. Machine Learning 1(1), 81–106 (1986)

Redner, R., Walker, H.: Mixture densities, maximum likelihood and the EM algorithm. SIAM Review 26(2), 195–239 (1984)

Reunanen, J.: Overfitting in making comparisons between variable selection methods. Journal of Machine Learning Research 3, 1371–1382 (2003)

Rubin, D.B.: Inference and missing data. Biometrika 63, 581–592 (1976)

Rubin, D.B.: Formalizing subjective notion about the effects of nonrespondents in samples surveys. Journal of the American Statistical Association 72, 538–543 (1977)

Rubin, D.B.: Multiple imputation for non-responses in surveys. John Wiley & Sons, New York (1987)

Russel, S., Norvig, P.: Artificial intelligence: a modern approach. Prentice Hall Series in Artificial Intelligence (1995)

Sachs, K., Perez, O., Pe'er, D., Lauffenburguer, D.A., Nolan, G.P.: Causal protein-signaling networks derived from multiparameter single-cell data. Science 308, 523–529 (2005)

Santos, E.B., Hruschka Jr., E.R., Nicoletti, M.C.: Conditional independence based learning of Bayesian classifiers guided by a variable ordering genetic search. In: Proc. of CEC 2007, vol. 1, pp. 1–10. IEEE Press, Los Alamitos (2007)

Schllimmer, J.C.: Concept acquisition through representational adjustment. Doctoral Dissertation, Department of Information and Computer Science. University of California, Irvine (1987)

Schafer, J.L.: Analysis of incomplete multivariate data. Chapman & Hall/CRC, Boca Raton (2000)

Schafer, J.L., Graham, J.W.: Missing data: our view of the state of the art. Psychological Methods 7(2), 147–177 (2002)

Sebastiani, P., Yu, Y.-H., Ramoni, M.F.: Bayesian machine learning and its potential applications to the genomic study of oral oncology. Advances in Dental Research 17, 104–108 (2003)

Spiegelhalter, D.J., Lauritzen, S.L.: Sequential updating of conditional probability on direct graphical structures. Networks 20, 576–606 (1990)

Spirtes, P., Glymour, C., Scheines, R.: Causation, predication, and search. Springer, New York (1993)

Spirtes, P., Meek, C.: Learning Bayesian networks with discrete variables from data. In: KDD 1995, pp. 294–299 (1995)

Suzuki, J.: A construction of Bayesian networks from databases based on an MDL scheme. In: Proc. of 9th Conference on Uncertainty in Artificial Intelligence, Washington, DC, pp. 266–273 (1993)

Tanner, M.A., Wong, W.H.: The calculation of posterior distributions by data augmentation (with discussion). Journal of the American Statistical Association 82, 528–550 (1987)

Troyanskaya, O.G., Cantor, M., Sherlock, G., Brown, P., Hastie, T., Tibshirani, R., Botstein, D., Altman, R.B.: Missing value estimation methods for DNA microarrays. Bioinformatics 17(6), 520–525 (2001)

White, A.P.: Probabilistic induction by dynamic path generation in virtual trees. In: Bramer, M.A. (ed.) Research and Development in Expert Systems III, pp. 35–46. Cambridge University Press (1987)

Witten, I.H., Frank, E.: Data mining – practical machine learning tools and techniques with Java implementations. Morgan Kaufmann Publishers, USA (2000)

Wu, C.F.J.: On the convergence properties of the EM algorithm. The Annals of Statistics 11(1), 95–103 (1983)

Zeng, Y., Luo, J., Lin, S.: Classification using Markov blanket for feature selection. In: Proc. of The International Conference on Granular Computing (GrC 2009), pp. 743–747 (2009)

Zio, M.D., Scanu, M., Coppola, L., Luzi, O., Ponti, A.: Bayesian networks for imputation. Journal of the Royal Statistical Society, Series A (Statistics in Society) 167(2), 309–322 (2004)

Chapter 6
Evolving Intelligent Systems: Methods, Algorithms and Applications

Andre Lemos, Walmir Caminhas, and Fernando Gomide

Abstract. Evolving intelligent systems (EIS) are highly adaptive systems able to update its own parameters and structure based on a date stream. These systems have been developed to address problems of modeling, control, prediction, classification and data processing in a nonstationary, dynamic changing environment. Pioneers works in this area are dated from the around the turn of the centuries and were focused in areas of neural networks, fuzzy rule-based systems and neural-fuzzy hybrids. In this century the area has been expanded to also address statistical models, hardware implementations and so on. The aim of this chapter is to provide an introduction and a state of the art view about this subject. The purpose is to present the paradigm and the associated concepts, address the main learning approaches, and detail recently developed models based on participatory learning and fuzzy trees.

6.1 Introduction

The continuous increase in availability of large amounts of data has motivated considerable research to develop new online algorithms to process data streams. Mining fast-moving, quickly changing data streams brings unique problems and requires considerable effort. One of the most important issues in online data mining concerns the intrinsic nonstationarity of the data streams. For instance, in industry, machines suffer from stresses, aging, and

Andre Lemos · Walmir Caminhas
Departament of Eletronic Engineering, Universidade Federal de Minas Gerais, Brazil
e-mail: {andrepl,caminhas}@cpdee.ufmg.br

Fernando Gomide
Department of Computer Engineering and Automation, University of Campinas, Brazil
e-mail: gomide@dca.fee.unicamp.br

faults; in economic systems, performance indicators and stock indices vary; in communication systems, parameters and conditions of transmission media are also subject to continuous changes. This increasing interest in adaptive system modeling has motivated the development of highly adaptive and intelligent systems, denominated evolving intelligent systems (EIS), whose models are self-developed from a stream of data.

EIS are able to address problems of modeling, control, prediction, classification, data processing and feature selection in a non-stationary, dynamic changing environment. Such systems embody online learning methods and one-pass incremental algorithms that evolve or gradually change individual models to guarantee life-long learning and self-organization of the system structure.

Pioneers works in this area are dated from around the turn of the centuries and were focused in the areas of neural networks [20, 57], fuzzy rule-based systems [5, 44, 38] (evolving fuzzy systems) and neural-fuzzy hybrids [31, 40]. In the last 10 years, the area has been expanded to also address statistical models [24], granular computing [34], hardware implementations, etc. Figure 1 illustrates different types of evolving intelligent systems, focusing on evolving fuzzy systems. Real world applications have already been reported in areas such as intelligent sensors and actuators [7, 8, 45], autonomous unmanned systems [61, 9], process monitoring & control [19, 56, 43, 35], biomedical data processing, etc [29].

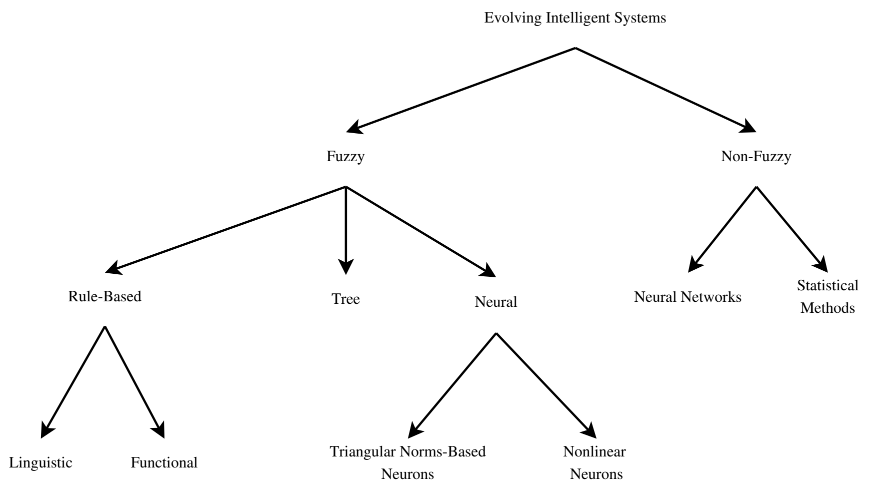


Fig. 1 Different types of EIS

The term “evolving” was used to differ from adaptive systems. EIS are able to adapt not only its parameters (something generally attributed to the term “adaptive”) but also its structure based on a data stream. Thus, the term “evolving” was employed to define a higher level of adaptation, where the topology of the model is not fixed [10].

One of the first models proposed is the evolving Takagi-Sugeno (eTS) [5]. This model is a functional fuzzy rule-based system, whose structure continuously evolves based on clusters created/excluded by a recursive version of the subtractive clustering algorithm. The consequent parameters are updated with the recursive least squares algorithm and its variations. Several other similar approaches can be found in literature [31, 5, 6, 44, 41] containing a similar topology (rule-based systems) and differing on the clustering mechanism used to evolve the rule-base. These models also propose improvements in the model structure adaptation, like, for instance, proposing pruning algorithms to avoid overfitting.

Recently an evolving rule-based model was developed from the idea of Participatory Learning named evolving Participatory Learning (ePL) [41]. Participatory learning is a learning paradigm, which assumes that learning and beliefs about the system to be modeled depend on what the learning process has already learned. This learning mechanism is ideal for developing of evolving systems because it balances learning and changing while still respecting the accumulated knowledge.

The ePL model has been extended recently by a fuzzy rule-based system with multivariable membership functions named evolving Multivariable Gaussian (eMG) [38]. This model accounts for the possibility that input variables may interact, avoids the curse of dimensionality when handling clusters formation, and introduces a sound and systematic approach for learning that results in an algorithm with few parameters.

Fuzzy rule-based models are not the only candidate for EIS. Recently an evolving tree structure modeling approach was introduced [37], denominated evolving Fuzzy linear regression Tree (eFT). The approach concerns a fuzzy linear regression tree whose topology can be continuously updated using a statistical model selection test. A fuzzy linear regression tree is a fuzzy tree with linear model in each leaf.

The aim of this chapter is to provide an introduction and the state of the art view on intelligent evolving modeling, focusing on evolving fuzzy systems. The purpose is to present the paradigm and the associated concepts, address the main learning approaches, and detail recently developed models based on participatory learning and fuzzy trees.

The chapter proceeds as follows. Section 6.2 reviews the idea of evolving fuzzy systems, describing one of the pioneering works in this area, the eTS model, and briefly describing other similar evolving models, presenting some concepts and learning approaches developed for incremental model structure adaptation. The next section details the eMG model [38], based on participatory learning, describing the model structure and learning algorithm. Section

6.4 describes a new recently proposed approach [37] for evolving intelligent modeling, defined as an evolving fuzzy linear regression tree (eFT model). Next, section 6.5 focuses on applications of the described models in time series forecasting modeling. Finally, the conclusions and further developments are summarized in section 6.6.

6.2 Evolving Fuzzy Systems

The idea of flexible systems, capable of adapt not only its parameters but also is structure automatically, date from the 90's, when some works proposed adaptive neural networks [20, 57, 33], with dynamic changing topologies. The first evolving fuzzy systems proposed on literature are dated from the beginning of this century [11, 31]. These models were developed in order to address a growing need for flexible, adaptive and interpretable models, used for developing intelligent sensors, autonomous systems, etc. Evolving fuzzy systems have advantages over other evolving black-box models, like evolving neural networks, since they are linguistically interpretable. It is possible to extract information from the model topology, as information granules (linguistic terms represented by fuzzy sets) [3].

There are several works in literature proposing evolving fuzzy models capable of addressing problems like system identification, time series forecasting, pattern classification and so on. Many of these works [4, 11, 31, 5, 6, 44, 41] propose functional fuzzy models, based on a set of Takagi-Sugeno rules [54]. The proposed models have an adaptive structure, that is, the number of rules and the antecedents fuzzy sets are defined at each iteration (for each new data sample processed) based on unsupervised recursive clustering algorithms [47, 23]. Figure 2 details the mechanism used for updating the model structure. The remaining parameters of these models, that is, the consequent parameters are updated using recursive least squares or its variations [59, 42].

In order to illustrate such models, the evolving Takagi-Sugeno model (eTS) is detailed. This model is one of th pioneers works proposing an evolving fuzzy system. It was initially proposed in [11] and further improved in [5].

6.2.1 Evolving Takagi-Sugeno (eTS)

The eTS model is composed by a set of functional rules:

$$R_i \;:\; \text{If } x_1 \text{ is } A_{i1} \text{ and } \cdots \text{ and } x_m \text{ is } A_{im} \text{ then } y_i = a_{i0}x_1 + \cdots + a_{im}x_m \quad (1)$$

where R_i is the ith fuzzy rule, for $i = 1, \cdots, g^k$, g^k is th number of fuzzy rules at iteration k, x_j for $j = 1, \cdots, m$ are the m input variables, A_{ij} are the antecedent fuzzy sets, y_i are the consequent linear model outputs and a_{ij} are the linear model parameters associated with the ith rule.

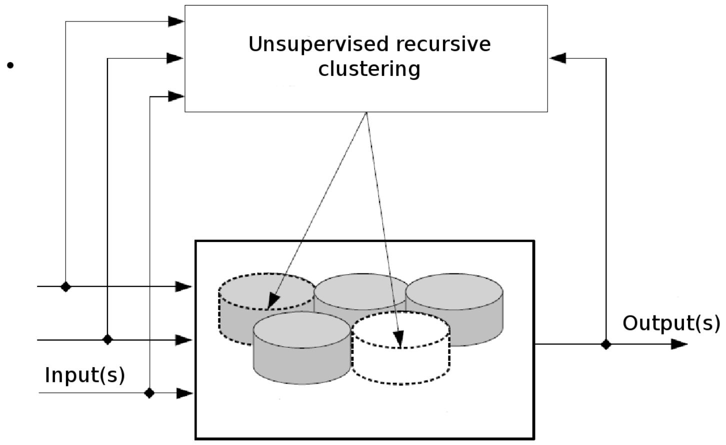


Fig. 2 Rule base updating mechanism

The antecedent fuzzy sets are defined using Gaussian membership functions:

$$\mu_{ij} = \exp\left(-\frac{4}{r^2}||x_j - x^*_{ij}||^2\right) \tag{2}$$

where r is a parameter (positive value) that defines membership functions spread and the influence zone of the model present on the ith rule, $||.||$ is the Euclidean norm and x^*_i is the focal point of the fuzzy set.

The firing degree of each rule is defined as the aggregation, using the product t-norm [48], of the fuzzy sets membership values:

$$\tau_i = \mu_{i1}(x_1) \times \mu_{i2}(x_2) \times \cdots \times \mu_{im}(x_m); \tag{3}$$

The model output is defined as the weighted average of the linear models present on all rules:

$$y = \sum_{i=1}^{g^k} \lambda_i y_i \tag{4}$$

where $\lambda_i = \tau_i / \sum_{j=1}^{g^k} \tau_j$ is the normalized firing degree and y_i is the linear model output of the ith rule, respectively.

An unsupervised recursively clustering algorithm is used to adapt the model rule-base at each iteration, that is, for each new data sample presented. The clustering is performed on the input-output space $z = [x^T \;\; y]^T$. At each iteration, the existing clusters are projected on each input variable space generating the antecedents fuzzy sets of the rules. Figure 3 illustrates the cluster projection method used for defining the fuzzy sets for each input variable.

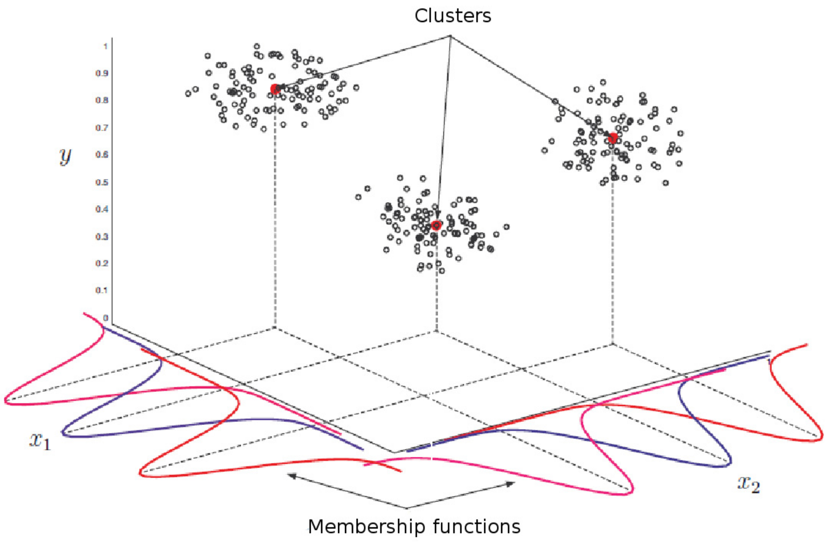


Fig. 3 Fuzzy sets generated by cluster projection

The clustering algorithm updates the cluster structure at each iteration. For each new data sample, the algorithm may create a new cluster or simply update the parameters of an existing cluster.

The eTS uses an incremental version of the subtractive clustering algorithm [15]. This clustering algorithm considers each data sample as a potential cluster center, and defines a measure of the *potential* of each data sample [15].

The potential of a data sample z^k is a measure of its distance to all other data samples:

$$P(z^k) = \frac{1}{k-1}\sum_{i=1}^{k-1}\exp\left(-r||z^k - z^i||^2\right) \tag{5}$$

where $k = 2, 3, \cdots$ is the index of the data samples already processed.

The potential function is used to find data samples that can be defined as centers of regions with high data density, as illustrated on Figure 4. In this figure, A has a lower potential than B. Analysing (5) and Figure 4, one can note that, the potential of a given data sample is proportional to the number of data samples present on a given region of the clustering space.

In [5] a recursive equation is deduced, used to compute the potential of a new data sample:

$$P^k(z^k) = \frac{k-1}{(k-1)(\vartheta^k + 1) + \sigma^k - 2\nu^k} \tag{6}$$

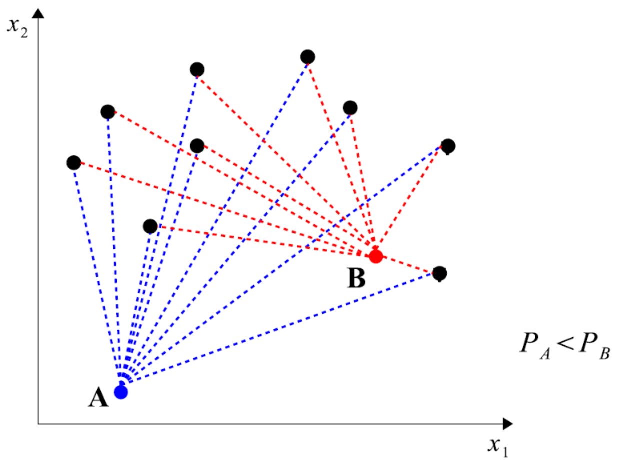


Fig. 4 Illustration of the data samples potential

where $\vartheta^k = \sum_{j=1}^{m+1}(z_j^k)^2$, $\sigma^k = \sum_{l=1}^{k-1}\sum_{j=1}^{m+1}(z_j^l)^2$ and $\nu^k = \sum_{j=1}^{m+1} z_j^k \beta_j^k$, where $\beta_j^k = \sum_{l=1}^{k-1} z_j^l$.

The parameters ϑ^k and ν^k are computed using z^k, β_j^k and σ^k can be computed recursively [5].

A recursive equation for updating the potential of the center of an existing cluster is also proposed in [5]:

$$P^k(z_l^*) = \frac{(k-1)P^{k-1}(z_l^*)}{k-2+P^{k-1}(z_l^*)+P^{k-1}(z_l^*)\sum_{j=1}^{m+1} d_j^{k(k-1)}} \tag{7}$$

where z_l^* is the center of the cluster l $(1 \times m+1)$ and $d_j^{k(k-1)} = z_j^k - z_j^{k-1}$.

For each new data sample, its potential is computed and the cluster centers potentials are updated. If the potential of a new data sample is greater than the potential of all existing cluster centers, the new data sample is defined as a cluster center. If the data sample is close enough to a existent cluster, the data sample replaces the cluster center. Otherwise, the data sample is defined as the center of a new cluster.

If the new data sample potential is less than all cluster centers potentials, the parameters of the consequent of the rule associated the closest cluster are updated using the recursive least squares algorithm or the recursive weighted least squares algorithm [59, 42].

The algorithm 1 summarizes the eTS learning procedure.

Algorithm 1. eTS learning algorithm

```
1: Compute the new data sample potential $P(z^k)$
2: for $j = 1, \cdots, g^k$ do
3:    Compute the center $c^j$ potential
4: end for
5: if $P(z^k) > P(c^j)\ \forall j$ then
6:    if $z^k$ is close enough to some cluster $j$ then
7:       $z^k$ replaces $c^j$ as the center of cluster $j$
8:    else
9:       A new cluster is created centered in $z^k$
10:   end if
11: else
12:   Update the consequent parameters of the closest cluster
13: end if
```

6.2.2 Other Evolving Fuzzy Models

Several other models can be found on literature with topology and learning algorithm similar with eTS. These models differ from eTS, mostly, by the clustering algorithm used to update the rule-base at each iteration. Some models also present new concepts used during the inference process and/or during rule-base update, for instance, methods for ignoring or removing inactive rules.

The DENFIS (Dynamic Evolving Neural-Fuzzy Systems) [31] is a model similar with eTS, where the rule-base is defined using a recursive clustering algorithm based on the Euclidean distance between new data samples and the existing clusters, denominated ECM (Evolving Clustering Method). A new cluster is created if the distance between a new data sample and all existing cluster centers is greater than a given threshold, defined as an algorithm parameter. The clustering is performed only on the input space and the center of each cluster is defined as the cluster sample mean vector (and not by a particular data sample, as eTS does). This model uses triangular membership functions, and because of that, a modified version of the Takagi-Sugeno inference process is used. For each new data sample, only the m rules associated with the m closest clusters are used for estimating the corresponding output.

A simplified version of eTS was suggested in [6] to reduce the complexity of potential calculations. The Simpl_eTS replaces the notion of information potential by the concept of *scatter* to provide a similar, but computationally more effective algorithm. This model also uses Cauchy type membership functions, instead of Gaussian ones. Moreover, Simpl_eTS proposes a methodology for ignoring inactive rules at each iteration. The *population* of each rule is computed recursively and expresses the number of data samples that are represented by cluster. The population is updated as:

$$POP^j = POP^j + 1; \;\; j = \arg\min_j ||z^k - z_j^*||^2 \tag{8}$$

The population of each cluster is constantly monitored. For each iteration, if the population of a given cluster represents less than 1% of all data samples already processed, the rule is ignored by setting its fire degree to 0.

In [10] an extended version of eTS is proposed capable of updating the radius of influence (r parameter of eTS) recursively based on the data samples. This model also defines a new measure of the cluster quality, denominated *age*. The cluster age is defined as the number of samples already processed minus the average sum of the time indices of the data samples:

$$AGE^j = k - \frac{2A^j}{k+1} \tag{9}$$

where $A^j = \sum_{i=1}^{POP^j} idx^j$ denotes the accumulated time of arrival and idx^j is the time instant when the ith data sample was processed.

This index assumes values in $(0,k]$. Values close to 0 indicates that recent data is being processed by a young cluster and values close to k indicates that no data is being processed by an old cluster. Old clusters can be replaced by data samples that have high increment of the potential [10].

The FLEXFIS (Flexible Fuzzy Inference System), proposed in [44], uses a recursive clustering algorithm based on an incremental version of the vector quantization technique [21], denominated eVQ (Evolving Vector Quantization). Similar with DENFIS, this model also uses a distance threshold for creating new clusters. However, this threshold is defined considering the input sample dimension, in order to avoid the *curse of dimensionality.* This is because the higher the space dimension, the greater the distance between two adjacent data points [22]. Thus, if the threshold does not accounts for the dimension, then as the dimensionality increases more observations will have the corresponding distances exceeding this threshold. Therefore more clusters are created, the model becomes more complex and over-fitting may occur.

The FLEXFIS distance threshold used for creating new cluster is:

$$\rho = \text{fac}\frac{\sqrt{m+1}}{\sqrt{2}} \tag{10}$$

where m is the input space dimension and fac is a parameter that must be tunned for each problem. How to adjust fac automatically is still an open question [44].

The FLEXFIS recognizes the problem of the curse of dimensionality and proposes a solution. However, this solution depends of a parameter that must be adjusted based on a priori information or cross-validation.

The SOFMLS (Self-Organizing Fuzzy Modified Least-Squares Network) [53] employs an evolving nearest-neighborhood clustering algorithm. This model is based on zero-order Takagi-Sugeno rules:

$$R_i \ : \ \text{If } x_1 \text{ is } A_{i1} \text{ and } \cdots \text{ and } x_m \text{ is } A_{im} \text{ then } y_i = v_i \tag{11}$$

where v_i is a constant.

Since SOFMLS uses zero-order rules, its topology can be represented as a two layer feed-forward network. The first layer computes the rules firing degrees and the output layer computes the model output. Figure 5 illustrates the model topology. This model also proposes a methodology for removing inactive rules based on the concept of rule density, similar with the concepts proposed by Simpl_eTS and xTS.

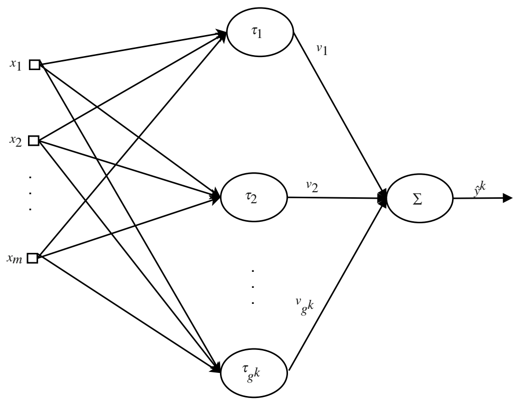


Fig. 5 SOFMS topology

The SAFIS (Sequential Adaptive Fuzzy Inference System) also uses zero-order Takagi-Sugeno rules and can also be represented as a feed-forward networks. This model uses the concept of *influence* of a fuzzy rule in order to adapt the rule-base, creating or excluding rules. The influence of a rule is defined as the contribution of a given rule for the model output.

Several other evolving fuzzy models can be found in literature with a feed-forward network topology [18, 40, 30]. These models have a multilayer topology than can be interpreted as a set of functional fuzzy rules. Learning algorithms are proposed for updating the network weights (that can be interpreted as antecedent and/or consequent parameters of functional fuzzy rules) and to grow/prune neurons (that can be interpreted as parts of fuzzy rules).

The recursive clustering algorithms used in most of the eFS found in literature can be summarized by a two-step procedure. First, a distance measure is defined and an initial number of clusters (and respective cluster centers) estimated from *a priori* knowledge about the problem; alternatively, a single cluster with the center at the first data sample is created. Second, for each new data sample, the distance between the existing clusters and the sample is computed and if the distance exceeds a threshold, then a new cluster is created; otherwise, the parameters of the closest cluster are updated using recursive algorithms [31, 18, 40, 44, 53]. Despite their effectiveness, the clustering algorithms constructed upon this two-step procedure have a major drawback in the sense that they lack robustness to noisy data. Whenever noisy data or outliers exceed a threshold, the two-step algorithms create new clusters instead of reject or smooth the data. This procedure may lead to overfitting, since it may generate an excessive number of clusters, increasing the model complexity.

A robust evolving TS system was developed in [41] using a recursive clustering algorithm inspired by the idea of participatory learning [58]. Participatory learning is a learning paradigm which assumes that learning and beliefs about the system to be modeled depend on what the learning process has already learned. An essential characteristic of this learning process is that a new observation impact in causing learning or belief revision depends on its compatibility with the current system belief. Therefore, clustering algorithms based on this learning process tend to be robust to noisy data because single outliers are likely to be incompatible with the current system belief and consequently can be either discarded or have their effect smoothed.

6.3 Evolving Multivariable Gaussian

This section describes the evolving fuzzy model proposed in [38], denominated eMG (evolving Multivariable Gaussian). The eMG uses an evolving Gaussian clustering algorithm rooted in the concept of participatory learning [58] in order to adapt the rule base for each input sample. At each iteration the clustering approach is used to define the rule base, that is, at each iteration a rule can be created, extracted or two rules can be merged.

The clustering procedure used by eMG considers the possibility that input variables may interact with each other. Clusters are estimated using a normalized distance measure (similar to the *Mahalanobis* distance) and trigger ellipsoidal clusters whose axes are not necessarily parallel to the input variables axes, as it would be the case if the Euclidean distance were used [31, 44, 5]. The idea is to preserve information about interactions between input variables. The fuzzy sets of the rules antecedents are multivariable Gaussian membership functions characterized by a center vector, and a dispersion matrix representing the dispersion of each variable and interactions between variables. Similarly as in other evolving system modeling approaches [44, 5], the parameters of the fuzzy rules consequents are updated using weighted recursive least squares.

The eMG model uses membership functions of the form:

$$H(x) = \exp\left[-\frac{1}{2}(x-v)\Sigma^{-1}(x-v)^T\right] \tag{12}$$

where x is an $1 \times m$ input vector, v is the $1 \times m$ center vector and Σ is a $m \times m$ symmetric, positive definite matrix. The center vector v is the modal value and represents the typical element of $H(x)$. The matrix Σ denotes the dispersion and represents the spread of $H(x)$ [48]. Both, v and Σ, are parameters of the membership function to be associated with cluster center and cluster spread, respectively.

Most of evolving fuzzy systems perform clustering in the input or input-output data space, and create rules using one-dimensional, single variable fuzzy sets which are projections of the clusters on each input variable space. During fuzzy inference, the fuzzy relation induced by the antecedent of each fuzzy rule is computed using an aggregation operator (e.g. a t-norm) and the input fuzzy sets. This approach is commonly used, but it may cause information loss if input variables interact [32, 1]. For instance, system identification and time series forecasting usually use lagged values of the input and/or output as inputs, and these lagged values tend to be highly related.

To avoid information loss, the algorithm introduced herein uses multivariable, instead of single variable Gaussian membership functions to represent each cluster developed by the recursive clustering algorithm. The parameters of the membership functions are extracted directly from the corresponding clusters. These multivariable membership functions use the information about the dispersion matrix of each cluster (estimated by the clustering procedure) and thus provide information about input variables interactions.

6.3.1 Gaussian Participatory Evolving Clustering

The evolving clustering algorithm described in this paper is based on the concept of participatory learning [58]. Participatory learning assumes that learning and beliefs about the system to be modeled depend on what the model has already learned. In other words, the current knowledge about the system is part of the learning process itself and influences the way in which new observations are used in learning.

An essential characteristic of this learning process is that a new observation impact in causing learning or belief revision depends on its compatibility with the current system belief. Therefore, clustering algorithms based on this learning process tend to be robust to noise and outliers because they are likely to be incompatible with the current system belief and consequently can be discarded or smoothed. The participatory learning clustering algorithm provides an automatic mechanism to decide if a new observation lying far outside current cluster structure denotes either a new cluster to be included into the model, or if it is an outlier who should be discarded or smoothed.

The evolving clustering algorithm assumes that the knowledge about the system to be modeled is the cluster structure, i.e, the number of clusters, the corresponding cluster centers v_i^k for $i = 1, \cdots, c^k$, where c^k is the number of clusters at step k, and the shape of clusters encoded in Σ^k. At each step, the learning process may create a new cluster, modify the parameters of an existing one, or merge two similar clusters.

The cluster structure is updated using a compatibility measure $\rho_i^k \in [0,1]$ and an arousal index, $a_i^k \in [0,1]$. The compatibility measure computes how much an observation is compatible with the current cluster structure, while the arousal index is the output of a arousal mechanism that acts as a critic to remind when the current structure should be revised in front of new information contained in data.

Thresholds are defined for the compatibility measure (T_ρ) and the arousal index (T_a). If at each step the compatibility measure of the current observation is less than the threshold for all clusters, i.e, $\rho_i^k < T_\rho \;\; \forall \;\; i = 1, \cdots, c_k$, and the arousal index of the cluster with the greatest compatibility is greater than the threshold, i.e, $a_i^k > T_a \;\;$ for $i = \arg\max_i \rho_i^k$, then a new cluster is created. Otherwise the cluster center with the highest compatibility is adjusted as follows:

$$v_i^{k+1} = v_i^k + G_i^k (x^k - v_i^k) \tag{13}$$

where G_i^k is defined as:

$$G_i^k = \beta (\rho_i^k)^{1-a_i^k} \tag{14}$$

and $\beta \in [0,1]$ is the basic learning rate.

According to [58], the compatibility ρ_i^k is a function that measures the compatibility between the current belief of the model, represented by each cluster center, and the current observation:

$$\rho_i^k = F(x^k, v_i^k) \tag{15}$$

The function $F(x^k, v_i^k) \in [0,1]$ is such that it should approach *zero* as observations become contradictory with the current belief, i.e, the cluster centers, and approach *one* as the observations become in complete agreement with the current belief. For example, if x^k is equal to a cluster center, then $F(x^k, v_i^k) = 1$.

If $a_i^k = 0$, then $G_i^k = \beta \rho_i^k$ and the PL procedure has no arousal. The learning rate is modulated by the compatibility measure only. Observations with $\rho_i^k = 0$ provides no new information because $v_i^{k+1} = v_i^k$, while an observation with $\rho_i^k = 1$ does bring new information. For example, if $\beta = 1$ and $\rho_i^k = 1$, then $v_i^{k+1} = x^k$.

In the above, notice that the basic learning rate β is modulated by the compatibility ρ. When there are no participatory considerations, β is often made a small value to preclude great swings due to spurious values of inputs which are far from the current cluster structure. Small values of β protect against the influence of bad inputs, but may slow down the learning process. The introduction of the participatory term ρ allows the use of higher values of β. The learning rate of the participatory learning model is dynamic and ρ acts in a way that it lowers the learning rate when large deviations occur. However, when the compatibility is large, ρ is such that it speeds up learning.

The arousal index is the output of an arousal mechanism used to measure the confidence about the current knowledge of the system. For example, while a single low value of the compatibility measure causes aversion to learning, a sequence of low values of the compatibility measure should imply on a revision of the current knowledge about the system.

The arousal mechanism is defined as a monitoring mechanism of the dynamics of the compatibility measure. This mechanism monitors the values of the compatibility level and its output is interpreted as the complement of the confidence about the current knowledge. A low value of a_i^k implies in a high confidence about the system belief, while a high value indicates the necessity to revise the current belief. Analysis of expression (14) shows that, as the arousal index increases, the compatibility measure has a reduced effect, indicating that if a sequence of observations presents low compatibility values, then it is more likely that the current knowledge is incorrect and must be revised. When this happens, the value of the compatibility measure is reduced and the current observation will provide more information about the system when compared with the information provided without the arousal mechanism. As explained later in this section, the extreme case is when the arousal index exceeds a threshold and a new cluster is generated.

Fig. 6 illustrates the participatory learning procedure, including the basic idea of using the current beliefs on the learning process and the arousal index monitoring mechanism.

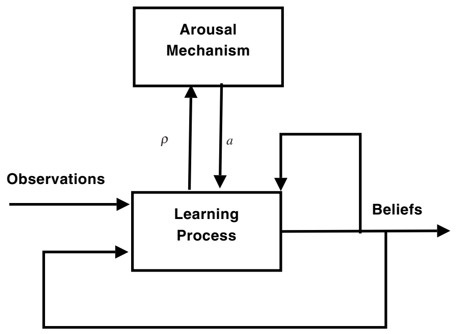


Fig. 6 Participatory learning procedure

The compatibility measure ρ_i^k suggested in [38] uses the squared value of the normalized distance between the new observation and cluster centers (*M-Distance*):

$$M(x^k, v_i^k) = (x^k - v_i^k)(\Sigma_i^k)^{-1}(x^k - v_i^k)^T \tag{16}$$

To compute the *M-Distance*, the dispersion matrix of each cluster Σ_i^k must be estimated at each step. The recursive estimation of the dispersion matrix proceeds as follows:

$$\Sigma_i^{k+1} = (1 - G_i^k)(\Sigma_i^k - G_i^k(x^k - v_i^k)(x^k - v_i^k)^T) \tag{17}$$

The compatibility measure at each step k is given by:

$$\rho_i^k = \exp\left[-\frac{1}{2}M(x^k, v_i^k)\right] \tag{18}$$

To find a threshold value for the compatibility measure, we assume that the values $M(x^k, v_i^k)$ can be modeled by a Chi-Square distribution. Thus, given a significance level α, the threshold can be computed as follows:

$$T_\rho = \exp\left[-\frac{1}{2}\chi^2_{m,\alpha}\right] \tag{19}$$

where $\chi^2_{m,\alpha}$ is the α upper unilateral confidence interval of a Chi-Square distribution with m degrees of freedom, where m is the number of inputs.

The compatibility measure is based on a normalized distance measure (16). The corresponding threshold (19) must be adjusted considering the input space dimension to avoid the *curse of dimensionality.* This is because, as the input space dimension increases, the distance between two adjacent points also increases [22]. If a fixed threshold value is used and it does not depend of the input space dimension, then the number of threshold violations will increase, which may lead to an excessive generation of clusters [44]. Looking at expression (19), one can note that the compatibility measure threshold includes information about the data space dimensionality because $\chi^2_{m,\alpha}$ is a function of the number m of inputs. Therefore no manual adjust is needed and the *curse of dimensionality* is automatically avoided. In other words, the clustering method has an automatic mechanism to adjust the compatibility measure threshold according to input space dimension. As the data dimension increases, the distance between two adjacent points also increases, and the respective compatibility measure decreases. However, the compatibility measure threshold also decreases avoiding excessive threshold violations.

The arousal mechanism adopted by the eMG model uses a sliding window assembled by the last w observations. More specifically, we define the arousal index as the probability of observing less then nv violations of the compatibility threshold on a sequence of w observations. Low values of the arousal index are associated with no or few violations of the compatibility threshold, implying a high confidence about the system knowledge. High values of the arousal index are associated with several threshold violations, meaning that the current cluster structure must be revised.

To compute the arousal index for each observation, a related *occurrence* value o_i^k is found using the following expression

$$o_i^k = \begin{cases} 0, \text{ for } M(x^k, v_i^k) < \chi^2_{m,\alpha} \\ 1, \text{ otherwise} \end{cases} \tag{20}$$

Notice that the occurrence value $o_i^k = 1$ indicates threshold violation.

Occurrence value o_i^k can also be viewed as the output of a statistical test to evaluate if the values of $M(x^k, v_i^k)$ are the expected ones. The null hypothesis of the corresponding test is that $M(x^k, v_i^k)$ can be modeled by a Chi-Square distribution with m degrees of freedom. Under null hypothesis, the probability of observing $o_i^k = 1$ is α because α defines $\chi^2_{m,\alpha}$ and it is the probability of observing a false positive, i.e., $M(x^k, v_i^k) > \chi^2_{m,\alpha}$.

Since the nature of o_i^k is binary and the probability of observing $o_i^k = 1$ is known, the random variable associated with o_i^k can be described by a *Bernoulli* distribution with probability of success α.

Given a sequence assembled by the last w observations, the number of threshold violations nv_i^k is:

$$nv_i^k = \begin{cases} \sum_{j=0}^{w-1} o_i^{k-j}, & k > w \\ 0, & \text{otherwise} \end{cases} \tag{21}$$

Notice that nv_i^k is computed during the first w steps. This means that the algorithm has an initial latency of w steps. However this causes no problem because usually w is much smaller than the number of steps in which learning occurs. For instance, in real-time applications learning can happen continuously.

The discrete probability distribution of observing nv threshold violations on a window of size w is $P(NV_i^k = nv)$, with NV_i^k assuming the values $nv = 0, 1, \cdots, w$. Thus, because NV_i^k is the sum of a sequence of i.i.d. random variables drawn from a *Bernoulli* distribution with the same probability of success α, $P(NV_i^k = nv)$ can be characterized by the *Binomial* distribution:

$$P(NV_i^k = nv) = \begin{cases} \begin{pmatrix} w \\ nv \end{pmatrix} \alpha^{nv}(1-\alpha)^{w-nv}, & nv = 0, \cdots, w \\ 0, & \text{otherwise} \end{cases} \tag{22}$$

The binomial distribution gives the probability of observing nv threshold violations in a sequence of w observations. High probability values enforce the assumption that observations fit the current cluster structure while low probability values suggests that the observations should be described by a new cluster.

The arousal index is defined as the value of the cumulative probability of NV_i^k, i.e.

$$a_i^k = P(NV_i^k < nv) \tag{23}$$

The threshold value of the arousal index T_a is $1-\alpha$, where α is the same as the one that defines the threshold for the compatibility measure. The minimum number of compatibility threshold violations on a window of size w necessary to exceed T_a can be computed numerically looking for the first value of nv for which the discrete cumulative distribution is equal to or greater than $1-\alpha$. More formally

$$nv^* = \arg\min_{nv} \left| \sum_{k=1}^{nv} \begin{pmatrix} w \\ nv \end{pmatrix} \alpha^k (1-\alpha)^{w-k} - (1-\alpha) \right| \tag{24}$$

The clustering algorithm proposed continually revises the current cluster structure and eventually merges similar clusters. The compatibility between the updated or created cluster and all remaining cluster centers is computed at each step. If, for a given pair, the compatibility exceeds the threshold T_ρ, then the two clusters are merged, i.e., if $\rho_i^k(v_j^k, v_i^k) > T_\rho$ or $\rho_j^k(v_i^k, v_j^k) > T_\rho$, then clusters j and i are merged.

The compatibility between two clusters i and j is computed as follows:

$$\rho_i^k(v_j^k, v_i^k) = \exp\left[-\frac{1}{2}M(v_j^k, v_i^k)\right] \tag{25}$$

where $M(v_j^k, v_i^k)$ is the *M-distance* between cluster centers i and j, that is:

$$M(v_j^k, v_i^k) = (v_j^k - v_i^k)(\Sigma_i^k)^{-1}(v_j^k - v_i^k)^T \tag{26}$$

To check if two clusters are similar, is necessary to compute $\rho_i^k(v_j^k, v_i^k)$ and $\rho_j^k(v_i^k, v_j^k)$ because usually $\Sigma_i^k \neq \Sigma_j^k$.

Algorithm 2 summarizes the Gaussian participatory evolving clustering procedure.

Algorithm 2. Gaussian participatory evolving clustering algorithm

```
 1: Compute $\rho_i$ and $a_i$ for all clusters
 2: Select the cluster with the highest compatibility ($j$)
 3: if $\rho_i < T_\rho \ \forall i$ and $a_j > T_a$ then
 4:     Create a new cluster
 5: else
 6:     Update the parameters of cluster $j$
 7: end if
 8: Select the updated/created cluster ($idx$)
 9: for all Clusters ($i$) do
10:     if Compatibility between $c_i$ and $c_{idx}$ is greater than $T_\rho$ then
11:         Merge redundant clusters
12:     end if
13: end for
```

Notice that the clustering algorithm has only three parameters:

- the learning rate β used to compute v_i^k and Σ_i^k;
- the window size w used by the arousal mechanism;
- the confidence level α to compute thresholds T_ρ and T_a.

The learning rate is usually set to a small value, i.e., typically $\alpha \in [10^{-1}, 10^{-5}]$.

The window size w is a problem specific parameter because it defines how many consecutive observations must be considered to compute the arousal index. In other words, considering the current system knowledge, w defines the length of the anomaly pattern needed to classify data either as a new cluster or as a noise or outlier.

The value of the significance level α depends on w. It must be set such that the arousal threshold T_a corresponds to more than one compatibility threshold violation, i.e., $nv > 1$ when $a_i^k > T_a$. Suggested ranges for values of α, given w, are:

$$\alpha \geq \begin{cases} 0.01, \text{ if } w \geq 100 \\ 0.05, \text{ if } 20 \leq w < 100 \\ 0.1, \quad \text{if } 10 \leq w < 20 \end{cases} \tag{27}$$

The clustering process can be started using either a single observation or an initial data set. If initial data set is available, then an off-line clustering algorithm can be used to estimate the initial number of clusters and their respective parameters. The off-line algorithm should be capable to provide both, cluster centers and respective dispersion matrices. If the clustering process starts with a single observation, then an initial dispersion matrix Σ_{init} must be chosen, eventually using *a priori* information about the problem.

Whenever a new cluster is created during the clustering process, the new cluster center is set as the current observation, and the new dispersion matrix is the initial value Σ_{init}.

If two clusters are merged, then the center of the resulting cluster is the average of the corresponding clusters centers and the dispersion matrix is Σ_{init}.

6.3.2 Evolving Multivariable Gaussian Fuzzy Model

The eMG model uses the Gaussian participatory evolving clustering algorithm described in order to define the rule base. The number of eMG rules is the same as the number of clusters found by the clustering algorithm at each step. At each iteration a new cluster can be created, an existing cluster removed, or existing clusters updated. In other words, rules can be created, merged, or adapted at each step of the algorithm. Rules antecedents are of the form:

$$x^k \text{ is } H_i \tag{28}$$

where x^k is a $1 \times m$ input vector and H_i is a fuzzy set with multivariable Gaussian membership function (12) and parameters extracted from the corresponding cluster center and dispersion.

The model is formed by a set of functional fuzzy rules:

$$R_i \ : \ \text{If } x^k \text{ is } H_i \text{ then } y_i^k = \gamma_{io}^k + \sum_{j=1}^{m} \gamma_{ij}^k x_i^k \tag{29}$$

where R_i is the i^{th} fuzzy rule, for $i = 1, \cdots, c^k$, c^k is the number of rules , and γ_{io}^k and γ_{ij}^k are the parameters of the consequent at step k.

The model output is the weighted average of the outputs of the each rule, that is:

$$\hat{y}^k = \sum_{i=1}^{c^k} \Psi_i(x^k) y_i^k \tag{30}$$

with normalized membership functions:

$$\Psi_i(x^k) = \frac{\exp\left[(x^k - v_i^k)\Sigma_i^{-1}(x^k - v_i^k)^T\right]}{\sum_{i=1}^{c^k} \exp\left[(x^k - v_i^k)(\Sigma_i^k)^{-1}(x^k - v_i^k)^T\right]} \tag{31}$$

where v_i^k and Σ_i^k are the center and dispersion matrix of the i^{th} cluster membership function at step k.

The parameters of the consequent are updated using the weighted recursive least squares [42, 12] algorithm, similarly as other TS evolving fuzzy models [5, 44]. Hence, the consequent parameters and matrix Q_i of the update formulas for rule i at each iteration k are:

$$\begin{aligned} \gamma_i^{k+1} &= \gamma_i^k + Q_i^{k+1} x^k \Psi_i(x^k)\left[y_i^k - ((x^k)^T \gamma_i^k)\right] \\ Q_i^{k+1} &= Q_i^k - \frac{\Psi_i(x^k) Q_i^k x^k (x^k)^T Q_i^k}{1 + (x^k)^T Q_i^k x^k} \end{aligned} \tag{32}$$

The eMG algorithm can be initialized either with an existing data set, or with a single observation.

If the eMG starts with an existing data set, then an offline clustering algorithm can be used to estimate the number and parameters of the initial set of rules. Clustering can be done in the input space and a rule created for each cluster. The antecedent parameters of each rule is extracted from the clusters, and the consequent parameters estimated by the weighted least squares algorithm.

If the eMG starts with a single observation, then one rule is created with the antecedent membership function centered at the observation, and the respective dispersion matrix set at the pre-defined initial value. The consequent parameters are initialized as $\gamma^0 = [y^0 \;\; 0 \;\; \cdots \;\; 0]$ and $Q^k = \omega I_{m+1}$, where I_{m+1} is an $m+1$ identity matrix and ω is a large real value, for example, $\omega \in [10^2, 10^4]$ [12].

As new data is input, the eMG algorithm may create, update or merge clusters. Thus, the set of rules, the rule-base, must be updated as well. This is done as follows.

If a new cluster is created, then a corresponding rule is also created with antecedent parameters extracted from the cluster and consequent parameters computed as the weighted average of the parameters of the existing clusters:

$$\gamma_{new}^k = \frac{\sum_{i=1}^{c^k} \gamma_i^k \rho_i^k}{\sum_{i=1}^{c^k} \rho_i^k} \tag{33}$$

The matrix Q is set as $Q_{new}^k = \omega I_{m+1}$.

If an existing cluster is updated, then the antecedent parameters of the corresponding rule are updated accordingly.

Finally, if two clusters i and j are merged, then the consequent parameters of the resulting rule are computed as follows:

$$\gamma^k_{new} = \frac{\gamma^k_i \rho^k_i + \gamma^k_j \rho^k_j}{\rho^k_i + \rho^k_j} \tag{34}$$

The matrix Q is set as $Q^k_{new} = \omega I_{m+1}$.

Algorithm 3 summarizes the eMG learning procedure.

Algorithm 3. eMG learning algorithm

```
 1: Compute model output
 2: Update the cluster structure
 3: if Cluster was created then
 4:     Create a new rule
 5: end if
 6: if Cluster was modified then
 7:     Update antecedent parameters of the respective rule using cluster parameters

 8:     Update consequent parameters of the respective rule using WRLS
 9: end if
10: if Two cluster were merged then
11:     Merge corresponding rules
12: end if
```

6.4 Evolving Fuzzy Linear Regression Trees

This section describes a recently proposed alternative approach for evolving fuzzy modeling using fuzzy linear regression trees (eFT) built from a stream of data in an incremental manner [37]. Linear regression trees [51] are generalizations of regression trees [14] in which zero-order models (mean value of the output variable) in the tree leaves are replaced by linear models. The internal nodes of these trees contains splitting tests to partition the input space in non overlapping regions, and associate a linear model to each one of them. A linear model tree is a binary decision tree with a linear model in each leaf. Fuzzy linear regression trees replaces the binary splitting decisions at each internal node by a pair of membership functions, similarly as in fuzzy decision trees [60, 28], and partitions the input space in overlapping regions. By fuzzyfing decisions at each internal node of the tree, the sharp partition boundaries of the original tree will cease to exist and, for each input sample, all branches of the tree will fire with some degree, resulting in a regression model defined by a weighted sum of linear local models.

Most of the evolving fuzzy modeling methods proposed in literature use information about the spatial organization of the input or input-output samples (for example, by clustering) in order to adapt the model structure. The eFT model partitions the input space selecting split points that improves the goodness of fit of the resulting model. In other words, while most of the eFS uses spatial information for input partition, the approach we propose uses model quality improvement as the criterion for input space partition.

6.4.1 Fuzzy Linear Regression Trees

Linear regression trees are generalizations of regression trees where each leaf is associated with a linear model of the input variables instead of a zero-order model (i.e., the mean value of the output). These trees are recursive structures capable to perform a piecewise linear regression, that is, each region of the input space is associated with a linear model of the form $y_i = \gamma_0 + \gamma_1 x_1 + \cdots + \gamma_m x_m = \sum_{i=0}^{m} \gamma_i x_i$, where $x_0 = 1$ and m is the number of input variables.

Figure 7 gives an example of a linear regression tree.

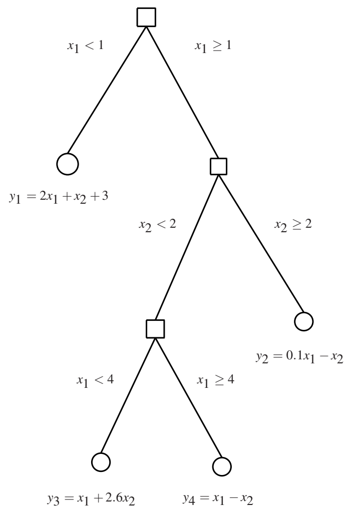


Fig. 7 Example of a linear regression tree

To estimate the output associated with a given input, one must start from the root (top) node and apply splitting decision tests until a leaf is reached. The output is computed using the input values and the linear model present in the leaf. For example, for the tree illustrated in Figure 7, the output for inputs $x_1 = 4$ and $x_2 = 3$ is -2.6.

Each leaf of the tree represents a region of the input space delimited by the splitting points of the internal nodes in the path starting at root and ending in the leaf. Each region is modeled by the linear model associated with the corresponding leaf. Figure 8 illustrates the partition of the input space induced by the linear regression tree of Figure 7, where $x_1 \in [0,6]$ and $x_2 \in [0,6]$.

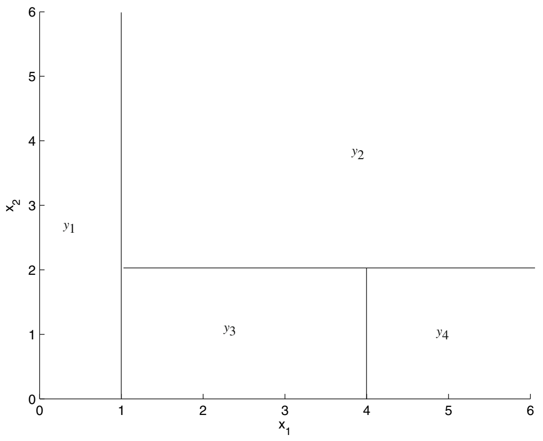


Fig. 8 Input space partition of the linear regression tree of Figure 7

There are several algorithms to grow linear regression trees and classic regression trees using batch [14, 51, 55, 17] or incremental learning approaches [49, 26, 27].

Fuzzy linear regression trees replaces each splitting decision test of the internal nodes by two membership functions describing the concepts of *less than* and *greater than*. Introducing fuzzy sets in each internal node of the tree, the sharp partition boundaries of the original tree will cease to exist and, for each input sample, all branches of the tree will fire to some degree. This results in an overlapping partition of the input space and a regression model based on a weighted sum of local linear models.

The approach introduced in this paper uses sigmoidal membership functions in each internal node of the tree ([35]). Sigmoidal membership functions have two parameters, the center c and the spread σ. Depending of the sign of the parameter σ, the sigmoid is inherently open to the right or to the left. They are natural candidates to represent the concepts of *less than* or *greater than c*. Figure 9 shows an example of membership functions describing the concepts *less than 5* and *greater than 5* assuming $\sigma = -0.5$ (*less than*) and $\sigma = 0.5$ (*greater than*).

$$\mu(x) = \frac{1}{1 + \exp - \frac{1}{\sigma}(x - c)} \tag{35}$$

Each leaf of the tree is associated with a region of the input space and a corresponding linear model. However, the input space partition overlaps, i.e., some regions of the input space are associated with more than one leaf. Figure 10 shows the resulting partition of the input space ($x_1 \in [0,6]$ and $x_2 \in [0,6]$) when the splitting tests of the tree illustrated on Figure 7 are replaced by pairs of sigmoidal membership functions with c being the original split points and $|\sigma| = 0.2$ for all internal nodes.

To compute an output for a given input, one must start at the root node and find the membership values for each pair of membership functions in the internal nodes of all paths from the root to the leaves. Next, the membership values are aggregated using an aggregation operator such as a *t-norm*. This results in a membership value linked to the model of each tree leaf. Algorithm 4 details the recursive algorithm to evaluate the local linear models and to

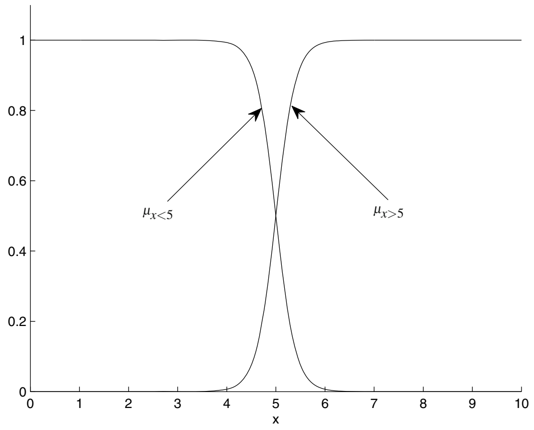


Fig. 9 Membership functions for *less than* and *greater than* 5

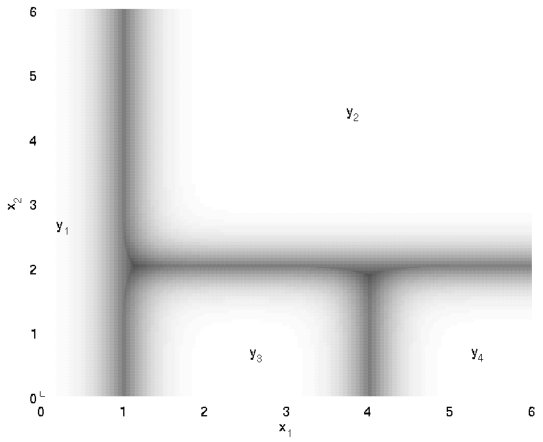


Fig. 10 Partition of the input space by the fuzzy linear regression tree

compute membership values of all leaves of the tree. The algorithm starts with the tree root node and with $w = 1$. These values are used to compute the output as the weighted average of all linear models outputs:

$$\hat{y} = \frac{\sum_{i=1}^{l} y_i w_i}{\sum_{i=1}^{l} w_i} \tag{36}$$

where l is the number of leaves, and w_i is found using:

$$w_i = \mu_1(x)\ t\ \mu_2(x)\ t\ \cdots\ t\ \mu_o(x) = T_{j=1}^{o} \mu_j(x) \tag{37}$$

where t is a *t-norm* [48], o is the number of internal nodes reached from the root to the leaf i, and μ_j is one of the sigmoidal membership functions associated with internal node j.

Algorithm 4. Compute membership values and linear models outputs for all tree leaves

```
 1: if $node_i$ is a not a leaf then
 2:     Aggregate the left membership function value with $w$ using a t-norm
 3:     $i =$ index of left subtree root node
 4:     Evaluate the left subtree
 5:     Aggregate the right membership function value with $w$ using a t-norm
 6:     $i =$ index of right subtree root node
 7:     Evaluate the right subtree
 8: else
 9:     Store the membership value $w_i$
10:     Compute the linear model output $y_i$
11: end if
```

It is interesting to notice that fuzzy functional rules can be easily extracted from the tree structure. Each leaf represents a fuzzy rule whose antecedent is a *t-norm* aggregation of all internal nodes in a path from the root to the leaf, and the consequent is the linear model in the leaf. For instance, from the example of Figure 7 we get the following fuzzy rules:

$$\begin{array}{l} \text{If } x_1 < 1 \text{ then } y = 2x_1 + x_2 + 3 \\ \text{If } x_1 > 1 \text{ and } x_2 < 2 \text{ and } x_1 < 4 \text{ then } y = x_1 + 2.6x_2 \\ \text{If } x_1 > 1 \text{ and } x_2 < 2 \text{ and } x_1 > 4 \text{ then } y = x_1 - x_2 \\ \text{If } x_1 > 1 \text{ and } x_2 > 2 \text{ then } y = 0.1x_1 - x_2 \end{array} \tag{38}$$

where "$< c$" and "$> c$" are fuzzy sets with sigmoidal membership functions representing *less than* and *greater than*.

It is also possible to reduce the number of fuzzy sets of each rule assuming that we can combine two or more fuzzy sets associated with the same input variable into a single fuzzy set. For example, the fuzzy sets $x_1 > 1$ and $x_1 > 4$ can be replaced by $x_1 > 4$, preserving the rule interpretability. Also, combination of the fuzzy sets $x_1 > 1$ and $x_1 < 4$ can be viewed as a fuzzy set describing the interval $1 < x_1 < 4$ with membership function $\mu_{1<x_1<4} = \mu_{x_1<1}\ t\ \mu_{x1_{<}4}$, For instance, if we choose the product operator as the *t-norm*, then the analytical expression of this membership function, given c_1 and c_2 (with $c_1 < c_2$) and σ is:

$$\mu_{c_1<x<c_2} = \frac{1}{1+\exp -\frac{1}{\sigma}(x-c_1)+\exp \frac{1}{\sigma}(x-c_2)+\exp \frac{1}{\sigma}(c_1-c_2)} \tag{39}$$

The rule reduction technique results in the following rule base:

$$\begin{array}{l} \text{If } x_1 < 1 \ \text{ then } y = 2x_1 + x_2 + 3 \\ \text{If } x_1 > 4 \text{ and } x_2 < 2 \text{ then } y = x_1 - x_2 \\ \text{If } 1 < x_1 < 4 \text{ and } x_2 < 2 \text{ then } y = x_1 + 2.6x_2 \\ \text{If } x_1 > 1 \text{ and } x_2 > 2 \text{ then } y = 0.1x_1 - x_2 \end{array} \tag{40}$$

Clearly, the rule base (6) is much simpler and interpretable than the original (4). Figure 11 shows the membership functions extracted from the linear regression tree of Figure 7.

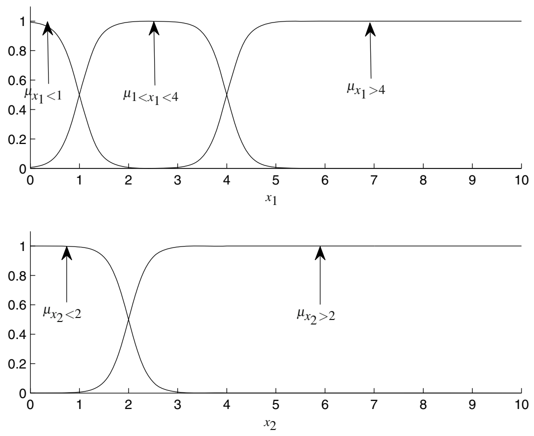


Fig. 11 Membership functions extracted from the fuzzy tree

6.4.2 Incremental Learning Algorithm

This section details the incremental learning algorithm for the fuzzy linear regression tree described in the previous section. The algorithm starts with a single leaf tree and its linear model, and evolves the tree replacing leaves by subtrees using a statistical model selection test and input data. The algorithm does not store any past value and all decisions are taken based on recursively estimated statistics.

To grow the tree, we assume that each tree leaf has k candidate splits for each of the m input variables, totaling $k \times m$ possible splits. Any of these can be used to replace an existing leaf. Every candidate split has a subtree composed by an internal node, with two sigmoidal membership functions (*less than* and *greater than*) centered on the split point, followed by two leaves, the left and the right one, containing linear models. Figure 12 illustrates a generic candidate split.

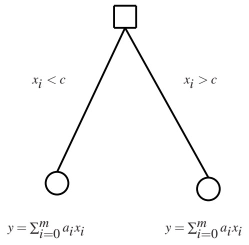

$x_i < c$
$x_i > c$
$y = \Sigma_{i=0}^{m} a_i x_i$
$y = \Sigma_{i=0}^{m} a_i x_i$

Fig. 12 Generic candidate split

As discussed latter, each leaf represents a region of the input space. To define candidate split points, the range of the leaf for each input variable is computed and split points chosen to uniformly divide the range into $k+1$ intervals. Although each leaf defines a fuzzy region of the input space, the range is sharp and is found using the centers of the internal nodes membership functions reached from the root to the leaf. For example, for the tree shown in the previous section, the leaf associated with the linear model $y = 0.1x_1 - x_2$ assume the following: $x_1 \in [1, max(x_1)]$ and $x_2 \in [2, max(x_2)]$, where $max(x_i)$ is the maximum value observed for the variable i. The maximum and minimum values for each input variable are updated whenever new observations exceeds the observed limits.

For each new observation, the corresponding output is estimated using (36) and the linear models of all leaves updated using the weighted recursive least squares algorithm [59, 42]. The linear parameters and matrix Q_i of the update formulas for leaf i at each iteration k are:

$$\begin{aligned} \gamma_i^{k+1} &= \gamma_i^k + Q_i^{k+1} x^k \Psi_i(x^k) \left[y_i^k - ((x^k)^T \gamma_i^k) \right] \\ Q_i^{k+1} &= Q_i^k - \frac{\Psi_i(x^k) Q_i^k x^k (x^k)^T Q_i^k}{1 + (x^k)^T Q_i^k x^k} \end{aligned} \tag{41}$$

where $\Psi_i(x^k)$ for $i = 1, \cdots,$ are the normalized firing degrees associated with each local linear model:

$$\Psi_i(x^k) = \frac{w_i}{\sum_{j=1}^{l^k} w_j} \tag{42}$$

where l^k is the number of tree leaves at iteration k.

Next, the membership function spreads (σ) of all internal nodes in the path from the root to the leaf associated with the highest firing degree are revised.

For each membership function in the path, the spread of the sigmoidal function is adjusted using the gradient descent algorithm. For example, consider a fuzzy linear regression tree with a topology shown in Figure 7. Assume that, for a given input, the leaf with the highest firing degree is y_2. In this case, the membership functions describing $x_1 \geq 1$ and $x_2 \geq 2$ will have their spreads updated.

Membership functions spreads update aims, at each iteration, to minimize an error measure involving the model output and the corresponding desired output:

$$e^k = \frac{1}{2}\left(\hat{y}^k - y^k\right)^2 \tag{43}$$

The recursive equation to update the spreads of the membership functions present on the internal nodes is:

$$\sigma_i = \sigma_i - \beta \frac{\partial e^k}{\partial \hat{y}^k} \frac{\partial \hat{y}^k}{\partial \mu_i} \frac{\partial \mu_i}{\partial \sigma_i} \tag{44}$$

where β is the basic learning rate. The partial derivatives are:

$$\frac{\partial e^k}{\partial \hat{y}^k} = \hat{y}^k - y^k \tag{45}$$

$$\frac{\partial \mu_i}{\partial \sigma_i} = \frac{\left(\exp - \frac{1}{\sigma_i}(x^k - c_i)\right)(x^k - c_i)}{1 + \exp - \frac{1}{\sigma_i}(x^k - c_i)^2 \sigma_i^2} \tag{46}$$

The partial derivative $\partial \hat{y}^k / \partial \mu_i$ can be obtained from (36) and (37). To compute this derivative, one must first find all paths from the root node to all leaves that passes through the membership function μ_i and then compute the partial derivative as:

$$\frac{\partial \hat{y}^k}{\partial \mu_i} = \frac{\sum_{r=1}^{np^k} \frac{w_r}{\mu_i} y_r - \hat{y}^k \sum_{r=1}^{np^k} \frac{w_r}{\mu_i}}{\sum_{j=1}^{l^k} w_j} \tag{47}$$

where w_r is the *t-norm* aggregation of the membership functions on a path that passes through μ_i and np^k is the number of paths satisfying this condition at iteration k. Note that (47) is only valid for the product *t-norm*, thus this is the *t-norm* adopted in all experiments presented in this paper.

The linear models of all candidate splits associated with the selected leaf are also updated using (41). For each candidate split, the weights (42) for all tree leaves are revised, including the ones present in the candidate split. Next, the output of the resulting tree (with the selected leaf replaced by the candidate split) is computed. Finally the linear models in the leaves of the selected candidate split are updated using (41). The algorithm to compute candidate splits parameters and to update the tree is summarized in Algorithm 5.

Algorithm 5. Algorithm to update the parameters of the tree

```
1: Compute the output and membership value of all leaves
2: Update the linear models of all leaves using WRLS
3: Select the leaf associated with the highest membership value
4: Update the spread of the internal nodes present on the path from the root node
   to the selected leaf
5: for all inputs ($m$) do
6:    for all candidate splits ($k$) do
7:       Replace the selected leaf with the candidate split subtree
8:       Compute the output for the resulting tree
9:       Update the candidate split linear models using WRLS
10:   end for
11: end for
```

Once the leaf is updated, tests are performed to evaluate if the subtree of each candidate split can replace the corresponding leaf. The test is a goodness of fit test that considers the accuracy and the number of parameters of the tree, with and without the subtree. The test compares the quality of two models, a simpler one (the original tree) and the more complex one (the original tree with the subtree added), assuming that the simpler model can be nested in the complex model, and that the complex is also more accurate. The test tries to answer following the question: the gain in the accuracy (measured by using points already observed) worths the *cost* of adding more free parameters to the tree (which may lead to over fitting)?

The original test for nested models [2] assumes that the parameters of the two models are estimated using the same data set and computes the following statistics:

$$F = \frac{(RSS_1 - RSS_2) \times (n - p_2)}{RSS_2 \times (p_2 - p1)} \tag{48}$$

where RSS_1 and RSS_2 is the sum of residual squares of the simpler and the complex models respectively, p_1 and p_2 is the number of free parameters of each model and n is the number of data samples used to estimate the parameters of the models. The number of parameters of a tree is the number of parameters of the linear models in the leaves:

$$p = (m+1) \times \text{number of leaves} \tag{49}$$

where m is the dimension of the input.

Assuming that the distribution of the residuals is normal, F (48) follows a Fisher's F distribution with $(p_2 - p_1, n - p_2)$ degrees of freedom.

However, this test cannot be used to evolve the tree because the number of samples to estimate the tree parameters may not be equal to the number of samples used to estimate each candidate split parameters because a new candidate split is created whenever a new leaf is inserted on the tree. Thus, the number of samples needed to estimate the tree parameters would always be greater or equal to the number of samples for the candidate splits. A modification of the test mechanism to handle models trained with different samples sizes, suggested in [49], compute the statistics as follows:

$$F_{inc} = \frac{(RSS_1 - RSS_2) \times (n_2 - p_2)}{RSS_2 \times (n_1 - n_2 + p_1)} \tag{50}$$

where n_1 and n_2 are the number of samples used to estimate the tree and candidate splits parameters, respectively.

F_{inc} is distributed according to Fisher's F distribution with $(n_2 - n1 + p_2 - p_1, n_2 - p2)$ degrees of freedom. Hence, the use of this statistics requires computation of the p-values (probability in the tail of the distribution) for all candidate splits of the last modified leaf. The candidate split associated with the smallest p-value is selected. The subtree of the selected candidate split replaces the corresponding leaf if its p-value is smaller than a confidence level α. However, it is necessary to introduce a multiple-comparison statistical correction because the same hypothesis is tested $k \times m$ times using the same data set [49]. Thus, the Bonferroni correction [46] must be applied by dividing the desired significance level by the number of tests. Finally, the selected subtree is added to the model if

$$\text{p-value} < \frac{\alpha}{k \times m} \tag{51}$$

To use the model selection test just described, the sum of squared residuals and the number of samples of the tree and of all candidate splits must be updated whenever the respective models are. Initially the estimate of the tree output is computed using (36). Next, the leaf associated with the highest membership value is selected and, for all candidate splits, the output is computed replacing the related leaf with the candidate split subtree. The simplest way to perform this operation is to replace the leaf with the subtree and recompute the output using (36). This operation can be optimized using local models outputs already computed, the membership values for all leaves of the original tree, and replacing the model output and membership value of the selected leaf by the models outputs, and membership values of the candidate split.

Algorithm 6 details the algorithm to evolve fuzzy regression trees.

Algorithm 6. Learning algorithm to evolve fuzzy linear regression models

```
1: Compute the output and membership value of all leaves
2: Update the linear models
3: Select the leaf with the highest membership value
4: for all inputs ($m$) do
5:     for all candidate splits ($k$) do
6:         Estimate the output replacing the selected leaf with the candidate split
7:         Compute the p-value of the model selection test for the candidate split
8:     end for
9: end for
10: Select the candidate split associated with the minimum p-value
11: if p-value $< \frac{\alpha}{k \times m}$ then
12:     Replace the selected leaf by the candidate split
13: end if
```

The learning algorithm has 4 parameters:

- significance level adopted during model selection test, α;
- number of candidate splits for each variable, k.
- initial spread of the sigmoid membership functions, σ_{init}.
- basic learning rate to update the spread of the membership functions, β.

The significance level is usually set to values such as 0.05 or 0.01.

The number of candidate splits must be chosen based on the trade-off between modeling accuracy and computational cost. Low values of k may decrease the performance of the tree because fewer candidate splits will be created for each leaf. High values increases performance, but also increases the number of linear models to be updated.

The initial spread of the membership functions is chosen from a priori information about data scale. The appropriate adjustment of this parameter speeds the convergence to the optimal spread value.

Finally, the basic learning rate is usually set to a small value, i.e., $\beta \in [10^{-5} 10^{-1}]$.

The algorithm also needs initial information about the range of the variables to create initial candidate splits. This range can be found using an initial data set. The algorithm updates the initial range whenever necessary.

6.5 Experiments

The performance of the models described in the previous sections were evaluated using time series forecasting problems. The results obtained were compared with alternative evolving modeling approaches. The models used for comparison were: eTS [5], xTS [10] and DENFIS [31].

The Matlab DENFIS implementation used is available from the Knowledge Engineering and Discovery Research Institute (KEDRI) (http://www.aut.ac.nz /research/research-institutes/kedri/books). The Java eTS and xTS implementation used was provided by the respective authors.

In all experiments, modeling performance was evaluated using the root mean squared error (*RMSE*) and/or the non-dimensional error (*NDEI*). The *NDEI* is non dimensional error index defined as the ratio of the root mean squared error by the standard deviation of the target data. The error measures are computed as follows:

$$RMSE = \left(\frac{1}{n} \sum_{k=1}^{n} (y^k - \hat{y}^k) \right)^{\frac{1}{2}} \tag{52}$$

$$NDEI = \frac{RMSE}{std(y^k)} \tag{53}$$

where n is the size of the test data set, y^k is the target output, and $std()$ is the standard deviation.

These error indices are good indices to measure the model accuracy. However, they do not reveal whether the results from one model is statistically superior to any other model. Therefore, a statistical test was performed to compare the models in terms of accuracy.

The MGN test [16] is a parametric test used to compare the accuracy of two forecasting models. The statistic for this test is found as:

$$MGN = \frac{\hat{\rho}_{sd}}{\sqrt{\frac{1-\hat{\rho}_{sd}^2}{n-1}}} \tag{54}$$

where $\hat{\rho}_{sd}$ is the estimated correlation coefficient between $s = r_1 + r_2$, and $d = r_1 - r_2$, with r_1 and r_2 the residuals of the two models adjusted. In this case, the statistic is distributed as Student's T distribution with $n-1$ degrees of freedom. For this test, if the forecasts are equally accurate, then the correlation between s and d will be zero.

6.5.1 Short Term Electricity Load Forecasting

In this section the eMG and eFT models were tested using the short term electricity load forecasting problem. Forecasts of load demand are very important in the operation of electric energy systems because several decision making process, such as system operation planning, security analysis and market decisions are strongly influenced by the future values of the load. In this context, a significant error in the load forecast may result in economic losses, security constraints violations, and system operation drawbacks. Accurate and reliable load forecasting models are essential for a suitable system operation.

The problem of load forecasting can be classified in long, medium and short-term, depending on the situation. Long-term forecasting is important for capacity expansion of the power system. Medium term is important to organize the fuel supply, maintaining operations and interchange scheduling. Short-term forecasting is generally used for daily programming and operation of the power system, energy transfer and demand management [52].

Particularly, in the context of short-term hydrothermal scheduling, load forecasting is important to elaborate the next day operation scheduling because errors in load forecasting can cause serious consequences, affecting the efficiency and the safety of the system (cost increasing, insufficient electrical energy supply to the existing demand).

The goal of short-term load forecasting is to accurately predict the 24 hourly loads of the next operation day, one-step-ahead. The effectiveness of the proposed approach is illustrated using load data of a major electrical utility located at the Southeast region of Brazil. In evaluating the forecast model, a database consisting of hourly loads from August 1st, 2000 to August 31th, 2000 is used to develop he model and to produce load forecasts. The measures are expressed in kilowatts per hour (kW/hour).

The models are one-step ahead forecasters whose purpose is to predict the current load value using lagged load values in the series. The sample partial autocorrelation function [13] for the first 36 observations of the series suggests the use of the last 2 previous values of load values as inputs of the models, that is, the forecast model is of the form:

$$\hat{y}^k = f(y^{k-1}, y^{k-2}) \tag{55}$$

The experiment has been conducted as follows. The hourly load for first 28 days were input to the learning algorithms and the resulting models performance evaluated using data of the last 3 days, keeping the models structure and parameters fixed at the values found after evolving during the period of 28 days. The parameters of the eMG model were chosen as $\alpha = 0.05$, $w = 20$, $\Sigma_{init} = 10^{-2}I_2$ e $\beta = 0.01$. The parameters of the eFT model were chosen as $\sigma_{init} = 0.01$, $\alpha = 0.05$, $k = 20$ and $\beta = 0.01$. The data was normalized between 0 and 1 to preserve privacy.

In this experiment, the parameters of eMG, eFT and the alternative methods were selected to generate models with similar structures, that is, similar number of fuzzy rules. The xTS model has only 1 parameter, that does not influences the resulting number of rules. Therefore the parameters of all models were adjusted in order to generate models with similar number of rules as xTS.

Figures 14 and 15 show the forecasting results for the eMG and eFT models, respectively.

Figure 16 sketches the final 5 clusters found by the eMG model after learning. Looking at this figure one can note that the resulting clusters have distinct orientations and most of them are not parallel to the input axes.

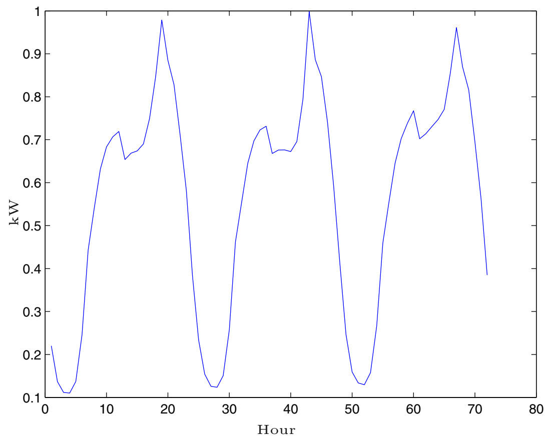


Fig. 13 Normalized load data for the first 3 days of August 2000

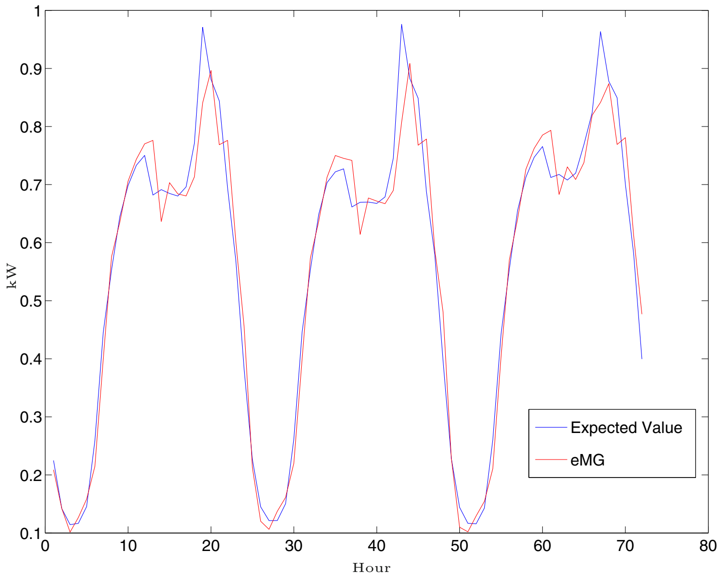


Fig. 14 Electricity load forecast for the eMG model

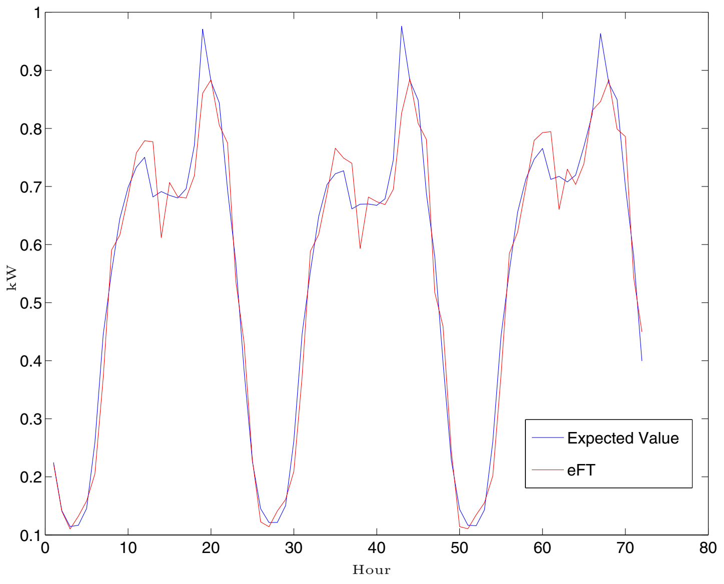


Fig. 15 Electricity load forecast for the eFT model

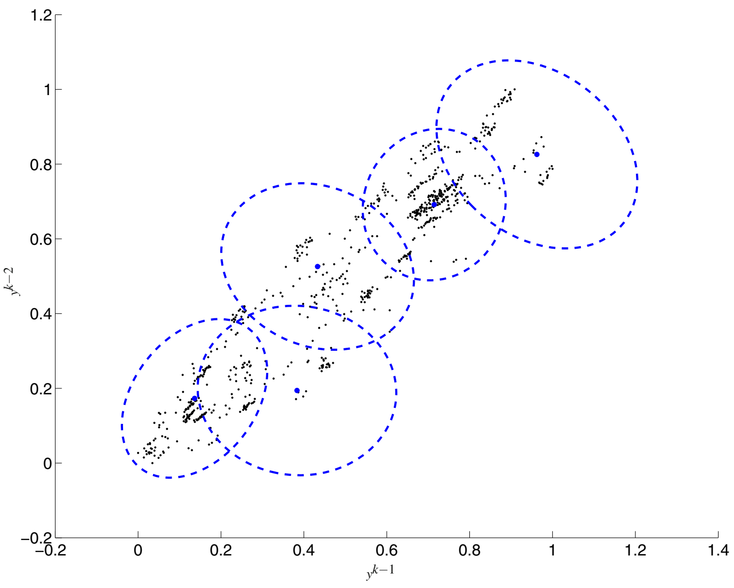


Fig. 16 eMG resulting cluster structure for load forecasting

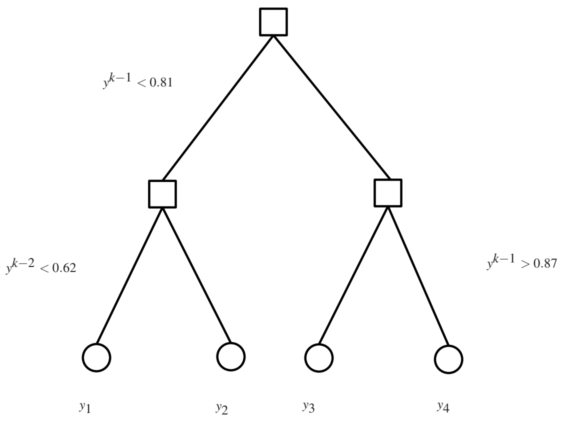


Fig. 17 Fuzzy linear regression tree model for load forecast

The resulting eMG model is composed by the following rules:

$$\begin{array}{l} \text{If } x^k \text{ is } A_{[0.1363\;\;0.1731]} \text{ then } y_1 = 0.0361 + 2.0103y^{k-1} - 0.9894y^{k-2} \\ \text{If } x^k \text{ is } A_{[0.9623\;\;0.8260]} \text{ then } y_2 = 0.4985 + 0.6757y^{k-1} - 0.3350y^{k-2} \\ \text{If } x^k \text{ is } A_{[0.4331\;\;0.5260]} \text{ then } y_3 = 0.0756 + 1.7899y^{k-1} - 0.9618y^{k-2} \\ \text{If } x^k \text{ is } A_{[0.7146\;\;0.6916]} \text{ then } y_4 = -0.1018 + 1.8821y^{k-1} - 0.7274y^{k-2} \\ \text{If } x^k \text{ is } A_{[0.3834\;\;0.1942]} \text{ then } y_5 = 0.1334 + 1.5142y^{k-1} - 0.9020y^{k-2} \end{array} \tag{56}$$

where x^k is the input vector, i.e., $x^k = [y^{k-1}\;\; y^{k-2}]^T$; and $A_{[x\;\;y]}$ is a multivariable Gaussian membership function centered at $[x\;\;y]^T$.

Figure 17 shows the resulting fuzzy linear regression tree (eFT model). The following rules can be extracted from the tree topology:

$$\begin{array}{l} \text{If } y^{k-1} < 0.8095 \text{ and } y^{k-2} < 0.6190 \text{ then } y_1 = 0.0574 + 1.6736y^{k-1} - 0.8215y^{k-2} \\ \text{If } y^{k-1} < 0.8095 \text{ and } y^{k-2} > 0.6190 \text{ then } y_2 = -0.0329 + 2.1482y^{k-1} - 1.0931y^{k-2} \\ \text{If } 0.8095 < y^{k-1} < 0.8730 \text{ then } y_3 = 0.0608 + 1.7244y^{k-1} - 0.8448y^{k-2} \\ \text{If } y^{k-1} > 0.8730 \text{ then } y_4 = -0.1242 + 0.9342y^{k-1} + 0.1310y^{k-2} \end{array} \tag{57}$$

Table 1 shows how the eMG and eFT perform against evolving modeling methods using *RMSE* and *NDEI* error measures. The parameters of the eTS model were set to $r = 0.5$ and $\Omega = 750$. The xTS has a $\Omega = 750$. The DENFIS has a distance threshold equals to 0.18.

Table 1 suggests that the eFT performs best among all the models.

Table 2 includes the pairwise comparisons between the forecasts of the models using the MGN test ([54]). As this table suggests, the eFT and eMG models presents statistically significant evidence of superior performance over all alternative models, for a significance level of 0.05. The last row of this table also suggests that eMG has a similar performance as eFT.

Table 1 Performance of electricity load forecast methods

Model	# of rules	RMSE	NDEI
DENFIS	5	0.0665	0.2568
xTS	4	0.0634	0.2447
eTS	5	0.0584	0.2254
eMG	5	0.0499	0.1929
eFT	4	0.0496	0.1916

Table 2 MGN Test Evaluation for Electricity Load Forecast

Models	*MGN*	*p-value*
eMG vs DENFIS	4.0965	0.0001
eMG vs eTS	3.1094	0.0013
eMG vs xTS	4.2864	0.0000
eFT vs DENFIS	3.8964	0.0001
eFT vs eTS	3.3938	0.0006
eFT vs xTS	2.1605	0.0171
eFT vs eMG	0.2705	0.3938

6.5.2 Tree Rings

In this section a high dimensional data set is used in order to evaluate the robustness of the eMG and eFT models for high dimensional input vectors. The tree rings time series contains yearly measures of tree rings width in dimensionless units. The series used was measured in Argentina for the 441-1974 period and corresponds to the *arge030* data set of the Time Series Data Library [25].

The models are one-step ahead forecasters whose purpose is to predict the width for the next year (y^{k+1}) using actual and lagged load values in the series. Previous work [50] suggests the use of the last 10 previous values ($y^k, \cdots, y^{k-9}$, excluding y^{k-4} and y^{k-6}) as inputs of the model , totalling 8 inputs.

The experiment has been conducted as follows. The first 1013 data points were input to the learning algorithms and the evolved models performance evaluated using the last 511 data points, keeping the models structure and parameters fixed. The parameters of the eMG were chosen as $\alpha = 0.05$, $w = 25$, $\Sigma_{init} = 5 \times 10^{-1} I_3$ and $\beta = 0.05$. The parameters of the eFT model were chosen as $\alpha = 0.01$, $\sigma_{init} = 0.04$, $\gamma = 25$ and $\beta = 0.01$.

Both models are composed by only 2 fuzzy rules. Figure 20 illustrates the resulting tree for the eFT model.

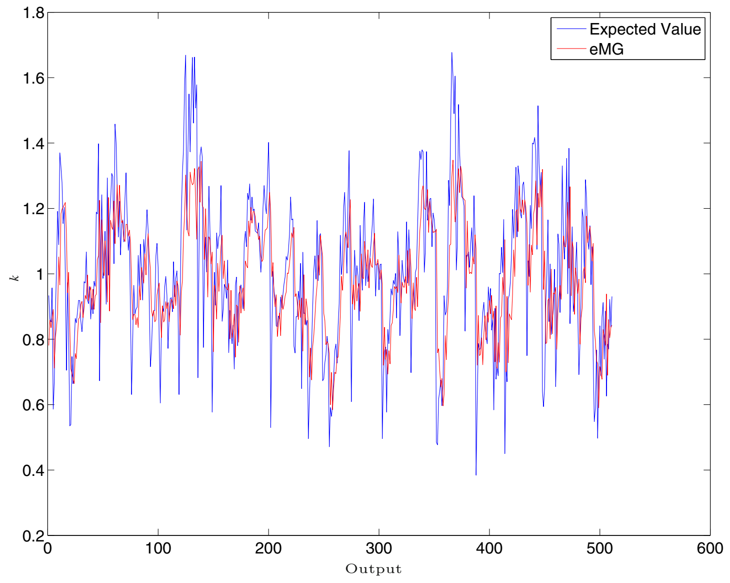


Fig. 18 Tree rings forecast for the eMG model

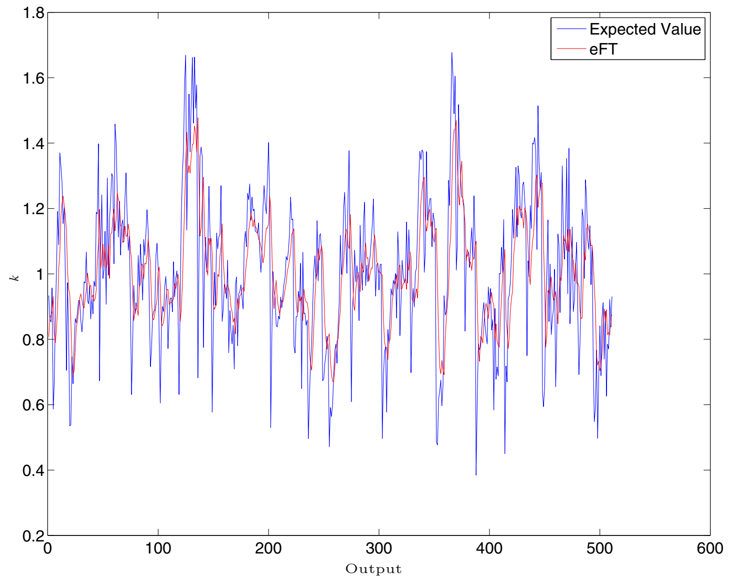


Fig. 19 Tree rings forecast for the eFT model

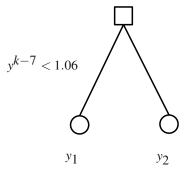


Fig. 20 Fuzzy linear regression tree model for tree rings forecast

Table 3 summarizes the performance of the eMG and eFT models against evolving modeling methods using the *NDEI*. For this experiment, only the *NDEI* error measure was used because the data set was not normalized and some methods (eTS and xTS) used for comparison requires data normalization. The parameters of the eTS model were set to $r = 2$ and $\Omega = 750$. The xTS has a $\Omega = 750$. The DENFIS has a distance threshold equals to 0.23.

Table 3 Performance of the tree rings forecast methods

Model	# of rules	NDEI
DENFIS	3	0.8415
xTS	16	0.8093
eMG	2	0.7767
eTS	2	0.7731
eFT	2	0.7717

Table 3 suggests that the eMG approach performs best among all the models. Table 4 shows the pairwise comparisons between the forecasts of the models using the MGN test ([54]). From this table one can note that the eFT and eMG models show statistically significant evidence of superior performance over DENFIS and xTS and similar performance with eTS, for a significance level of 0.05.

Table 4 MGN Test Evaluation for the tree rings forecast

Models	*MGN*	*p-value*
eMG vs DENFIS	1.7952	0.0366
eMG vs xTS	2.4959	0.0064
eMG vs eTS	0.2341	0.4075
eFT vs DENFIS	2.1625	0.0155
eFT vs xTS	2.0730	0.0193
eFT vs eTS	0.0846	0.4663
eFT vs eMG	0.0331	0.4868

6.6 Conclusion

Nowadays, it is noticed a high interest in the development of highly adaptive and flexible data based models, based on online learning methods that evolve or gradually changes the model parameters and structure to guarantee life-long learning. Such models have been proposed to address problems like system identification and pattern classification, on nonlinear and non stationary environments.

This chapter presented a review and state of the art of these adaptive modeling approaches. The chapter presented the main concepts, some learning approaches and recently developed evolving intelligent models based on participatory learning and fuzzy trees.

Two evolving models were detailed in this chapter. The first model is a fuzzy rule-based model that uses a recursive clustering procedure based on participatory learning to evolve a dynamic structure. This model differs from the ones proposed in literature because: it uses multivariable membership functions for the fuzzy sets of the rules antecedents to prevent information loss about input variables interactions; it is based on a recursive clustering procedure robust to noisy data and outliers because it derives from the concept of participatory learning; its rule creation mechanism is governed by an automatic mechanism to adjust threshold value considering input space dimension and, therefore, does not suffer from the *curse of dimensionality*.

The second model detailed in this paper, uses an alternative topology, defined as a fuzzy linear regression tree and also an alternative criterion for structure evolution. While most evolving methods proposed in literature uses information about the input space spatial organization in order to evolve the model structure, the eFT model updates the tree structure using recursive statistical tests, growing the tree by replacing leaves by subtrees that improves the goodness of fit of the resulting model.

These two models were evaluated using time series forecasting problems. The experiments performed and their results suggest that these methods area a promising alternative to build adaptive models.

Future work shall address the investigation of new topologies for develop evolving models, in particular, the development of evolving fuzzy neural networks. Fuzzy neural networks are neural networks composed by logic fuzzy neurons [48]. The main advantage of these networks is the transparency. It is possible to extract knowledge (linguistic fuzzy rules) from the network topology. It is also possible to define the network topology based on prior information, also described as a set of fuzzy rules.

Recently a fuzzy neural network based on an uninorm fuzzy neurons has been proposed [36]. This network is an universal function approximator, for a given selection of parameters and operators [39]. In [36] an offline learning algorithm is proposed for this network. Future work shall address the development of an evolving learning algorithm capable of update the neurons parameters and the network topology based on a data stream.

References

1. Abonyi, J., Babuska, R., Szeifert, F.: Modified gath-geva fuzzy clustering for identification of takagi-sugeno fuzzy models. IEEE Transactions on Systems Man and Cybernetics Part B (Cybernetics) 32(5), 612–621 (2002), doi:10.1109/TSMCB.2002.1033180
2. Allen, M.P.: Understanding Regression Analysis, 1st edn. Springer (1997)
3. Angelov, P.: Evolving takagi-sugeno fuzzy systems from streaming data, ets+. In: Angelov, P., Filev, D., Kasabov, N. (eds.) Evolving Intelligent Systems: Methodology and Applications. Wiley-Interscience/IEEE Press (2010)
4. Angelov, P., Buswell, R.: Evolving rule-based models: A tool for intelligent adaptation. In: Joint 9th IFSA World Congress and 20th NAFIPS International Conference, vol. 2, pp. 1062–1067 (2001)
5. Angelov, P., Filev, D.: An approach to Online identification of Takagi-Suigeno fuzzy models. IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics 34(1), 484–498 (2004)
6. Angelov, P., Filev, D.: Simpl_ets: a simplified method for learning evolving takagi-sugeno fuzzy models. In: The 14th IEEE International Conference on Fuzzy Systems, FUZZ 2005, pp. 1068–1073 (2005)
7. Angelov, P., Kordon, A.: Adaptive inferential sensors based on evolving fuzzy models. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 40(2), 529–539 (2010), doi:10.1109/TSMCB.2009.2028315
8. Angelov, P., Lughofer, E., Zhou, X.: Evolving fuzzy classifiers using different model architectures. Fuzzy Sets and Systems 159(23), 3160–3182 (2008), doi:10.1016/j.fss.2008.06.019
9. Angelov, P., R., Ramezani, X.Z.: Autonomous novelty detection and object tracking in video streams using evolving clustering and takagi-sugeno type neuro-fuzzy system. In: IEEE International Joint Conference on Neural Networks, IJCNN 2008 (IEEE World Congress on Computational Intelligence), pp. 1456–1463 (2008)
10. Angelov, P., Zhou, X.: Evolving fuzzy systems from data streams in real-time. In: 2006 International Symposium on Evolving Fuzzy Systems, pp. 29–35 (2006)
11. Angelov, P.P.: Evolving rule-based models: a tool for design of flexible adaptive systems. Springer, London (2002)
12. Astrom, K., Wittenmark, B.: Adaptive Systems, 1st edn. Addison-Wesley, USA (1988)
13. Box, G.E.P., Jenkins, G.: Time Series Analysis, Forecasting and Control. Holden-Day, Incorporated (1990)
14. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and regression trees. Wadsworth Mathematics Series, San Diego, California (1984)
15. Chiu, S.L.: Fuzzy model identification based on cluster estimation. Journal of Intelligent and Fuzzy Systems 2(3) (1994)
16. Diebold, F.X., Mariano, R.S.: Comparing predictive accuracy. Journal of Business and Economics Statistics 13, 253–263 (1995)
17. Dobra, A., Gehrke, J.: Secret: a scalable linear regression tree algorithm. In: KDD 2002: Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, pp. 481–487. ACM, New York (2002), http://doi.acm.org/10.1145/775047.775117
18. Er, M.J., Wu, S.: A fast learning algorithm for parsimonious fuzzy neural systems. Fuzzy Sets and Systems 126(3), 337 (2002)

19. Filev, D.P., Tseng, F.: Novelty detection based machine health prognostics. In: 2006 International Symposium on Evolving Fuzzy Systems, pp. 193–199 (2006)
20. Fritzke, B.: Growing cell structures: a self-organizing network for unsupervised and supervised learning. Neural Networks 7, 1441–1460 (1994), doi:10.1016/0893-6080(94)90091-4
21. Gray, R.: Vector quantization. IEEE ASSP Magazine 1(2), 4–29 (1984)
22. Hastie, T., Tibshirani, R., Friedman, J.: The Elements of Statistical Learning: Data Mining, Inference and Prediction. Springer, Berlin (2001)
23. Hirota, K., Pedrycz, W.: D-fuzzy clustering. Pattern Recogn. Lett. 16, 193–200 (1995), doi:10.1016/0167-8655(94)00090-P
24. Hisada, M., Ozawa, S., Zhang, K., Kasabov, N.: Incremental linear discriminant analysis for evolving feature spaces in multitask pattern recognition problems. Evolving Systems 1, 17–27 (2010)
25. Hyndman, R.J.: Time series data library (2010), `http://robjhyndman.com/TSDL`
26. Ikonomovska, E., Gama, J.: Learning model trees from data streams. In: Boulicaut, J.-F., Berthold, M.R., Horváth, T. (eds.) DS 2008. LNCS (LNAI), vol. 5255, pp. 52–63. Springer, Heidelberg (2008), `http://dx.doi.org/10.1007/978-3-540-88411-8_8`
27. Ikonomovska, E., Gama, J., Sebastião, R., Gjorgjevik, D.: Regression Trees from Data Streams with Drift Detection. In: Gama, J., Costa, V.S., Jorge, A.M., Brazdil, P.B. (eds.) DS 2009. LNCS, vol. 5808, pp. 121–135. Springer, Heidelberg (2009)
28. Janikow, C.: Fuzzy decision trees: Issues and methods. IEEE Transactions on Systems Man and Cybernetics Part B-Cybernetics 28(1), 1–14 (1998)
29. Kasabov, N.: Evolving Connectionist Systems: The Knowledge Engineering Approach. Springer-Verlag New York, Inc., Secaucus (2007)
30. Kasabov, N., Leee, S.M.O.: Evolving fuzzy neural networks for supervised/unsupervised on-line knowledge-based learning. IEEE Transactions on Systems, Man and Cybernetics 31, 902–918 (2001)
31. Kasabov, N., Song, Q.: DENFIS: Dynamic evolving neural-fuzzy inference system and its application for time-series prediction. IEEE Transactions on Fuzzy Systems 10(2), 144–154 (2002)
32. Kim, E., Park, M., Kim, S., Park, M.: A Transformed Input-Domain Approach to Fuzzy Modeling. IEEE Transactions on Fuzzy Systems 6(4), 596–604 (1998)
33. Kwok, T.Y., Yeung, D.Y.: Constructive algorithms for structure learning in feedforward neural networks for regression problems. IEEE Transactions on Neural Networks 8(3), 630–645 (1997)
34. Leite, D., Costa Jr., P., Gomide, F.: Granular Approach for Evolving System Modeling. In: Hüllermeier, E., Kruse, R., Hoffmann, F. (eds.) IPMU 2010. LNCS, vol. 6178, pp. 340–349. Springer, Heidelberg (2010), doi:10.1007/978-3-642-14049-5_35
35. Lemos, A., Caminhas, W., Gomide, F.: Fuzzy Multivariable Gaussian Evolving Approach for Fault Detection and Diagnosis. In: Hüllermeier, E., Kruse, R., Hoffmann, F. (eds.) IPMU 2010. LNCS, vol. 6178, pp. 360–369. Springer, Heidelberg (2010), doi:10.1007/978-3-642-14049-5_37
36. Lemos, A., Caminhas, W., Gomide, F.: New uninorm-based neuron model and fuzzy neural networks. In: 2010 Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS), pp. 1–6 (2010), doi:10.1109/NAFIPS.2010.5548195

37. Lemos, A., Gomide, F., Caminhas, W.: Fuzzy evolving linear regression trees. Evolving Systems 2(1), 1–14 (2011), doi:10.1007/s12530-011-9028-z
38. Lemos, A., Gomide, F., Caminhas, W.: Multivariable gaussian evolving fuzzy modeling system. IEEE Transactions on Fuzzy Systems 19(1), 91–104 (2011), doi:10.1109/TFUZZ.2010.2087381
39. Lemos, A., Kreinovich, V., Caminhas, W., Gomide, F.: Universal approximation with uninorm-based fuzzy neural networks. In: 2011 Annual Meeting of the North American Fuzzy Information Processing Society (NAFIPS), pp. 1–6 (2011)
40. Leng, G., McGinnity, T., Prasad, G.: An approach for on-line extraction of fuzzy rules using a self-organising fuzzy neural network. Fuzzy Sets and Systems 150(2), 211–243 (2005)
41. Lima, E., Hell, M., Ballini, R., Gomide, F.: Evolving fuzzy modeling using participatory learning. In: Angelov, P., Filev, D., Kasabov, N. (eds.) Evolving Intelligent Systems: Methodology and Applications. Wiley-Interscience/IEEE Press (2010)
42. Ljung, L.: System Identification. Prentice-Hall (1999)
43. Lughofer, E.: Extensions of vector quantization for incremental clustering. Pattern Recognition 41(3), 995–1011 (2008); Part Special issue: Feature Generation and Machine Learning for Robust Multimodal Biometrics
44. Lughofer, E.D.: FLEXFIS: A Robust Incremental Learning Approach for Evolving Takagi-Sugeno Fuzzy Models. IEEE Transactions on Fuzzy Systems 16(6), 1393–1410 (2008)
45. Macias-Hernandez, J.J., Angelov, P., Zhou, X.: Soft sensor for predicting crude oil distillation side streams using takagi sugeno evolving fuzzy models. In: IEEE International Conference on Systems, Man, and Cybernetics, pp. 3305–3310 (2007)
46. Miller, R.G.: Simultaneous statistical inference. McGraw-Hill, Inc., New York (1966)
47. de Oliveira, J.V., Pedrycz, W.: Advances in Fuzzy Clustering and its Applications. John Wiley & Sons, Inc., New York (2007)
48. Pedrycz, W., Gomide, F.: Fuzzy Systems Engineering: Toward Human-Centric Computing. Wiley Interscience, NJ (2007)
49. Potts, D.: Incremental learning of linear model trees. In: ICML 2004: Proceedings of the Twenty-First International Conference on Machine Learning, p. 84. ACM, New York (2004), http://doi.acm.org/10.1145/1015330.1015372
50. Pouzols, F., Lendasse, A.: Evolving fuzzy optimally pruned extreme learning machine for regression problems. Evolving Systems 1, 43–58 (2010), doi:10.1007/s12530-010-9005-y
51. Quinlan, R.: Learning with continuous classes. In: 5th Australian Joint Conference on Artificial Intelligence, pp. 236–243 (1992)
52. Rahman, S., Hazim, O.: A Generalized Knowledge-Based Short-Term Load-Forecasting Technique. IEEE Transactions on Power Systems 8(2), 508–514 (1993)
53. Rubio, J.d.J.: Sofmls: Online self-organizing fuzzy modified least-squares network. IEEE Transactions on Fuzzy Systems 17(6), 1296–1309 (2009), doi:10.1109/TFUZZ.2009.2029569
54. Takagi, T., Sugeno, M.: Fuzzy identification of systems and its applications to modeling and control. IEEE Transactions on Systems, Man, and Cybernetics 15(1), 116–132 (1985)

55. Torgo, L.: Functional models for regression tree leaves. In: ICML 1997: Proceedings of the Fourteenth International Conference on Machine Learning, pp. 385–393. Morgan Kaufmann Publishers Inc., San Francisco (1997)
56. Wang, W., Vrbanek, J.: An evolving fuzzy predictor for industrial applications. IEEE Transactions on Fuzzy Systems 16(6), 1439–1449 (2008)
57. Williamson, J.R.: Gaussian ARTMAP: A neural network for past incremental learning of noisy multidimensional maps. Neural Networks 9(5), 881–897 (1996)
58. Yager, R.: A Model of Participatory Learning. IEEE Transactions on Systems Man and Cybernetics 20(5), 1229–1234 (1990)
59. Young, P.: Recursive estimation and time-series analysis: an introduction. Springer-Verlag New York, Inc., New York (1984)
60. Yuan, Y., Shaw, M.: Induction of fuzzy decision trees. Fuzzy Sets and Systems 69(2), 125–139 (1995)
61. Zhou, X., Angelov, P.: Autonomous visual self-localization in completely unknown environment using evolving fuzzy rule-based classifier. In: IEEE Symposium on Computational Intelligence in Security and Defense Applications, CISDA 2007, pp. 131–138 (2007), doi:10.1109/CISDA.2007.368145

Chapter 7
Emerging Trends in Machine Learning: Classification of Stochastically Episodic Events

B. John Oommen and Colin Bellinger

Abstract. In this chapter we report some Machine Learning (ML) and Pattern Recognition (PR) techniques applicable for classifying Stochastically Episodic (SE) events[1]. Researchers in the field of Pattern Recognition (PR) have traditionally presumed the availability of a representative set of data drawn from the classes of interest, say ω_1 and ω_2 in a 2-class problem. These samples are typically utilized in the development of the system's discriminant function. It is, however, widely recognized that there exists a particularly challenging class of PR problems for which a representative set is not available for the second class, which has motivated a great deal of research into the so-called domain of One Class (OC) classification. In this chapter, we primarily report the novel results found in [2, 4, 6], where we extend the frontiers of novelty detection by the introduction of a new field of problems open for analysis. In particular, we note that this new realm deviates from

B. John Oommen
Chancellor's Professor ; *Fellow: IEEE* and *Fellow: IAPR*
School of Computer Science, Carleton University, Ottawa, Canada : K1S 5B6
e-mail: oommen@scs.carleton.ca
The first author is also an *Adjunct Professor* with the University of Agder in Grimstad, Norway. Both the authors are grateful for the partial support provided by NSERC, the Natural Sciences and Engineering Research Council of Canada.

Colin Bellinger
The School of Information Technology and Engineering, University of Ottawa, Ottawa, Canada
e-mail: cbell052@uottawa.ca

[1] The results reported here have already appeared in [2, 4, 6] which were refereed publications. This chapter is a collection of the results in this field. It first briefly describes two-class classifiers and one-class classifiers. It then reports the results currently available for the recognition of SE events. The intention is that this chapter can be a comprehensive record of the state-of-the-art when it concerns SE event recognition.

the standard set of OC problems based on the presence of three characteristics, which ultimately amplify the classification challenge. They involve the *temporal* nature of the appearance of the data, the fact that the data from the classes are “interwoven”, and that a labelling procedure is not merely impractical - it is almost, by definition, impossible. As a first attempt to tackle these problems, we present two specialized classification strategies denoted by Scenarios $S1$ and $S2$ respectively. In Scenarios $S1$, the data is such that standard binary and one-class classifiers can be applied. Alternatively, in Scenarios $S2$, the labelling challenge prevents the application of binary classifiers, and instead dictates the novel application of one-class classifiers. The validity of these scenarios has been demonstrated for the exemplary domain involving the Comprehensive Nuclear Test-Ban-Treaty (CTBT), for which our research endeavour has also developed a simulation model. As far as we know, our research in this field is of a pioneering sort, and the results presented here are novel.

Keywords: Pattern Recognition, Rare Events, Stochastic Events, Erroneous Data.

7.1 Introduction

7.1.1 Problem Formulation

A common assumption within supervised learning is that the distributions of the target classes can be learned, either parametrically or non-parametrically. Moreover, it is assumed that a representative set of data from these classes is available for the training of supervised learning algorithms; indeed, the latter implies the former.

Beyond this commonly-reported method of classification, there exists a special form of Pattern Recognition (PR), which is regularly denoted One Class (OC) classification [11, 13, 14, 16, 25, 26]. This “exceptional” category of binary classification is noteworthy in lieu of the significant challenge that it presents. Escalating the difficulty, is the fact that drawing a representative set of data to compose the second class (ω_2), which is fundamental to the derivation of a binary discriminant function, is abnormally arduous, if not altogether impossible. The difficulty of acquiring a sufficiently symbolic set may arise because of:

1. The natural *imbalance* in the classification task;
2. The difficulty (due to cost, privacy, etc.) of acquiring samples from the ω_2 class;
3. The task of obtaining representative samples of the ω_2 class is overwhelming, as a result of the vastness of the distribution.

PR tasks of this nature have previously been constituted as involving outlier (or novelty) detection in lieu of the fact that the vast majority of the data takes, what is assumed to be, a well-defined form that can be learned, and that samples from the ω_2 class will appear anomalously – outside the learned distribution. Although such problems can be significantly more difficult than those that involve two well-defined classes of data, the results reported in the literature demonstrate that satisfactory results can often be obtained (see [11, 13, 14, 16, 25, 26], for example).

7.1.2 SE Event Recognition

To expand the horizon of the field, we observe that there exists a further, and yet more challenging subset of the OC classification domain of problems, which remains unexplored. We have denoted this class of problems as Stochastically Episodic (SE) event recognition.

The problem of SE event recognition can be viewed in a manner that distinguishes it from the larger set of OC classification tasks. In particular, this category of problem has a set of characteristics that collectively distinguish it from its more general counterparts. The characteristics of this category can be best summarized as follows:

- The data presents itself as a time sequence;
- The minority class is challenging to identify, thus, adding unwarranted noise to the one-class training set;
- The state-of-nature is dominated by a single class;
- The minority class occurs both rarely and randomly *within* the data sequence.

Typically in PR solutions to so-called OC problems, the accessible class, and in particular, the data on which the OC classifier is trained, is considered to be well-defined. Thus, it is presumed that this data will enable the classifier to generalize an adequate function to discriminate between the two conceptual classes. This, for example, was demonstrated in [25], where the training set consisted exclusively of images of non-cancerous tissue. Similarly, in [13], a representative set of the target computer user's typing patterns, which are both easily accessible and verifiable, were utilized in the training processes.

The classification of SE events[2] is considerably more difficult because deriving a strong estimate of the target class's distribution is unfeasible due to the prospect of invalid instances (specifically members of the ω_2 class erroneously labelled ω_1) in the training set. In this work, we present solutions to this problem based on tradition one-class classifiers.

[2] Events of this nature are denoted stochastic because their appearances in the time-series are the results of both deterministic and non-deterministic processes. The non-deterministic triggering event could, for example, be the occurrence of an earthquake, while the transmission of the resulting p- and s-waves, which are recorded in the time-serise, are deterministic.

SE event recognition is additionally challenging because the validity of instances drawn from the target class are suspect, and the occurrences of the minority class are temporally (i.e. with respect to the time-axis) interwoven with the data from the majority class.

7.1.3 Characteristics of the Domain of Problems

To accentuate the difference between the problems that have been studied, and the type of problems investigated in this research, we refer the reader to Table 1. This table displays an assessment of six one-class classification problems, which, while only a small subset, cumulatively illustrate the traditional scope of the problem set. In addition, we include the problem of CTBT verification, which forms our exemplary SE event recogonition problem. The first column indicates whether the problem has traditionally been viewed as possessing an important *temporal* aspect. The three entries with an asterisk require special consideration. In particular, we note that while, traditionally, these domains have not been studied with a temporal orientation, they do indeed contain a temporal aspect. The subsequent column signals whether the manual labelling of data drawn from the application domain is a significant challenge. This is, for example, considered to be a very difficult task within the field of computer intrusion detection, where attacks are well disguised in order to subvert the system.

The following two columns quantify the presence of class imbalance. In the first of these, we apply a standard assessment of class imbalance, one which relies on the determination of the *a priori* class probabilities. Our subsequent judgement departs slightly from the standard view, and considers class imbalance that arises from the difficulty of acquiring measurements (due to cost, privacy, *etc.*). The final column specifies if the minority class occurs rarely, and randomly (in time and magnitude), and if it occurs within a *time sequence dominated by the majority class.*

Table 1 A comparison of well-known One-Class (OC) classification problems. The explanation about the entries is found in the text.

Dataset	Temporal	ID Challenge	Imbalance *Type I*	Imbalance *Type II*	Interwoven
Mammogram	No	Low	Yes	Medium	No
Continuous typist recognition	No	Low	Yes	Medium	No
Password hardening	No	Low	Yes	Medium	No
Mechanical fault detection	No*	Low	Yes	Medium	No
Intrusion detection	No*	High	Yes	High	No
Oil spill	No*	High	Yes	Medium	No*
CTBT verification	Yes	High	Yes	High	Yes

To summarize, in this section we have (briefly) both demonstrated the novelty of this newly introduced sub-category of PR problems, and positioned the CTBT verification task within it. We additionally note that the fault detection, intrusion detection, and oil spill problems could be reformulated to meet the requirements of our proposed category. This, indeed, suggests a new angle from which these problems can be approached.

7.1.4 Overview of Our Solution

As previously indiated, SE event recognition composes a particular challenging problem due to the combined affect of the four characteristics that are inherent in such problems. Under these circumstances, we envision two possible techniques for discriminating between the target class and the stochastically episodic events of interest. If the incoming training data contains a sufficient quantity of accurately identifiable stochastic events, a standard clustering/PR algorithm could be applied to label both the classes appropriately. Subsequent to the labelling procedure, a standard binary classifier could be trained and utilized to achieve the classification of novel instances. In this body of work, we refer to this scenario as S1, and the subsequent scenario as S2.

Alternatively, and more applicable in scenarios in which the SE events are extremely rare, all of the training data can be assigned to the target class, and an OC classifier can be applied. The details of, and justification for, this approach are described in the subsequent sections. Our primary objective in this chapter is to illustrate how standard supervised learning algorithms can be applied to discriminate rare stochastic episodes, which apart being unanticipated, are random in magnitude and position within the sequence of background data.

Put in a nutshell, the novel contributions of the results presented in [4] and [5] (which we re-iterate here), with respect to PR, are as follows:

- We introduce an important new category of PR, namely SE event recognition. In particular, we note that this new realm deviates from the standard set of one-class problems based on the presence of four characteristics: *(a)* the data presents itself as a time sequence; *(b)* the minority class is challenging to identify, thus, adding unwarranted noise to the OC training set; *(c)* the state-of-nature is dominated by a single class; and, *(d)* the minority class occurs both rarely and randomly within the data sequence.
- In addition, we present a first attempt at classifying SE events within the examplary verification problem suggested by the Comprehensive Test-Ban-Treaty (CTBT). Our initial approach is extremely accessible, as it is based on "off the shelf" PR solutions.
- More specifically, where the ω_2 is sufficiently large, we demonstrate how clustering/PR algorithms can be applied to label training data for the development of a sound binary classifier.

- Finally, in scenarios where training instances cannot be acquired from the second class (the so-called OC problem), and where the accessible class in known to contain noise due to labeling issues, we illustrate how, through novel means, standard OC classifiers can be applied as unsupervised learners.

We conclude this section by mentioning that our results probably represent the state-of-the-art when it concerns recognizing SE events.

7.2 Pattern Recognition: State of the Art

This section[3] serves to present the state-of-the-art in PR. In that regard, Duda, *et al.*, in [10] describe pattern recognition as follows:

"The act of taking in raw data and taking an action based on the 'category' of the pattern."

It is, indeed, natural that we should desire to 'teach' machines to recognize sets of patterns that are easily recognizable to humans, such as handwritten characters, speech and faces, as computers present the possibility of increased efficiency and do not become tired of mundane tasks. Furthermore, the benefits of training machines to classify complex patterns, typically left to doctors and scientists with considerable specialization in the domain, are equally apparent. Thus, researchers have continued to push the state-of-the-art in PR systems since the advent of the modern computer.

7.2.1 Supervised Learning

Prior to application, the PR system must be trained to discriminate between the objects of interest in its particular application domain. For multi-class problems, such as discrimination between handwritten characters, the PR system is said to learn a mapping that discriminates between the individual inputs by directing them to their corresponding categories. Alternatively, in the special scenario, which is of primary interest in this work, termed OC learning, instances of a single target category are available for the training of the PR system. As a result, the system takes a recognition-based approach, and attempts to learn a function that maps novel instances of the target category to the target class, and all others to the outlier class.

Broadly speaking, standard PR systems for supervised learning are trained on datasets drawn from their prospective application domains, in which each

[3] This brief section has been included in the interest of completeness. Although these issues are considered commonplace for the general PR problem, they are still fairly non-standard for OC problems - which advocates the necessity of the section.

feature vector has been accented with its corresponding class label. The objective of the training process is the derivation of a set of models that articulate the individual characteristics of the classes. Thus, while the performance on the training set is of little interest, rather, the focus shifts to the selection of a model that will perform well on novel instances in the future. The derivation of these models is algorithm-specific, however, there exists commonalities between all learners. Generally speaking, regardless of the learning strategy, the accuracy of the derived model on novel instances will increase with the size of the training set. In addition, all learners strive to optimize the balance between specialization and generalization [18].

Under ideal circumstances, the training procedure for a binary learner is able to rely on an ample supply of data that has been uniformly drawn from both classes. As a result, increasingly accurate models of the classes in question can be constructed, and therefore, an effective classifier of novel instances is produced.

7.2.1.1 "Traditional" Pattern Recognition

Standard, or rather "traditional" PR problems/solutions typically assume the existence of data that was drawn independently and identically from the application domain, and that the data can be divided upon class lines into representative sets. The availability of such data facilitates the training of binary classifiers, which have been shown to be proficient at learning class distributions, and thus at labelling novel instances.

In all brevity, we mention that the binary classifiers used in this study were the Multi-layer Perceptron (MLP), the Support Vector Machine (SVM), the Nearest Neighbour (NN), the Naïve Bayes (NB) and the Decision Tree (J48), all of which are fairly well known, and so their descriptions are omitted here. However, we mention that their implementations were from Weka[4].

Alternatively, OC classifiers rely on instances drawn from a single class in the derivation of a discriminant function. A broad set of OC classifiers exists in the literature, each of which applies a slightly different strategy to the construction of a binary discriminant function from a single class. However, in simple terms, the process can be articulated as one in which the selected classifier learns to recognize, in some general terms, novel instances that are similar to those viewed during the training process. Thus, novel instances that do not appear to fit into the learned distribution are designated to the alternate class, ω_2 .

The autoassociator (AA), for example, applies a neural network structure to compress/decompress instances of the concept class exclusively. Thus, an unsuccessful compression/decompression results in the instance being assigned to the second class [14].

Hempstalk *et al.*, in [13], converted the one-class classification problem into binary tasks by estimating the distribution of the concept class and

[4] Interested readers should refer to [10, 12, 18] for more details of these strategies.

generating instances of the non-concept, accordingly. Finally, a standard binary classifier is trained. This process has been denoted the Combined Probability and Density Estimator (PDEN).

Alternatively, the one-class Nearest Neighbour (ocNN) algorithm [8] learns a *target rejection rate*, τ, where τ is the distance between the two nearest neighbours with the greatest separation in the training data. Subsequently, all novel instances whose nearest neighbours are at greater distances than τ are classified as outliers. We have additionally implemented a modified version of the ocNN in Weka, and denoted it as the scaled ocNN (socNN). Contrary to the ocNN, the socNN classifier is capable of learning a model that accounts for the noise in the training set (we refer the reader to [5] for more details on how this is achieved). Subsequent research also explored the performance of the often extolled one-class SVM [23].

Our previous work, as highlighted here, demonstrated that for extreme cases of SE event recognition, the application of static OC classifiers is superior to the static binary classifiers mentioned above. These results, however, do not preclude those that may be produced through other learning strategies, such as those discussed in the following subsections. Indeed, the failure to incorporate the time dimension into the hypothesis space is a particular weakness of both the binary and OC solutions. This is particularly the case as the temporal nature of the phenomena suggests that valuable information may have been lost. Moreover, under certain conditions, sampling, unsupervised and semi-supervised learning strategies have been shown to be beneficial. Thus, they also warrant further consideration.

7.2.2 Alternative Learning Paradigms

Under less than ideal circumstances, however, none or very few labelled training instances are available. The former case, where no labelled instances are available, is referred to as *unsupervised* learning. In the context of data categorization, unsupervised learning typically relies on clustering algorithms [28]. The latter scenario, which is commonly known as *semi-supervised* learning, is characterized by the availability of a small set of labelled training instances. The objective, here, is to leverage the knowledge stored in the labelled instances to incrementally categorize the larger unlabelled portion of the training set [7].

In terms of SE event recognition, if we assume the unknown target distributions be static, which we maintain in this early work, then semi-supervised learning offers a means by which we can extract more labelled SE events from the training data. These additional SE events can subsequently be applied to a learn an improved model.

Semi-supervised learning for data labelling has previously been applied in [7]. Unfortunately, in scenario S2, due to the extreme class imbalance, even a perfect *a priori* labelling phase is unlikely to produce a sufficient number of SE event instances for binary learning. This is, quite literally, a result of the fact that they do not exist in nature.

7.2.3 Sampling

Sampling is often applied in multi-class classification problems as a means to overcome under-representation in one or more of the classes [17]. As previously noted, significant imbalance in the training set can, very often, lead to the development of a poor classification model. In addition, some evaluation metrics are biased by class imbalance [15].

In order to deal with the issue of under-representation in the training data, two sampling strategies have received considerable attention in the literature. Oversampling increases the size of the minority portion of the training set by sampling from that portion, with replacement. As a result, some instances of the minority class will appear more than once in the regenerated training set. The oversampling process is continued until a "sufficient" portion of the training data is drawn from the minority class.

The primary limitation of this form of sampling is that there is a risk of over-representing noisy and/or outlier instances in the generated training set. As a result, the learned function may become overly fit to the minority class, and thus be, possibly, biased in a manner that leads to poor classification.

Alternatively, under-sampling draws a subset of instances (without replacement) from the majority class, to produce a more balanced training set. The major problem with such an approach is that there is no guarantee that the most informative instances will, in fact, be sampled. Thus, depending on the degree to which the majority class must be under-sampled, there is a high risk of discarding meaningful information.

Modifications to these strategies have attempted to address the above limitations. However, there is no single solution. Moreover, with respect to SE event recognition, the degree of class imbalance is so great that over-sampling is very much unadvised. Similarly, the large amount of background data implies that an excessive amount of under-sampling would be required at great expense to the informativeness of the training set. This is a particular problem given the complexity of the background class.

7.2.4 Dynamic Classification

Sequential classification algorithms constitute a special form of classifiers, which are particularly apt at leveraging systematic variations in data, such as time series data. When applied to the appropriate problem, this advanced learning process can often produce results that are superior to those obtained by the static algorithms previously discussed.

In general terms, time series classification is conducted in one of two ways. The simplest strategy is to convert the sequence into a form suitable for static learning. This, in essence, ignores the sequential aspect and, thus, loses many of the desirable features of dynamic classification.

Alternatively, thresholds on the distance between the learned class pattern and those to be classified can be applied.

In addition, algorithms based on artificial neural networks and hidden Markov models can be trained to detect anomalous events and make predictions into the future based on the data seen thus far [19, 20].

Due to the systematic variations in SE event data, times series classification and anomaly detection offer considerable promise within this domain. However, for the present, we leave this for future work.

7.3 Modelling the Problem

To this point, we have described a novel sub-category of PR, which is characterized by the detection of a minute number of SE events interwoven in a time-series. Indeed, a number of interesting PR problems fit this form, including advanced earthquake, tsunami and machine failure warning systems, to name but a few. In this section, we present a series of experiments based on the verification of the CTBT. These experiments are designed to both illustrate the domain of SE events, and to exhibit a first attempt at SE events recognition.

7.3.1 Application Domain

The CTBT aims to prevent nuclear proliferation through the banning of all nuclear detonations in the environment. As a result, a number of verification strategies are currently under study, aimed at ensuring the integrity of the CTBT. The primary verification technique being explored relies on the quantity of radioxenon measured continuously at individual receptor sites, distributed throughout the globe. Radionuclide monitoring, in general, has been identified as the sole technique capable of unambiguously discriminating low yield nuclear detonations from the background emissions. More specifically, verification of the treaty based on the four radioxenon isotopes, ^{131}Xe, ^{133}Xe, ^{133m}Xe and ^{135}Xe, has been promoted due to the relatively low background levels, their ideal rates of decay, and their inert properties [22, 24].

In general, the measured radioxenon levels are expected to have resulted from industrial activities, such as nuclear power generation and the production of medical isotopes. However, they are also the byproducts of low yield clandestine nuclear weapons tests, which are the subject of the CTBT.

7.3.2 Procuring Data: Aspects of Simulation

While it is generally beneficial to develop and study classifiers on “real” data, this is, indeed, impossible within the CTBT verification problem due to the absence of measured detonations, and the limited availability of background instances. It has, however, been demonstrated that artificial data can be utilized for PR system development, and to generate controlled experiments (generalized case-studies), in the absence of "real" measurements [1, 9]. In this vein, as a means of acquiring experimental datasets for this research, we utilized the simulation framework presented by Bellinger and Oommen in [6]. Their simulation framework models SE events, such as earthquakes, nuclear explosions, etc., as they propagate through the background noise, in this case representing radioxenon emitted from the industry into the earth's atmosphere.

7.3.2.1 Simulation Scenario

In order to explore the PR of low yield clandestine nuclear tests, we devised a simulation scenario to capture the effects of a diverse set of detonation possibilities, within a realistic background scenario. In particular, and accordance with the majority of the CTBT's International Monitoring Station (IMS), the IMS in the simulated environment was impacted by a single industrial emitter. In this simulation scenario, the industrial emitter was positioned 3,000 km away from the IMS. Thus, when the atmospheric conditions transported the emitted radioxenon directly from the source to the receptor, and when the conditions were not conducive to the dispersion of the radioxenon, the background concentration could reached significant levels. However, due to the realistic atmospheric conditions that were built into the model, such as the fluctuations in wind speed and direction, along with atmospheric stability, the background levels were generally low. This fact is displayed by the histogram in Figure 1. The figure specifically demonstrates that the majority of the ^{131}Xe concentrations measured at the IMS site during the simulation were less than 0.5 $Bq\ m^{-3}$.

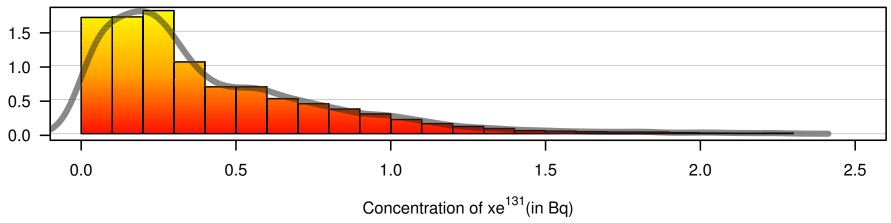


Fig. 1 This figure displays a histogram of the measured concentrations of ^{131}Xe at the IMS, resulting from the background source during the simulation.

It is, however, highly probable that a clandestine detonation will occur at distances beyond the industrial source, thus, causing no, or only a minute, change in the radioxenon concentrations measured at the IMS, depending on the angular direction to the detonation site, and the prevailing meteorological conditions. Therefore, the classification of this type of SE event is extremely challenging.

With the above fact in mind, we considered the performance's of the PR systems as a function of distances. This is to specifically assess the probability of detecting detonations at various distances. In particular, 23 subcategories of datasets were generated. In each case, the modelled environment contained the same industrial source and IMS at the receptor site. As a result, for each simulation the background readings can be assumed to follow the distribution displayed in Figure 1. The 23 subsets formed a series of incremental detonation ranges, which commenced with all detonations occurring between 500 km and 1000 km, as illustrated in Figure 2.

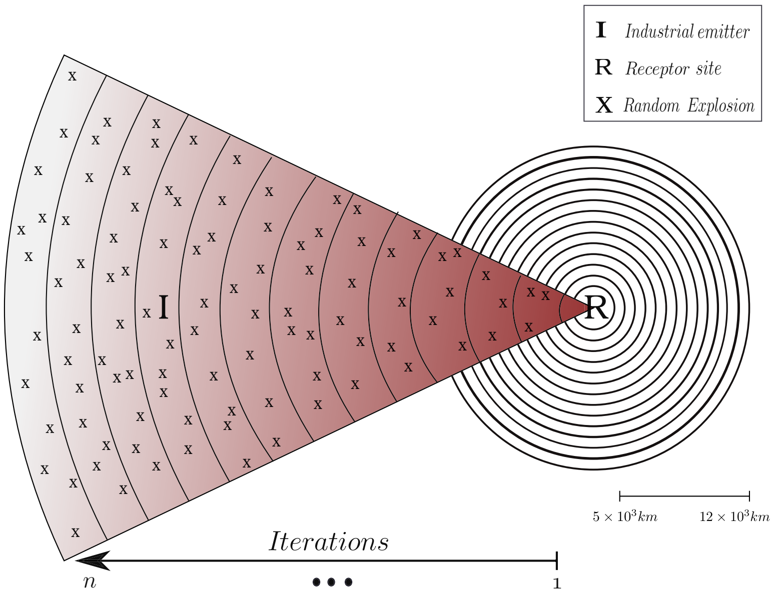


Fig. 2 This figure demonstrates the iterative composition of the simulated domain. In each iteration of the simulation, a fixed number of explosions are probabilistically generated as uniform, random events in time, space and magnitude, and dispersed according to the prevailing meteorology, which may or may not carry the pollutant cloud past the receptor site

The detonation range was iteratively increased by 500 km for each successive set. This incremental approach enabled the examination of performance as a function of distance, in addition to the more general considerations of performance.

As a binary classification problem, the generated sets were composed of two classes, in this case a background class and a detonation class. In addition to the class label, each instance was composed of the concentrations of the four isotopes measured by the IMS at the receptor site over the period of an hour. The simulation system contains two phases, the first phase simulates the effect of the background emission source on the receptor sites, thereby producing instances of the background class (labelled 0). Thus, an instance measured over hour i, takes the following form:

$$\mathbf{x}_{i,0} \;=\; {}^{131}Xe_{i,0},\; {}^{133}Xe_{i,0},\; {}^{133m}Xe_{i,0},\; {}^{135}Xe_{i,0},\; 0. \tag{1}$$

The second phase generates the data for the detonation class (labelled 1). This is done by generating random (in time, space and magnitude) low yield explosions, and measuring their impact on the receptor site. Subsequently, the effect of the detonation is combined with that of the background source over the appropriate period of time, and written to the dataset with the detonation label. Therefore, a detonation instance measured over hour j, takes the following form:

$$\mathbf{x}_{j,1} = \mathbf{x}_{j,0} \;+\; {}^{131}Xe_{j,1},{}^{133}Xe_{j,1},{}^{133m}Xe_{j,1},{}^{135}Xe_{j,1},\; 1. \tag{2}$$

7.3.3 Generated Datasets

A total of 230 datasets were derived and applied to scenario S1 and S2, according to the simulation procedure previously described. More specifically, 10 datasets were generated for each of the 23 detonation ranges, each of which was subsequently divided into training and testing components.

Intuitively, the first scenario presents a slightly easier classification problem, because a set, albeit small, of SE events can be extracted from the application domain and applied to train and/or test the PR systems. More specifically, within this scenario, we assume that the ω_2 class is both identifiable and available in quantities that facilitate the training of binary classifiers. However, in many ways, the classification problem still presents itself as a so-called OC classification task, and thus warrants exploration on both fronts. The datasets specifically contain a 90% background data (ω_1) and 10% explosion data (ω_2).

Alternatively, each set involved in the S2 scenario is divided with 99% background data (ω_1) and 1% explosion data (ω_2). In order to simulate the challenge of manually labelling the instances drawn from class ω_2, and in accordance with the disguised nature of the SE events, all of the ω_2 training instances were erroneously labelled ω_1.

Alternatively, the test sets included appropriately labelled instances from both classes, with proportions following the predefined states-of-nature. This enabled us to assess each classifier's ability to generalize the "real" background data from the noisy training set.

7.4 PR Solutions

In this section, we present a series of experiments designed to both illustrate the demonstration domain, and to exhibit a first attempt at classifying this sub-category of PR problems.

7.4.1 Classification Scenarios

As mentioned in the introductory section, within this challenging domain of classification problems, there exist two conceivable scenarios, which we have denoted as S1 and S2. These scenarios explicitly influence the choice of the classification scheme applied to the task of recognizing the SE events.

Intuitively, the first scenario presents a slightly easier classification problem, because a set, albeit small, of SE events can be extracted from the application domain and applied to train and/or test the PR systems. More specifically, within this scenario, we assume that the outlier class is both identifiable and available in quantities that facilitate the training of binary classifiers. However, in many ways, the classification problem still presents itself as a so-called OC classification task, and thus warrants exploration on both fronts.

Alternatively, the second scenario presents itself as a much more difficult PR task, and in many ways more accurately reflects the PR problem suggested by the detection of SE events, in general, and the verification of the CTBT, in particular.

In accordance with the general domain characteristics, as they were originally defined, the data presents itself as a time-series of background measurements that are interwoven with a minute number of SE events. However, unlike the ideal scenario depicted in S1, here we attempt to assume a state-of-nature that is more appropriate for the CTBT task. In particular, we assume that there is a 1% *a priori* probability of a detonation, which, while still an overestimate, is a more accurate depiction, while it still provides insight into the behaviour of PR systems on the class of SE events.

Raising the difficulty further, is the recognition that, in practice, the clandestine nature of the SE events are such that manually identifying a distant clandestine occurrence in the acquired time-series of readings is extremely difficult, if not impossible. Thus, this prohibits the derivation of a labelled training set, which dictates that practitioners are left to utilize a training set composed largely of background instances, but with a minute number of *unidentifiable* members of the SE event class.

In the absence of a labelled training set, we propose the application of standard OC learners as unsupervised classifiers. When applying OC classifiers to an unlabelled training set, the practitioner must rely on the knowledge of a domain expert to acquire estimates of the *a priori* class probabilities.

In particular, estimates of the state-of-nature are required to appropriately specify the parameters of the OC classifiers, such as the rejection rate, or error rate. This technique aims to prevent the inclusion of the SE event instances in the generalized description of the background class. Our reliance on an error, or rejection rate, presumes that the SE events will reside on the periphery of the background class, and thus, by marginally tightening the generalization of the background class, those instances of the SE event class will no longer be included.

7.4.2 Classification

Standard PR problems typically assume the existence of data that was drawn independently and identically from the application domain, and that the data can be divided upon class lines into representative sets. The availability of such data facilitates the training of binary classifiers, which have been shown to be proficient at learning class distributions, and thus at labelling novel instances.

In all brevity, we mention that the binary classifiers used in this study were the Multi-layer Perceptron (MLP), the Support Vector Machine (SVM), the Nearest Neighbour (NN), the Naïve Bayes (NB) and the Decision Tree (J48), all of which are fairly well known, and so their descriptions are omitted here. However, we mention that their implementations were obtained from Weka.

Alternatively, OC classifiers rely on instances drawn from a single class in the derivation of a discriminant function. A broad set of OC classifiers exists in the literature, each of which applies a slightly different strategy to the construction of a binary discriminant function from a single class. However, in simple terms, the process can be articulated as one in which the selected classifier learns to recognize, in some general terms, novel instances that are similar to those viewed during the training process. Thus, novel instances that do not appear to fit into the learned distribution are designated to the ω_2 class.

Although these classifiers were briefly outlined earlier, to summarize:

- The autoassociator (AA), for example, applies a neural network structure to compress/decompress instances of the concept class exclusively. Thus, an unsuccessful compression/decompression results in the instance being assigned to the second class [14].
- Hempstalk *et al.*, in [13], converted the OC classification problem into binary tasks by estimating the distribution of the concept class and generating instances of the non-concept, accordingly. Finally, a standard binary classifier is trained. This process has been denoted the Combined Probability and Density Estimator (PDEN).
- Alternatively, the one-class Nearest Neighbour (ocNN) algorithm [8] learns a *target rejection rate*, τ, where τ is the distance between the two nearest neighbours with the greatest separation in the training data. Subsequently, all novel instances whose nearest neighbours are at greater distances than τ are classified as outliers.
- We have additionally implemented a modified version of the ocNN in Weka, and denoted it as the scaled ocNN (socNN). Contrary to the ocNN, the socNN classifier is capable of learning a model that accounts for the noise in the training set.
- Subsequent research also explored the performance of the often extolled one-class SVM [23]. However, due to the poor results which were generally equivalent to those yielded by the ocNN, it is not included in the present discussion.

7.4.3 Classifier Assessment Criteria

As discussed in the previous section, this research considers the performance of the classifier within two distinct scenarios. Within each of the scenarios, namely S1 and S2, we considered the performance of the classifier according to a set of criteria. These criteria are discussed in greater detail.

In particular, we examined the general performance of the classifiers across all of the simulated detonation ranges. Performance in this category is particularly important, as, in practice, the detonation ranges are largely unpredictable. The results of this assessment are presented in Sections 7.5.1 and 7.6.1. In addition, we explored the performance of the classifier within two shorter detonation ranges, the result of which is presented in Sections 7.5.2 and 7.6.2.

The performance of the classifier, as a function of distance, was also examined. The results of this comparison are detailed in Sections 7.5.3 and 7.6.3.

Finally, in light of the inherent challenge of distinguishing these two very similar classes according to the four radioxenon isotopes, we were motivated to explore an expanded CTBT feature space. Based on the significant role held by meteorology in affecting the pollutant levels at the receptor site, we surmised that the inclusion of meteorological features would improve the performance of the classifiers. The results of our experiments with an expanded feature space are provided in Sections 7.5.4 and 7.6.4. Indeed, subsequent research by Bellinger and Japkowicz, in [3], further demonstrated that the inclusion of a simple wind direction feature can significantly increase the prospect of classifying challenging detonation events, and suggests the predictive power of meteorological features in general. In doing so, they presented classification results from four complex simulated scenarios.

7.5 Results: Scenario 1

In this section, we present the results that were obtained according to the four assessment criteria that were motivated in the previous section, on the first classification scenario, S1. We commence our exploration of PR performance by examining the Area Under the ROC Curve (AUC) scores produced by each classifier over the 23 detonation ranges.

7.5.1 General Performance

In this section, we present a general overview of the performance levels of each of the considered classifiers on the simulated CTBT domain. More specifically, we present an assessment of the five binary classifiers and the four one-class classifiers, in terms of their AUC scores averaged over the 230 datasets that spanned the 23 detonation ranges. In light of the fact that the SE events, which are to be identified, will, in practice, occur at random and unpredictable distances, these results are a particularly insightful overview of the general performance levels.

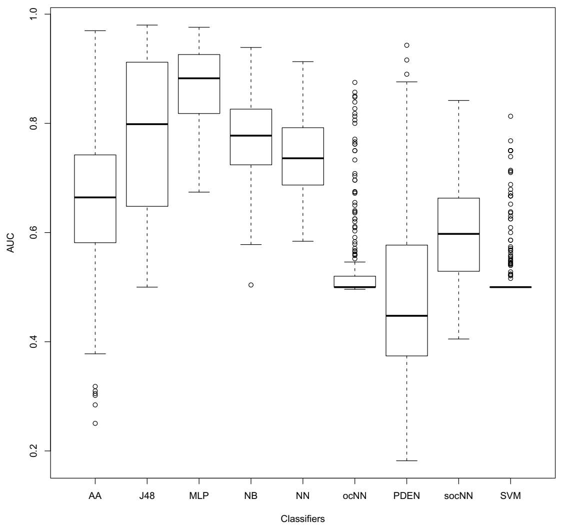


Fig. 3 This figure displays the performance of the nine classifiers, in terms of their AUC scores on the 230 generated CTBT datasets, in the form of a series of boxplots.

The results depicted in Figure 3 were compiled as a series of boxplots; one for each classifier.

The solid lines that bisect the boxes represent the median AUC score produced by the particular classifier. The box itself indicates the distribution of the middle half of the AUC scores produced by the classifier. Thus, it stretches from the 25th percentile (at the lower hinge) to the 75th percentile (at the upper hinge). The boxes that are evenly divided indicate that the classifier's scores are evenly distributed throughout the central region. This is, indeed, the case for AA and NB.

The fact that there is no box around the median indicator for the SVM, suggests that nearly all of the AUC results were equivalent, and in this case, approximately 0.5. The relatively large number of circles extending up from the median, individually identify outliers. This suggests that, in general, the SVM classifier performed poorly, but that it occasionally produced anomalously strong results, which stretched slightly beyond 0.8.

Alternatively, the scenario where the median does not produce an even bisection of the box indicates that the distribution of the inter-quartile range is skewed. This is the case, for example, with PDEN, where the upper-quartile is large, indicating that the points composing the upper-quartile are spread over a larger distance.

The dashed lines, or whiskers, stretch to either the maximum and minimum values, where outliers do not exist, or to 1.5 times the range of the inter-quartile region in scenarios with outliers, such as in the case with the SVM classification results.

The SVM classifier is, surprisingly, by far the worst-performing classifier on this data, and in spite of its bias, it is, on average, worse than the OC classifiers, AA and socNN. This is reiterated in Table 2, which contrasts the mean AUC scores of AA and socNN as 0.656 and 0.603, respectively, with the mean value for the SVM classifier being 0.528. Moreover, all four OC classifiers appear to be superior to the SVM when considered in terms of their maximum AUC scores.

When assessing the classifiers according to the boxplot, the median value provides a good indication of their performances, in general. However, most interesting are the ranges of the inter- and outer-quartiles along with the presence of the outliers, when combined with a high median value, as these components provide a strong indication of how likely it is that the classifiers will reproduce the median result.

In these terms, the binary classifier, the MLP, stands out as the superior classifier, with J48, NN, and NB contending for the intermediate positions. The results posted in Table 2 confirm that the MLP is the strongest of the classifiers considered here. Furthermore, it indicates that the J48 and NB are very similar, and that the NN is the fourth-ranking binary classifier according to the mean and maximum scores. However, the NN is second when ranked according to the minimum AUC scores.

Table 2 This table displays the general classification results, in terms of AUC.

	Mean	Max	Min	STDV
NB	0.772	0.939	0.504	0.074
MLP	0.869	0.976	0.674	0.067
NN	0.741	0.913	0.584	0.071
J48	0.774	0.98	0.500	0.148
SVM	0.528	0.813	0.500	0.065
ocNN	0.540	0.875	0.496	0.087
PDEN	0.487	0.943	0.182	0.156
socNN	0.603	0.842	0.405	0.094
AA	0.656	0.970	0.251	0.140

Notably, of the set of OC classifiers, the PDEN produced the most variable range of the AUC scores. It is our suspicion that this variability resulted from the PDEN's generation of an artificial second class in its training process. However, further exploration of this matter is required.

In general, the AA classifier is identified as the strongest OC classifier, both with respect to its mean and median values. While the socNN classifier achieved the second highest mean, it is more stable than the AA, and does not produce any anomalous results. Indeed, the socNN has a lower standard deviation, and furthermore, its boxplot spans a smaller range.

7.5.2 Performance on Short- and Long-Range Detonations

In Figure 4, we present the AUC results produced over two detonation ranges of particular interest. The Boxplot on the left in this figure contains the results for the datasets that included detonations ranging from 1,000 km and 5,500 km, while the Boxplot on the right has those with detonations between 5,500 km and 10,000 km. Together, these plots contrast the performance of the individual classifiers in the various detonation ranges. This experimental setup demonstrates one technique through which the performance of various receptor network topologies can be examined. For example, if PR within the second range is found to be a considerable challenge, the shorter range may, perhaps, be considered an upper bound on the acceptable distance between receptors.

There are two factors at play when hypothesizing about classifier performance within these ranges. Intuitively, detonations closer to the receptor site will be more visible at the receptor site, provided the meteorological conditions are such that the emissions are advected in the direction of the receptor. Conversely, detonations that occur farther afield are likely to have a smaller influence on the pollutant levels at the receptor site, leading to a more challenging classification problem. On the surface, then, it appears that nearby detonations should be easier to detect. Indeed, the very near detonations are often easily identifiable. However, the scenario is made more complex by the fact that during the simulation, the industrial source was positioned approximately in the middle of the shorter range. Thus, there was, in a sense, a great deal of competing background noise to distort the signal.

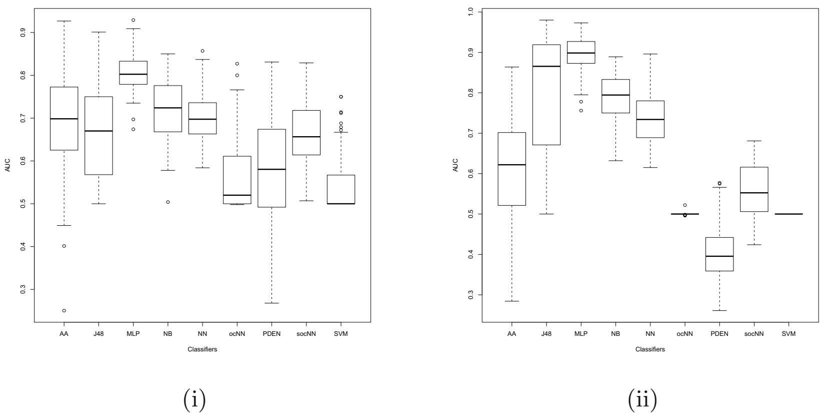


Fig. 4 In this figure, Boxplot (i) displays the performance of the nine classifiers, in terms of their AUC scores for detonations occurring between the distances of 1,000 km and 5,500 km, and Boxplot (ii) displays their performances for detonations between the distances of 5,500 km and 10,000 km.

Indeed, Figure 4 demonstrates that within this scenario it is possible for the performance of the classifiers to improve when detonations occur at greater distances. However, the fact that this only occurred for the binary classifiers, highlights the importance of the second class in the learning process. It turns out that the majority of the binary classifiers are able to, through the training process, utilize the low concentration instances of the detonation class, which resulted from explosions at great distances, to specialize their models to the counter-intuitive point where many of the instances with low concentrations were correctly identified as explosions.

Alternatively, the figure suggests that neither the one-class classifiers, nor the SVM, were able learn a model with this characteristic. Moreover, the SVM exclusively produces AUC scores of 0.5 within the second range, and the ocNN's performance was nearly equivalent. Finally, at greater distances, the PDEN's performance fell even further, with only a minute number of instances exceeding an AUC of 0.5.

Within the shorter range, it is notable that the stronger OC classifiers, namely the AA and socNN, are very comparable with most of the binary classifiers. However, the distinction in favour of the binary learners is emphasized for the larger detonation range.

7.5.3 Performance as a Function of Distance

In this sub-section, we present the performance of the classifier as a function of distance, where the performance is assessed both according to the AUC and the False Positive Rate (FPR).

A *false positive* occurs when the classifier mislabels a novel instance as a member of the positive class (in this case, a member of the background class), when it is, in fact, a member of the negative class (specifically, a member of the SE event class). Thus, the FPR is the total number of false positives over the total number of negative instances. As a metric, the FPR provides insight into whether the model is overly biased towards the positive class, which is a significant risk when the problem is extremely imbalanced.

These results are particularly interesting, as they provide greater insight into performance trends. Moreover, these suggest a performance scale for successively sparser receptor networks, and enable the interested parties to weigh the cost of receptor stations against the probability of detection.

The performance plots depicted both in Figure 5 and Figure 6 were produced by calculating the ensemble mean of each classifier's performance at the 23 detonation ranges, and then through the extrapolation of a performance function.

Within Figure 5, the MLP classifier is identifiably the superior classifier when compared to the remaining four binary learners in terms of the AUC, across the range of detonation distances. In addition, it is not subject to the abrupt fluctuations that J48, and to a lesser extent, NB, incur.

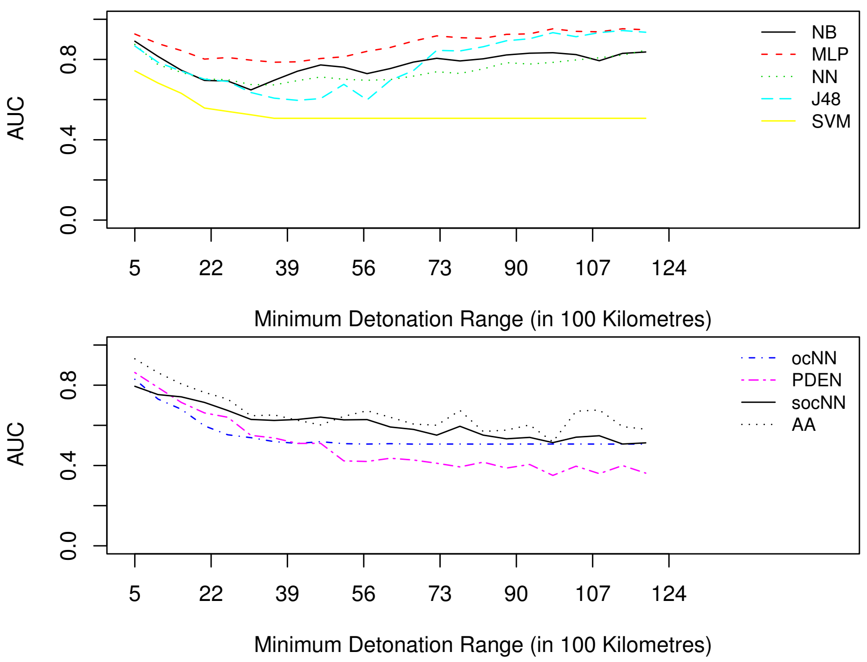


Fig. 5 In this figure, the plot on the left displays the performance of the five binary classifiers, in terms of their AUC scores, as a function of distance. Similarly, the plot on the right displays the performances of the four one-class classifiers as a function of distance, according to their AUC scores.

All of the classifiers, with the SVM appearing as the sole exception, have notable hulls in their performance curves that extend over varying distances and to distinct depths. In each case, a slow descent begins immediately, and is subsequently accompanied by a slow ascent. Alternatively, the SVM classifier suffers from a similar initial decline. However, it fails to recover from the degradation at greater distances.

In each case, the position of the performance hull roughly corresponds to the radial distance between the industrial source of radioxenon and the receptor site. Thus, this suggests that detonations occurring at approximately the same radial distance as that of the primary background emitter are a significant challenge for the detection systems.

The plot on the left in Figure 5 confirms our previous findings, which identified the MLP as the top classifier in this domain, the SVM as the worst, and the remaining three classifiers as contenders for the inner rankings. Indeed, while there are notable differences in the AUC plots for the J48, the NB, and the NN, the fact that their functions cross at numerous points, prohibits the derivation of a general ranking over the entire range of distances.

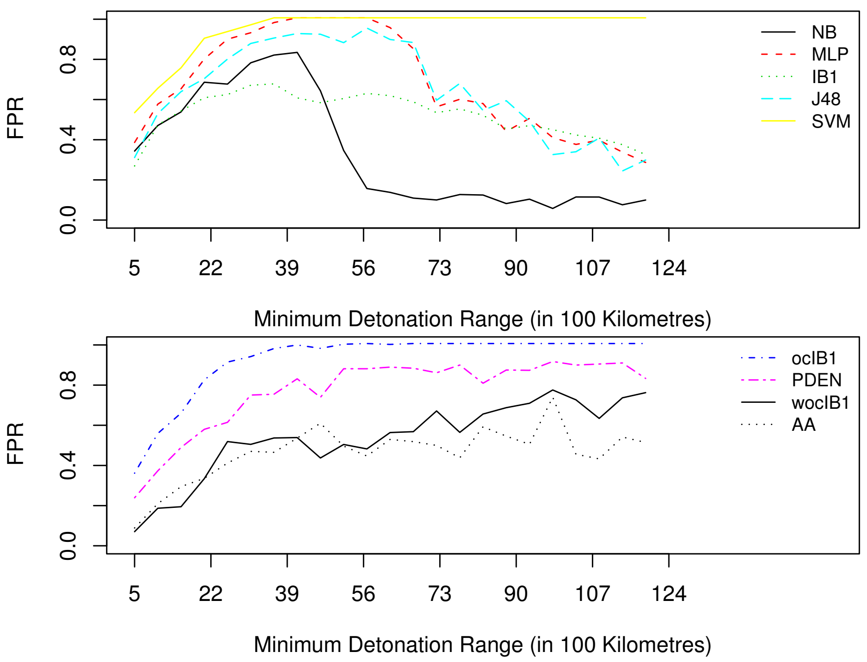


Fig. 6 In this figure, the plot on the left displays the performance of the five binary classifiers, in terms of their FPR scores, as a function of distance. Similarly, the plot on the right displays the performances of the four one-class classifiers as a function of distance, according to their FPR scores.

The plot on the right in Figure 5 presents the performance of the one-class learners as a function of distance. In general, the plot demonstrates that all of the one-class classifiers follow a similar downward trend from their initial peaks, which occurred between 0.8 and 0.9, towards, or beyond in the case of the PDEN, an AUC of 0.5.

Moreover, the performance functions are broadly divisible into two categories. Both the ocNN and the PDEN descend relatively quickly, while the AA and the socNN degrade in a slower, more linear fashion. Therefore, the AA and the socNN are the more suitable of the four one-class learners, with the AA appearing generally superior to the socNN.

The performance of the nine classifiers, measured in terms of the FPR metric, are plotted as a function of distance in Figure 6. In this figure, the plot on the left emphasizes the significant challenge incurred by the binary learners when the detonations occur at a distance similar to the noise source. Although we previously identified the MLP as the strongest binary classifier on this domain, for a relatively broad range (roughly between 25,000 km and 65,000 km), the vast majority of instances, which are truly of the detonation class, were assigned to the background class. The results are similar for J48.

Interestingly, NB has the smallest area under its FPR curve. Thus, it least often identified members of the SE event class as background noise. While we do not consider the FPR results to be individually sufficient for model selection, they do provide some very intriguing insight into the behaviour of the classifiers.

The trends for the one-class classifiers in the plot on the left follow much the same trends previously seen in Figure 5. In particular, the AA and the socNN are superior to the PDEN and the ocNN. However, the distinction between the AA and the socNN is less clear.

7.5.4 Expanded Feature-Space

Through our exploration of this most interesting of classification problems, we recognized both the inherent challenge presented in the classification of SE events that are interwoven in background noise, and the role of meteorology in effecting the very noise levels that make the task so difficult. Our extensive consideration of this application domain has led us to identify the particularly

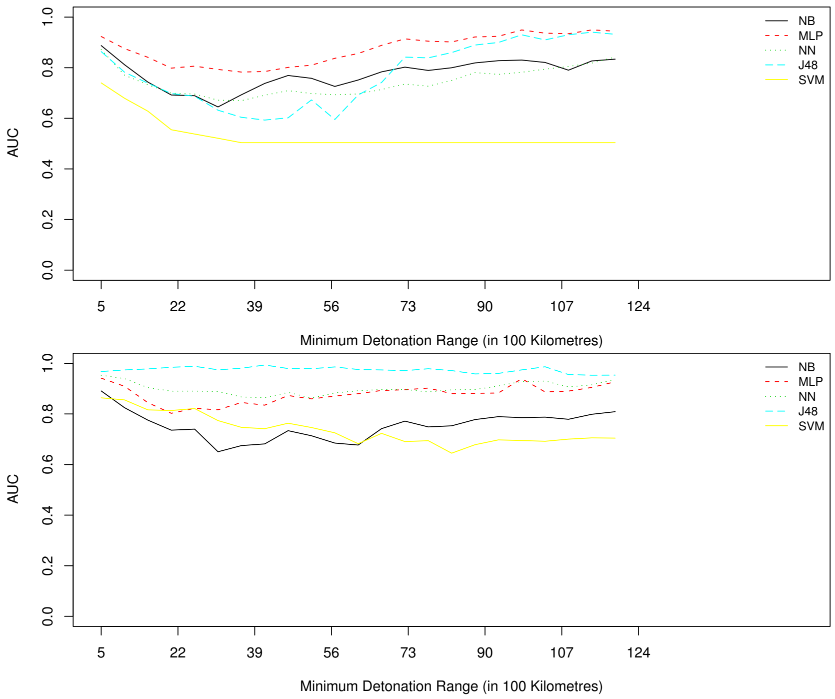
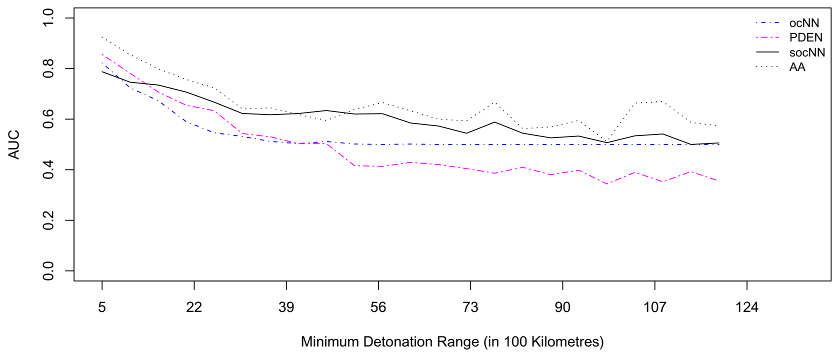


Fig. 7 This figure contrasts the performance of the binary classifiers, in terms of the AUC as a function of distance, on the standard feature-space (see the plot on the left), and when the feature-space is extended to include an assessment of the wind direction (see the plot on the right).

strong relationship between the wind direction and pollutant levels at the receptor, which suggests a possibly informative feature.

By expanding the standard CTBT feature space to include wind direction, we have produced a significant increase in the AUC. In particular, the top classifiers (MLP, AA, socNN), now demonstrate the ability to detect detonations that, when considered solely on the basis of the four radioxenon measurements, fit into the background distribution with a high probability. This fact is, indeed, depicted for many of the binary and one-class classifiers in Figure 7 and Figure 8.

In particular, while the depth to the hull in the performance of the MLP decreases only slightly, the J48's hull is entirely removed when the wind direction feature is added. Thus, the J48 classification ceases to be affected by the detonation distance when the new feature is included. In addition, its mean AUC is significantly improved.



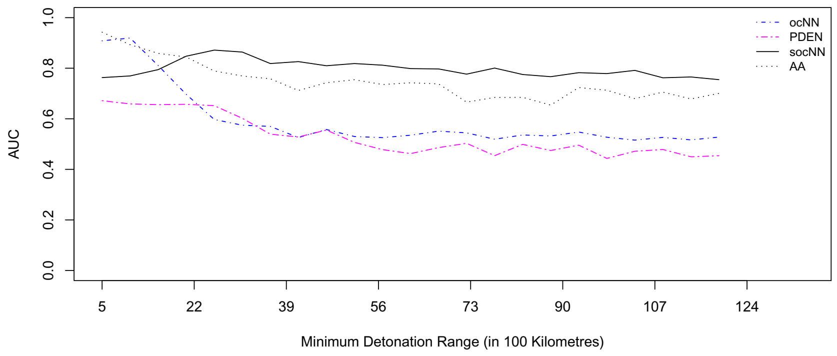


Fig. 8 This figure contrasts the performance of the one-class classifiers, in terms of the AUC as a function of distance, on the standard feature-space (see the plot on the left), and when the feature-space in extended to include an assessment of the wind direction (see the plot on the right).

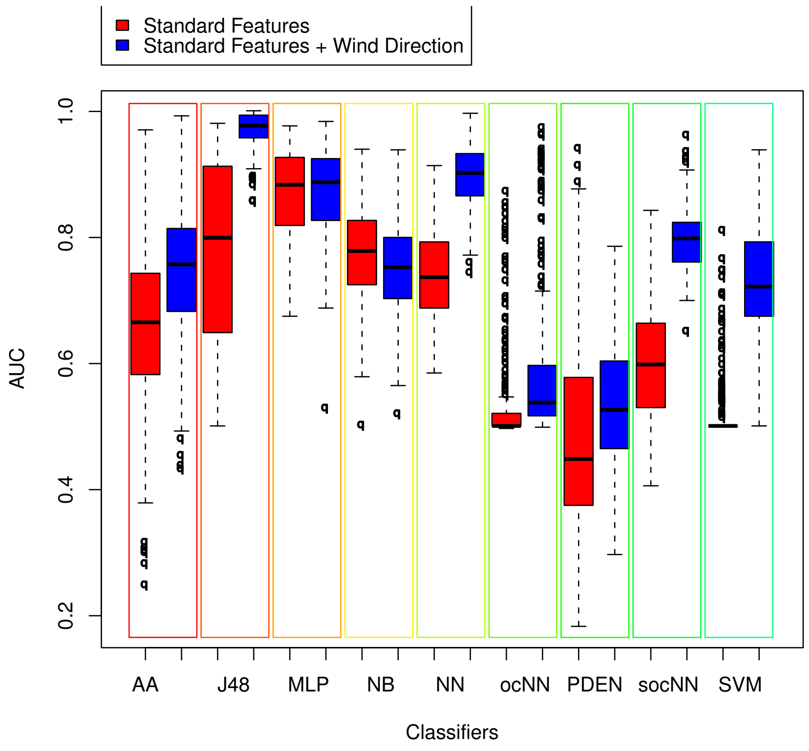


Fig. 9 This figure utilizes a series of boxplots to compare the performance of the nine classifiers and the standard feature-space, and with the extended feature-space, which is augmented by a wind direction indicator.

The NN and SVM classifiers also benefit from the inclusion of the wind direction feature. However, the new feature has a slightly negative effect on the NB. It has been noted in the literature, that many of the PR algorithms, including the MLP, SVM and NB may benefit from normalization of the features [10, 27]. Thus, it is conceivable that the performance of these classifier may be improved to some degree. However, these results provide a good baseline from which the individual classifiers can be compared.

By expanding the feature-space to include the wind direction, the OC learner, socNN, improves significantly, and becomes, in general, the top learner amongst its peers. The classifier, AA, also improves as a result of the new feature. However, its AUC scores do not increase to the same extent as the socNN.

Similar to the socNN, the PDEN's initial performance is lower in the newly expanded feature-space. However, the majority of its performance function is elevated. Finally, the ocNN benefits the least from the new feature, although, its initial performance is improved.

Thus, in the worst case, the wind direction feature produces marginal improvements in the performance of the four OC learners. However, it significantly improves both the AA and the socNN's ability to perform in scenarios where the detonations occur at distances equivalent to, and beyond the radial distance to the background source.

In Figure 9, a series of boxplots are utilized to facilitate the comparison of classifier performance in the two feature-spaces. Indeed, these results confirm the trends that we have previously identified. Particularly noteworthy is the depiction of J48's performance; this plot emphasizes both the significant increase in the J48's median AUC score, and the impressive stabilization of its classification results when the wind direction feature is added. The benefits to the SVM are also well visualized in this figure.

It is, indeed, well demonstrated in Figure 7, Figure 8, and Figure 9 that the additional information has assisted many of the classifiers to overcome the significant challenges inherent in identifying SE events within the field of background noise.

7.6 Results: Scenario 2

In this section, we present the results that were produced on the four assessment criteria that were motivated, and utilized in the previous sections. In this section, however, we explore the very intriguing classification scenario, which we previously denoted S2. This exploration follows the same structure that was previously applied in the exploration of the first classification scenario. Thus, we begin by examining the AUC scores produced by each of the one-class classifiers over the 23 detonation ranges; we then proceed to consider the performance over the two successive, smaller distances, the performance as a function of distance, and finally the benefit of expanding the feature-space to include an additional wind direction feature.

7.6.1 General Performance

In this section, we present a general overview of the performance of the set of one-class classifiers on the simulated CTBT domain. More specifically, we present an assessment of the four one-class classifiers, in terms of their AUC scores on the 230 datasets that covered the 23 detonation ranges.

Once again, in light of the fact that the SE event will, in practice, occur at random and unpredictable distances, these results are particularly insightful.

The results that are depicted in Figure 10 were compiled as a series of boxplots; one for each classifier. In addition, Table 3 contains a compilation of the mean, maximum, minimum and standard deviation of the each classifier's overall results.

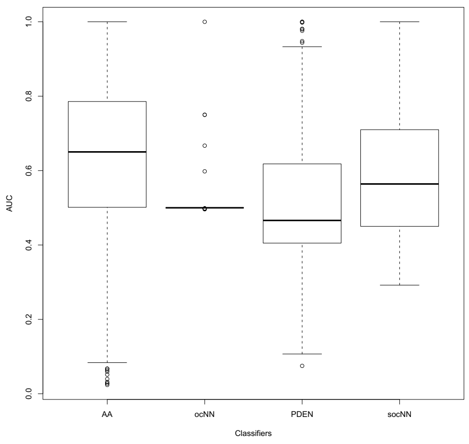


Fig. 10 This figure displays the performance of the four classifiers, in terms of their AUC scores on the 230 generated CTBT datasets, in the form of a series of boxplots.

Table 3 This table displays the general classification results, in terms of AUC.

	Mean	Max	Min	STDV
ocNN	0.505	1	0.496	0.042
PDEN	0.507	1	0.075	0.185
socNN	0.587	1	0.292	0.171
AA	0.621	1	0.024	0.225

Our assessments of both Figure 10 and Table 3 reveal that, similar to our findings on the S1 scenario, the AA classifier is superior, in terms of its mean, and median scores, to the other OC classifiers. Indeed, on this, which is a more challenging task, its mean and median values are only slightly lower than in the previous task. However, within this second scenario, it has the lowest minimum AUC scores, which appear as outliers in the boxplot. In addition, it is extremely unstable, with results ranging from perfect to near zero.

The socNN classifier ranks second after the AA according to its median and mean, and was considerably more stable, while the ocNN and PDEN classifiers produced values that were near or below 0.5.

7.6.2 Performance on Short- and Long-Range Detonations

In Figure 11, we present the results produced over two detonation ranges of particular interest. Specifically, the Boxplot on the left in the figure contains the results for the datasets that include detonations between the distances of

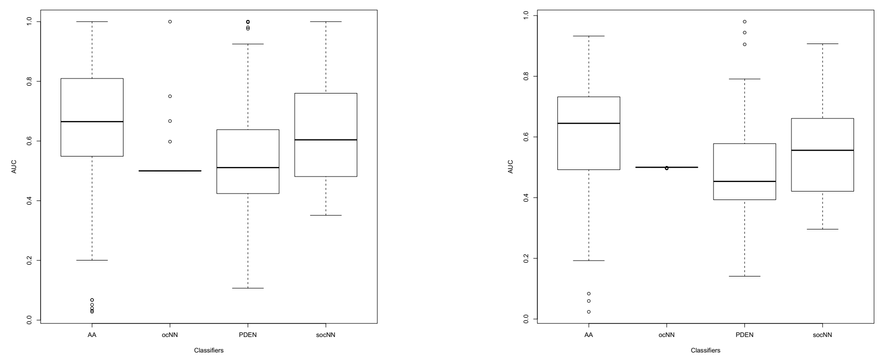


Fig. 11 In this figure, Boxplot on the left displays the performance of the four classifiers, in terms of their AUC scores for detonations occurring between the distances of 1,000 km and 5,500 km, and the Boxplot on the right displays their performances for detonations between the distances of 5,500 km and 10,000 km.

1,000 km and 5,500 km, while the Boxplot on the right has those with detonations between 5,500 km and 10,000 km. Together, these plots demonstrate, contrary to the previous results, that there is little change in performance at greater distances.

7.6.3 Performance as a Function of Distance

In this sub-section, we present classifier performance as a function of distance. As in the previous section, performance is assessed both according to the AUC and the FPR.

The AA and socNN are, once again, roughly identifiable as the best of the four classifiers in Figure 12 and Figure 13. However, all of the classifiers, with

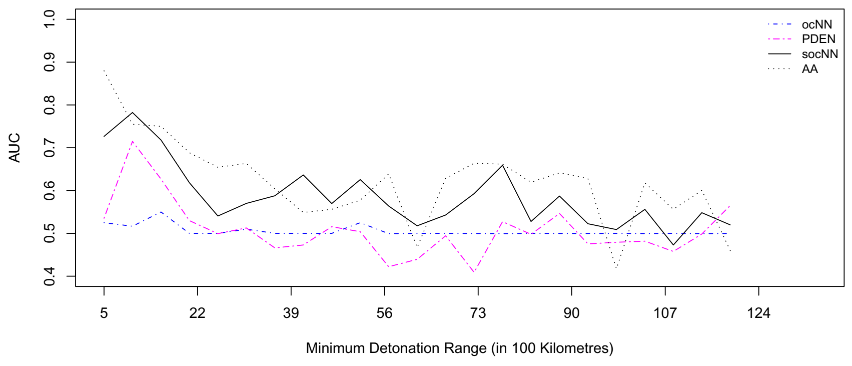


Fig. 12 This figure displays the performance of the four one-class classifiers as a function of distance, according to their AUC scores.

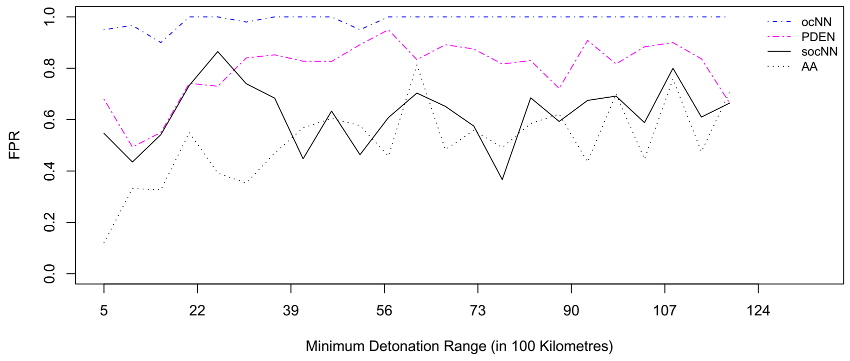
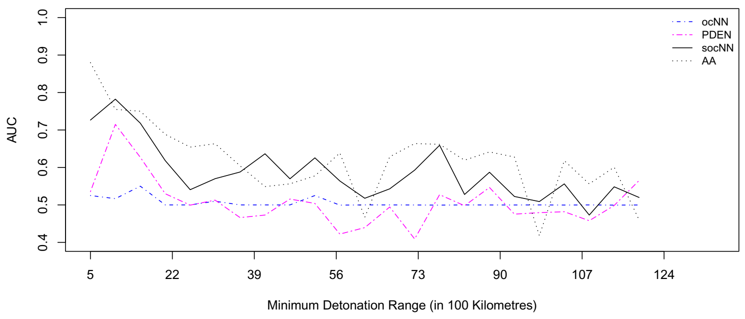


Fig. 13 This figure displays the performance of the four one-class classifiers as a function of distance, according to their FPR scores.



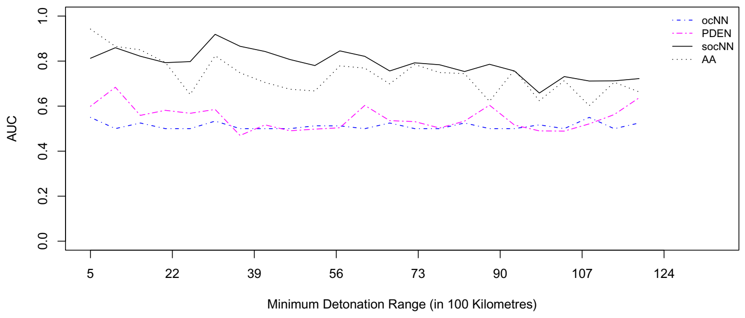


Fig. 14 This figure contrasts the performance of the one-class classifiers, in terms of the AUC as a function of distance, on the standard feature-space (see the plot on the left), and when the feature-space is extended to include an assessment of the wind direction (see the plot on the right).

the exception of ocNN, which rapidly converges to 0.5, suffer from significant and essentially random fluctuations. These fluctuations in performance suggest that the classifiers' results were as dependent on the nature of the SE events in the 230 datasets, as on the distance at which the events originally occurred.

7.6.4 Expanded Feature-Space

In this final section, we consider the benefits of extending the feature space to include a wind direction indicator. In Figure 14, both the original plot of the four classifiers' performances as a function of distance, and their performances on the extended feature-space are plotted. For an alternate view, the comparison is composed of a series of boxplots in Figure 15.

These figures illustrate that both the AA and the socNN significantly benefit from the expanded feature-space. Indeed, the socNN benefits the most, as it becomes superior to the AA for the vast majority of distances, and the variability in its results are significantly dampened.

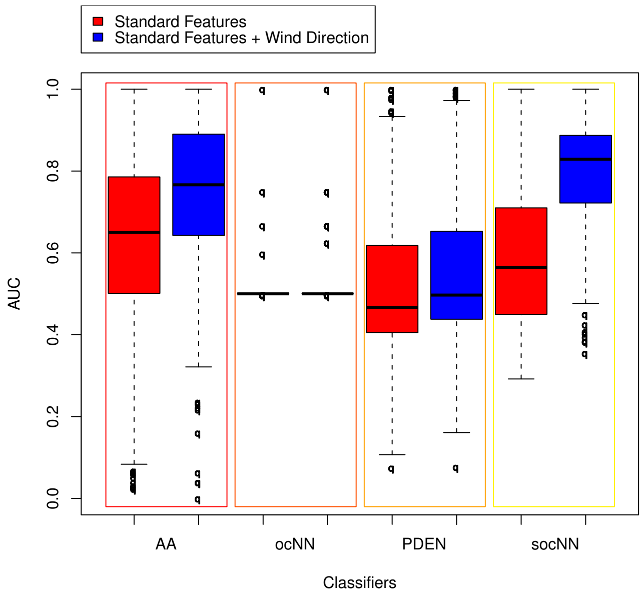


Fig. 15 This figure utilizes a series of boxplots to compare the performance of the four classifiers and the standard feature-space, and with the extended feature-space, which is augmented by a wind direction indicator.

7.7 Discussion

In this section, we consider the results previously reported for the OC classifiers in comparison to those reported for the binary learners. In particular, Section 7.7.1 compares the two classification strategies within the first scenario, namely S1. Alternatively, the OC classifiers are considered in comparison to the set of standard binary classifiers on scenario S2 in Section 7.7.2.

7.7.1 Results: S1

The relatively low mean and median AUC scores produced by the OC classifiers, combined with the considerable variability in their results on the standard CTBT feature-space, particularly in comparison with the top binary learners, clearly illustrate the many challenges inherent in applying OC learning to the derivation of a binary classifier. However, Hempstalk *et al.*, in [13], previously identified similar comparisons between binary and OC learners as “naïve” comparisons, when applied to scenarios that are accurately identifiable as OC problems.

In particular, in so-called OC problems, such as the detection of SE events, the second class is inherently ill-understood due to the fact that a characteristic set cannot be drawn from it. Thus, training and testing a binary learner as if one could draw a representative set from the second class, which is generally assumed when training a binary classifier, provides an upper bound on the classifier’s future performance.

The key differences in the performance of the two forms of classifiers is well illustrated in Figures 4 and 5. While the OC classifiers are very competitive on the initial radial ranges, when the detonation occurs further afield, their AUC scores drop considerably in comparison to all of the binary classifiers, with the exception of the SVM. The initial success of the OC classifiers suggests that they are very capable of associating anomalously high levels of radioxenon with the SE event class.

However, the binary learners are not only well adapted to classifying anomalously highly levels as members of the SE event class, through the binary learning process they are also capable of drawing on the anomalously low levels, which commonly result from detonations that occurred well beyond the radial distance to the background source, to specialize their decision boundaries such that similar events are recognized as belonging to the SE event class in the future.

The results of expanding the standard CTBT feature-space to include an indicator of the prevailing wind were, in general, very favourable, and lead to improved AUC scores for most of the classifiers, with the NB being the sole exception.

In its essence, the wind direction feature enabled the classifiers to learn the direction of the background source. As a result, the classifiers were able to identify detonations, which occurred at similar radial distances to the receptor site as the background emissions, and thus, had signatures that were similar to the background levels, but were transported from a different direction. This result is identified very clearly in Figure 7, and suggests that the further expansion of the feature-space might additionally improve performance.

7.7.2 Results: S2

A considerable portion of the previous analysis is applicable to this second, more challenging, classification scenario. Most importantly, the benefits of the extended feature-space were witnessed within S2 as well. However, due to the nature of the problem, only the OC classifiers were applied to this first attempt at performing PR within this new domain.

As a result of the formulation of the problem, we proposed the use of standard OC classifiers as unsupervised learners, and relied on inner mechanisms of the individual classifiers to facilitate the derivation of a model that segregated those instances of the training set that were accurately of the background class from the naïvely/erroneously labelled instances of the outlier class.

It is clear that the instability in performance that is depicted with respect to distance, and which is significantly more apparent in S2 than S1, results both from the erroneous instances in the training sets of S2, and the variability in classification challenges presented by the few members of the SE event class in the test sets. Indeed, the generation of random SE events over a domain as vast as the simulated CTBT domain, will inevitably produce both very easy, and nearly impossible classification tasks. Thus, when randomly including only a minute number of these events in the test sets, it is probable that performance on the SE event class will fluctuate significantly. This is, of course, why a large number of receptors are required in the global receptor network.

However, while the ensemble mean performance fluctuates considerably over the successive radial ranges, when considered in terms of the overall means, or medians, the performance of the OC classifiers on the S2 task is only slightly lower than on the S1 task. In addition, this is true if in Figures 5 and 12, we were to conduct our analysis according to a series of best-fit lines.

Finally, as is depicted in Figure 14, in addition to elevating the performance of the top classifiers, the inclusion of the wind direction in the feature-space significantly dampens the variability in their performance. Moreover, Saey, in an extensive study of background radioxenon concentrations in Europe and North America, found that a few outliers representing significant increases in the background concentrations can be expected [21]. These outliers are attributed to alternate background sources, and can be assumed to have arrived at the receptor site via short-lived, and anomalous alterations in meteorology. Based on the standard CTBT feature space, such events, undoubtedly, suggest the detonation of a nuclear weapon. However, provided a sufficient quantity of training data is available, it is conceivable that PR systems functioning with the wind direction feature may appropriately identify outliers of the background class.

7.8 Conclusions

This book chapter is a comprehensive overview of the work that is reported on the recognition of Stochastically Episodic events (like clandestine nuclear explosions), which was earlier reported as in [5] and [4]. In this research endeavor, we extended the frontiers of novelty detection through the introduction of a new field of problems open for analysis. In particular, we noted that this new realm deviates from the standard set of one-class problems based on the presence of three characteristics, which ultimately amplify the classification challenge. They involve the *temporal* nature of the appearance of the data, the fact that the data from the classes are “interwoven”, and that a labelling procedure is not merely impractical - it is almost, by definition, impossible.

To set the background, the paper first contained a brief overview of two-class and one-class classification methods. Thereafter, as a first attempt to tackle these problems, we presented two specialized classification strategies as demonstrated within the exemplary scenario intended for the verification of the CTBT. Here, we applied the simulation framework presented in [6], to generate CTBT inspired datasets, and demonstrated these classification strategies within the most challenging classification domain. More specifically, we have shown that OC classifiers can be successfully applied to classify SE events, which are unknown, although present, at the time of training.

Finally, we have added a weighting parameter to the OC nearest neighbour algorithm, thereby significantly increasing its performance on our experimental domain. We have also demonstrated that the expansion of the CTBT feature space significantly improves classifier performance on our simulated data, thus, motivating further exploration of the expansion of the standard CTBT feature space to include meteorological measurements [5], [4]. This result was further verified in [3] based on a complex set of new simulation scenarios.

References

1. Aha, D.W.: Generalizing from case studies: A case study. In: Proceedings of the Ninth International Conference on Machine Learning, pp. 1–10 (1992)
2. Bellinger, C.: Modelling and classifying stochastically episodic events. Master's thesis, Carleton University, Ottawa, Ontario (2010)
3. Bellinger, C., Japkowicz, N.: Motivating the inclusion of meteorological indicators in the ctbt feature-space. In: Proceedings of IEEE Symposium on Computational Intelligence for Security and Defense Applications, Paris, France (April 2011)
4. Bellinger, C., Oommen, B.J.: A new frontier in novelty detection: Pattern recognition of stochastically episodic events. In: Asian Conference on Intelligent Information and Database Systems (2011)
5. Bellinger, C., Oommen, B.J.: On the pattern recognition and classification of stochastically episodic events. In: Transactions on Computational Collective Intelligence (2011) (accepted for Publication)
6. Bellinger, C., Oommen, B.J.: On simulating episodic events against a background of noise-like non-episodic events. In: Proceedings of 42nd Summer Computer Simulation Conference, SCSC 2010, Ottawa, Canada (July 2010)
7. Chapelle, O., Schölkopf, B., Zien, A.: Semi-supervised Learning. MIT Press (2006)
8. Datta, P.: Characteristic concept representations. PhD thesis, University of California at Irvine, Irvine, CA, USA (1997)
9. Dietterich, T.G., Lathrop, R.H., Lozano-Pérez, T.: Solving the multiple instance problem with axis-parallel rectangles. Artificial Intelligence 89(1-2), 31–71 (1997)
10. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley, New York (2001)
11. Ghosh, A.K., Schwartzbard, A., Schatz, M.: Learning program behavior profiles for intrusion detection. In: Proceedings of the Workshop on Intrusion Detection and Network Monitoring, vol. 1, pp. 51–62 (April 1999)
12. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA data mining software: An update. ACM SIGKDD Explorations Newsletter 11(1), 10–18 (2009)
13. Hempstalk, K., Frank, E., Witten, I.H.: One-Class Classification by Combining Density and Class Probability Estimation. In: Daelemans, W., Goethals, B., Morik, K. (eds.) ECML PKDD 2008, Part I. LNCS (LNAI), vol. 5211, pp. 505–519. Springer, Heidelberg (2008), doi:10.1007/978-3-540-87479-9
14. Japkowicz, N.: Concept-Learning in the Absence of Counter-Examples: An Autoassociation-Based Approach to Classication. PhD thesis, Rutgers University (1999)
15. Japkowicz, N., Shah, N.: Evaluating Learning Algorithms: A Classication Perspective. Cambridge University Press (2011)
16. Kubat, M., Holte, R.C., Matwin, S.: Machine learning for the detection of oil spills in satellite radarimages. Machine learning 30(2), 195–215 (1998)
17. Kubat, M., Matwin, S.: Addressing the curse of imbalanced training sets: One-sided selection. In: Proceedings of the Fourteenth International Conference on Machine Learning, pp. 179–186. Morgan Kaufmann (1997)
18. Mitchell, T.M.: Machine learning. McGraw-Hill (1997)

19. Nairac, A., Townsend, N., Carr, R., King, S., Cowley, P., Tarassenko, L.: A system for the analysis of jet engine vibration data. Integrated Computer-Aided Engineering 6(1), 53–66 (1999)
20. Platzer, E.S., Nägele, J., Wehking, K.H., Denzler, J.: HMM-based defect localization in wire ropes–A new approach to unusual subsequence recognition. Pattern Recognition, 442–451 (2009)
21. Saey, P.R.J.: The influence of radiopharmaceutical isotope production on the global radioxenon background. Journal of Environmental Radioactivity 100(5), 396–406 (2009)
22. Saey, P.R.J., Bowyer, T.W., Ringbom, A.: Isotopic noble gas signatures released from medical isotope production facilities – Simulation and measurements. Applied Radiation and Isotpes (2010)
23. Schölkopf, B., Williamson, R.C., Smola, A.J., Shawe-Taylor, J., Platt, J.: Support vector method for novelty detection. In: Advances in Neural Information Processing Systems, vol. 12, pp. 582–588 (2000)
24. Stocki, T.J., Japkowicz, N., Li, G., Ungar, R.K., Hoffman, I., Yi, J.: In: Summary of the Data mining Contest for the IEEE International Conference on Data Mining, Pisa, Italy (2008)
25. Tarassenko, L., Hayton, P., Cerneaz, N., Brady, M.: Novelty detection for the identification of masses in mammograms. In: IEE Conference Publications, (CP 409), pp. 442–447 (1995)
26. Tax, D.M.J.: One-class classification; Concept-learning in the absence of counter-examples. PhD thesis, Technische Universiteit Delft, Netherlands (2001)
27. Witten, I.H., Frank, E.: Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann Publishers (2005)
28. Xu, R., Wunsch, D.C.: Clustering. Wiley-IEEE Press (2009)

Chapter 8
Learning of Defaults by Agents in a Distributed Multi-Agent System Environment

Henryk Rybinski, Dominik Ryżko, and Przemysław Więch

Abstract. The paper introduces a novel approach to machine learning in a multi-agents system. A distributed version of Inductive Logic Programming is used, which allows agents to construct new rules based on knowledge and examples, which are available to different memebrs of the system. The learning process is performed in two phases – first locally by each agent and then on the global level while reasoning.

8.1 Introduction

In today's world of many electronic devices equipped with sensors and computing power, knowledge is no longer available in one place only. Such entities must possess an efficient way of cooperating in order to reach solutions to the problems presented to them. Multi-agent systems (in the sequel MAS) bring tools for modeling such situations using the concept of an autonomous, intelligent and proactive agent.

One of the most important issues concerning MAS is an ability of agents to acquire knowledge in a distributed environment. In our previous work [18] two kinds of knowledge where identified, namely environmental knowledge, and domain knowledge. In particular, the environmental knowledge refers to the agent's communication experience, providing the agents with the answer for "what can be found where". So, the environmental knowledge is used by the agents to guide their search in a fragmented domain knowledge. In [18] means for learning environmental knowledge were considered. In this paper we focus on learning domain knowledge.

In [18], a formalism of the Distributed Default Logic (DDL) has been introduced in order to address the problem of finding knowledge in a distributed

Henryk Rybinski · Dominik Ryżko · Przemysław Więch
Institute of Computer Science, Warsaw University of Technology
e-mail: {hrb,d.ryzko,pwiech}@ii.pw.edu.pl

multi-agent system. To answer queries from users and other parties, agents perform distributed default reasoning. Agents should share their knowledge not only when there is a task to be solved, but they must act proactively. This may include different ways of knowledge indexing, or search for new sources of information. These preparations allow agents to act efficiently when new requests arrive.

We assume that each agent possesses a database of "observations", which serve as training examples. The origin of such a database is not important. Data can be obtained, for example, from sensor readings classified by an expert. An important issue with the agents is how to acquire new knowledge from the observations database. In particular, they can use observed examples to generate new default rules. Building new defaults allows agents to improve their performance and solve problems, which they could not handle before.

Some algorithms for default rule generation have been studied before in [5], [19], [9], [3]. However, these papers deal with centralized algorithms. The main novelty of our approach is that in the framework of MAS the agents learn defaults in a distributed environment. Such a distributed environment creates several problems for learning, which do not exist in the centralized approach. For example, one agent can learn a rule from its local database, but exceptions to this rule can be known only to another agent, or, in a worst case, several agents can learn different exceptions to a particular rule. Hence, the agents should be able to exchange and verify local knowledge among themselves. It may refer not only to basic facts from the local databases, but they can also exchange their rules, and based on them, generate other rules [21].

To solve these problems, additional steps in the learning process are needed, which will allow agents to construct a usable Distributed Default Theory, which can further be used for efficient distributed reasoning process. We show how agents exchange knowledge in the system and perform learning to reach these goals.

8.2 Related Work

Different approaches to construction of defeasible rules, including default rules, have been proposed. Benferhat et al. use big-stepped probabilities to extract new rules from binary tables [3]. Compared to other methods (e.g. [6]) it allows automatic detection of properties which can be discovered.

Another approach is taken in [8], where defeasible rules are discovered by means of data mining algorithms. The authors study different metrics, which can reduce the large number of candidate rules. In general, the problem of finding desired (minimal) theories is computationally expensive, so heuristics should be used in order to find good results in a short time ([10]).

Inductive Logic Programming (ILP) is a machine learning method, in which background knowledge, examples and hypotheses are all reperesented by means of logic programming. This approach has been studied in detail in several scientific papers (see e.g. [15] [4]). Especially works by Muggleton and de Readt are now considered classical in this field ([13] [14]). In [6] an application of Inductive Logic Programming to learning default theories is proposed. In the approach, the definitions of positive (p) and negative ($\neg p$) predicates are learned simultanously.

Example 1. With a given set of positive examples
$E^{+}_{flies} = \{3, 4, 5, 10, 11\}$

a set of negative examples
$E^{-}_{flies} = \{1, 2, 6, 7, 8, 9\}$

and background knowledge

$$\begin{aligned} B = \{\; & penguin(1), penguin(2), bird(3), bird(4), bird(5), mammal(6), \\ & mammal(7), mammal(8), mammal(9), bat(10), superpenguin(11), \\ & penguin(X) \rightarrow bird(X), \\ & superpenguin(X) \rightarrow penguin(X), \\ & bat(X) \rightarrow mammal(X)\} \end{aligned}$$

we can induce the following set of rules:

$$\begin{aligned} D = \Bigg\{ & \frac{bird(X) : flies(X), \neg penguin(X)}{flies(X)}, \\ & \frac{penguin(X) : \neg flies(X), \neg superpenguin(X)}{\neg flies(X)}, \\ & \frac{superpenguin(X) : flies(X)}{flies(X)}, \\ & \frac{bat(X) : flies(X)}{flies(X)}, \\ & \frac{mammal(X) : \neg flies(X), \neg bat(X)}{\neg flies(X)} \Bigg\}. \end{aligned}$$

This approach is sufficient for the case when we have all the knowledge in one location. However, if some information is missing, a single agent can no longer be able to learn the same concepts. For example, if an agent can see only positive cases, it will learn that all birds can fly.

$$bird(X) \rightarrow flies(X)$$

Even if it knows that $penguin(X) \rightarrow bird(X)$ but has no knowledge about penguins, the induced rule will be the same. This might seem irrelevant if we learn the negation of the predicate $flies$ at the same time. However, such a rule indicates that all birds can fly, so it is enough to classify any new instance to be a bird without checking the species to decide its flying abilities.

While dealing with learning in a multi-agent environment the important question is what is the relation between learning processes of individual agents and the knowledge of the whole multi-agent system. In [20], three types of multi-agent learning can be distinguished:

– Multiplied Learning: each agent learns independently of the other agents. While there may be interactions concerning the exchange of training data or outputs, no interference in the learning process itself takes place.
– Divided Learning: the learning task is distributed amongst a team of agents. The division takes place on a functional level, i.e., agents take different roles in a team and learn them separately.
– Interactive Learning: agents interact during learning and cooperate in generating a hypothesis beyond the pure exchange of data. In [20] it is described as a "cooperated, negotiated search for the solution of a learning task"

In general, this paper falls into the category of Logical-Based Learning [1].

8.3 Motivation

The motivation of our work is to introduce ways for learning of complete theories by a group of agents observing a limited part of the world, and possessing only parts of the whole knowledge. There are two main approaches to this problem. First, agents can exchange information about the rules they have discovered, and incorporate all exception that are missing in their local knowledge. This creates an initial overhead, but later the system is ready for complete reasoning and query answering. In the second approach agents perform learning on the fly, while answering queries. We will concentrate on the latter case, since in a dynamic MAS environment, with agents entering and leaving the system, we have to assume there will be no time to exchange all the necessary knowledge before other tasks have to be performed.

Let us consider the following example to illustrate a scenario, where different sources possess complementary information. We show the need to exchange information among agents in order to reach a correct answer to a query.

Example 2. Let us consider a multi-agent system consisting of three agents with partial knowledge. In the sequel, we denote by E_p^+ and E_p^- the sets of positive and negative examples of predicate p, respectively. We denote by B_A the background knowledge of an agent A, and presume that the agents A_1, A_2, A_3 have the following knowledge bases:

[1] Opposite to this approach is Reactive Learning, which does not use logic for knowledge reperesentation [1]. The implication is that we cannot fully comprehend the MAS activities.

$$\begin{aligned}
\mathbf{A_1} : E^{+}_{flies} &= \{3, 4, 5\} \\
E^{-}_{flies} &= \{8, 9\} \\
B_{A_1} &= \{brid(3), bird(4), bird(5), \\
&\quad mammal(8), mammal(9) \\
&\quad penguin(X) \rightarrow bird(X) \\
&\quad bat(X) \rightarrow mammal(X)\} \\
\mathbf{A_2} : E^{+}_{flies} &= \{\} \\
E^{-}_{flies} &= \{1, 2\} \\
B_{A_2} &= \{penguin(1), penguin(2) \\
&\quad superpenguin(X) \rightarrow penguin(X)\} \\
\mathbf{A_3} : E^{+}_{flies} &= \{10, 11\} \\
E^{-}_{flies} &= \{\} \\
B_{A_3} &= \{bat(10), superpenguin(11)\}
\end{aligned}$$

Based on this knowledge, the following additional rules can be learned by the agents locally with the use of inductive logic programming techniques:

$$\begin{aligned}
&A_1 : \{bird(X) \rightarrow flies(X), mammal(X) \rightarrow \neg flies(X)\} \\
&A_2 : \{penguin(X) \rightarrow \neg flies(X)\} \\
&A_3 : \{superpenguin(X) \rightarrow flies(X), bat(X) \rightarrow flies(X)\}
\end{aligned}$$

In such a setup, if another agent, say A_4, asks the agents if *tom* the superpenguin can fly, it will receive three answers:

1. The agent A_3 has a direct example of superpenguin, so it will answer that *tom* can fly.
2. A_2 knows that *tom* is a kind of penguin and penguins do not fly, so it gives a negative answer.
3. In DDL (as described in Section 8.4) agents exchange knowledge according to the established communication language. Therefore, A_1 learns from A_2, that $penguin(tom)$ has been deduced, and further on, reasons that $bird(tom)$ is true.

Using this procedure, agent A_4 will receive two positive answers that *tom* can fly and one negative answer. In order to decide, which answer is correct, A_4 has to know the rationale for each answer. The most reasonable criteria is generality/specificity of information. If an example directly matching the query is available it should be used in the first place, rather than a more general rule. If generality/specificity cannot be decided, some other measures have to be applied. For the DDL formalism, the reliability of agents can be used. In the example above the rationale provided by A_3 is the most specific, since it is an example, which directly matches the case to be solved. Agent A_3 can provide a concrete example of a superpenguin, which can fly. The information provided by both A_2 and A_1 is more general. These agents can only use rules, which state that penguins cannot fly and birds typically can fly.

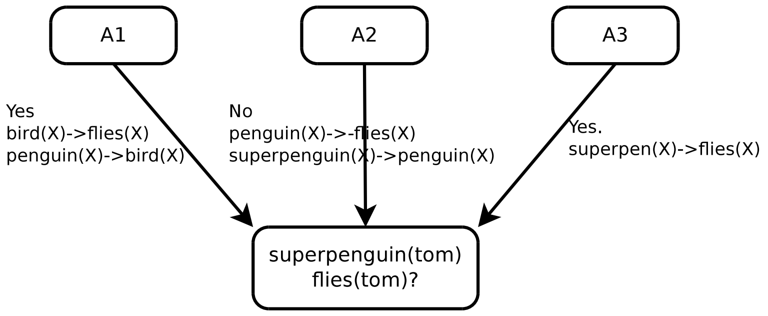


Fig. 1 Query

It is not always possible to decide the level of generality/specificity of answers from different agents. In the following example, it is impossible to decide which rule should be treated as more specific.

Example 3. Let us consider a multi-agent system consisting of two agents with positive and negative examples (E^+_{flies} and E^-_{flies}) of animals, which can or cannot fly.

$$\begin{aligned}\mathbf{A_1} : E^+_{flies} &= \{\}\\ E^-_{flies} &= \{1,2\}\\ B_{A_1} &= \{penguin(1), ostrich(2),\\ &\quad penguin(X) \rightarrow bird(X)\\ &\quad ostrich(X) \rightarrow bird(X)\}\end{aligned}$$

$$\begin{aligned}\mathbf{A_2} : E^+_{flies} &= \{3,4\}\\ E^-_{flies} &= \{\}\\ B_{A_2} &= \{bird(3), bird(4)\}\end{aligned}$$

Based on the knowledge of these agents the following rules can be created:

$A_1 : \{bird(X) \rightarrow flies(X)\}$
$A_2 : \{bird(X) \rightarrow \neg flies(X)\}$

Since the two rules are on the same level of generality/specificity, a query from another agent A_3 asking whether some bird can fly will produce two contradictory answers. Moreover, it can be stated that both of them are on the same level of generality/specificity, as both answers refer to the predicate *bird*.

This is a case when agent A_3 can either accept that two default theory extensions will be generated or it should decide which of the answers should be treated with higher priority. Since default rules specify *typical* situations, one of the measures, which could be used to assess the usefulness of such a rule, is its global support (see Section 8.4 for definition). Thus, when the number of flying birds exceeds the number of non-flying birds, the default rule that *typically, birds fly* will be chosen.

Regardless of how A_3 treats the answers, it can communicate back to A_1 and A_2 that they should revise their knowledge bases based on the information from the other agents. This way, agent A_2 could update its rule to include exceptions as follows:

$$\frac{bird(X) : flies(X) \wedge \neg penguin(X) \wedge \neg ostrich(X)}{flies(X)}$$

In Example 3 the birds were divided into species, which have or do not have the ability to fly. The next example shows a slightly different situation, as the learned feature does not refer to a single taxonomy. It leads to a new situation, where it is difficult to decide, which rule should have a higher priority.

Example 4. Let us consider a multi-agent system consisting of two agents with positive and negative examples (E^+_{flies} and E^-_{flies}) of animals, which can or cannot fly.

$$\begin{aligned} \mathbf{A_1} : E^+_{flies} &= \{1\} \\ E^-_{flies} &= \{\} \\ B_{A_1} &= \{stork(1), stork(X) \rightarrow bird(X)\} \end{aligned}$$

$$\begin{aligned} \mathbf{A_2} : E^+_{flies} &= \{\} \\ E^-_{flies} &= \{2\} \\ B_{A_2} &= \{brokenwing(2)\} \end{aligned}$$

The following rules can be created based on the knowledge of these agents:

$$\begin{aligned} &A_1 : \{stork(X) \rightarrow flies(X)\} \\ &A_2 : \{brokenwing(X) \rightarrow \neg flies(X)\} \end{aligned}$$

Let us consider a scenario, where agent A_3 would like to ask whether a stork with a broken wing flies ($stork(sam)$, $brokenwing(sam)$, $flies(sam)$?). The aswers from agents A_1 and A_2 will have unrelated rule predecessors. This makes it impossible to compare the generality/specificity of these answers. Hence, the choice between the rules generated by A_1 and A_2 cannot be based on the support measure. Indeed, although the rule from A_1 has a smaller support than the rule from A_2, we cannot discard the rule with smaller support. One of the solutions would be to use the confidence function for the agents [18]. Let us note however, that if the agents learn new facts

$$\begin{aligned}\mathbf{A_1}: E^{+}_{flies} &= \{1,3\}\\ E^{-}_{flies} &= \{\}\\ B_{A_1} &= \{stork(1), hawk(3),\\ &\quad stork(X) \rightarrow bird(X), hawk(X) \rightarrow bird(X)\}\end{aligned}$$

$$\begin{aligned}\mathbf{A_2}: E^{+}_{flies} &= \{\}\\ E^{-}_{flies} &= \{2\}\\ B_{A_2} &= \{brokenwing(2), bird(2)\}\end{aligned}$$

the induced rules will be

$A_1 : \{bird(X) \rightarrow flies(X)\}$
$A_2 : \{brokenwing(X) \wedge bird(X) \rightarrow \neg flies(X)\}$

and we can decide on the generality/specificity.

The examples above, show informally how the information exchange between the agents can enrich the global knowledge of MAS. It shows also that one of the essential factors is support of knowledge of particular agents, although in some cases it may turn out to be insufficient. After introducing basic concepts (in Section 4), we present our solution in Section 5.

8.4 Preliminaries

This section introduces the main formalisms used in the remainder of the paper i.e. Default Logic (DL), Distributed Default Logic (DDL) and Inductive Logic Programming (ILP).

8.4.1 Default Logic

Default logic is a non-monotinic logic introduced by Reiter [17]. It allows to reason in the case of incomplete knowledge by making assumptions and retracking the process in case of finding exceptions later on. A default theory Δ is described as a pair (D, W), where D represents a set of default rules while W is a set first-order formulas.

Definition 1. A **default** is in the form:

$$r = \frac{\alpha : \beta}{\gamma}$$

where α, β and γ are well-formed formulae. α is the **prerequisite**, β is the **justification** and γ is the **consequent**. We will denote by p(r) the prerequisite of rule r, by j(r) its justification and by c(r) its consequent. □

Such a rule is interpreted as: *If α is known, and β can be assumed (there is no evidence against it), then we can conclude γ.*

Definition 2. Let E be a set of closed formulae, and $\langle D, W\rangle$ be a closed default theory.

Let

$$E_0 = W$$

$$E_{i+1} = E_i \cup \left\{ \gamma \mid \frac{\alpha : \beta}{\gamma} \in D, \alpha \in Th(E_i) \text{ } and \text{ } \beta \notin Th(E) \right\}$$

$Th(E)$ is an **extension** of $\langle D, W\rangle$ iff

$$Th(E) = \bigcup_{i=0}^{\infty} Th(E_i)$$

□

A given default theory can have no, one or many extensions. There are several approaches to managining of multiple extensions [2], [16], [19], [12]. The Distributed Default Logic presented below is an example of a prioritized default theory.

8.4.2 Distributed Default Logic

Distributed Default Logic is a formalism introduced in order to allow efficient defeasible reasoning in a multi-agent system [18]. Agents store knowledge in the form of Distributed Default Rules, which include also environental knowledge in the form of links to other agents.

Definition 3. In a multi agent system $MAS = \{A_1, ..., A_n\}$ a **distributed default template** is in the form:

$$\frac{\alpha : \beta_1^{L_1}, ..., \beta_k^{L_k}}{\gamma}$$

where $L_k \subseteq MAS$ and α, β and γ have the same meaning as in standard defaults. □

The meaning of such a template is that agent can build a distributed default rule out of it by selecting one single agent from each list. The process of builing rule from a template is called materialization. Definitions 4 and 5 define the concepts in more formal way.

Definition 4. A **distributed default rule** is in the form:

$$\frac{\alpha : \beta_1^{A_1}, ..., \beta_k^{A_k}}{\gamma}$$

where $A_i \in MAS$ and α, β and γ have the same meaning as in standard default logic. □

Definition 5. We say that a distributed default rule r is a materialization of template T iff for every component $\beta_i^{L_i}$ in the template T we have $A_i \in L_i$.

Let Δ_{MAS} be a Distributed Default Theory divided into n partitions

$$\Delta_{MAS} = \{\Delta_1 ..., \Delta_n\}$$

$\{\Delta_i\}_{i<=n}$ will be called a partitioning of default theory Δ. By $S(\Delta_i)$ we will denote the signature of the partition (the set of non-logical symbols). Each of the partitions will contain a portion of global default theory

$$D_{MAS} = \{D_1, ..., D_n\}$$

$$W_{MAS} = \{W_1, ..., W_n\}$$

where D_i is a set of defaults of i-th partition and W_i is a set of first-order formulas of this partition.

For a given partitioning, a labeled and directed graph $G = (V, E, L)$ can be generated, which we will call the intersection graph. In this graph, each node corresponds to the individual partition Δ_i, so $V = 1..n$. Two nodes i, j are linked by the oriented edge leading from i to j iff

$$\exists d_i \in D_i, d_j \in D_j, a \in S(\Delta_{MAS}) : a \in c(d_i) \wedge a \in p(d_j)$$

or

$$\exists d_j \in D_j, a \in S(\Delta_{MAS}) : a \in W_i \wedge a \in p(d_j)$$

The edges are labeled with the set of symbols that the associated partitions Δ_i and Δ_j share ($L(i, j) = S(\Delta_i) \cap S(\Delta_j)$). $L(i, j) \subseteq L$ will be called communication language between partitions Δ_i and Δ_j.

In Distributed Default Logic the reasoning proces is driven by the intersection graph defined above. References to other agents in templates together with exchange of information about common predicate symbols allow agents to perform distributed reasoning when necessary. A detailed description of this process can be found in [18].

8.4.3 Inductive Logic Programming

The classical definition of ILP is the following. Given the background knowledge B a set of positive examples and a set of negative examples E^+ and E^-, we need to find a hypothesis H such that $\forall e \in E^+, B \wedge H \models p(e)$ and $\forall e \in E^-, B \wedge H \not\models p(e)$. In order to simplify our considerations, we assume that different agents observe disjoined sets of training examples.

In a typical centralized setup ILP uses the covering algorithm, which learns rules using a generalization procedure, that performs a search through an ordered space of all possible rules. After one rule is selected, all covered positive examples are removed from the training set and the next rule is learned from the remaining examples. The whole process is stopped when no more positive examples are left in the training set.

Algorithm 1: Covering

```
begin
    $Rules = \emptyset$
    while $E^+ \neq \emptyset$ do
        $R = learnrule(E^+, E^-, B)$
        $Rules = Rules \cup R$
        $B = B \cup R$
        $E^+ = E^+ - \{ExamplesCoveredbyR\}$
    return Rules
end
```

The main problem here is scalability of such approach. Selecting a rule which covers most examples is very computationally expensive and has to be performed in each step of the algorithm. Different approaches to the problem of efficient rule selection in ILP can be found in literature. Four major approaches can be identified here [7]:

1. new algorithms
2. reducing the number of hypotheses
3. efficient testing of candidate hypotheses
4. parallelization

The last class can be further divided into:

- parallel exploration of independent hypotheses
- parallel exploration of search space
- parallel execution over paritioned data

It has been shown that, although the last case is only locally consistent and limits possibilities of learning recursive rules, it gives good results for shared memory architectures. An important property of the method based on data partitioning mentioned above is that we explicitly divide the knowledge in order to obtain better performance. In this paper we analyse a situation, in which we are faced with a given knowledge partitioning on which we do not have influence. Therefore, the main problem is how to link knowledge in order to obtain results comparable to the centralized approach.

For the induced rules we will define support measures:

Definition 6. The **local support** for agent A_n of a rule induced with distributed ILP is the number of positive training examples covered by the rule, known by the agent performing the ILP algorithm.

$$Sup_{A_n}(rule) = \left|\{e \in E^+ : \text{rule covers } e\}\right| \qquad \square$$

Definition 7. The **global support** of a rule induced with distributed ILP is the number of positive training examples covered by the rule, known by all the agents performing the ILP algorithm.

$$Sup(rule) = \sum_{i=1}^{n} Sup_{A_i}(rule) \qquad \square$$

8.5 Learning DDL Theory by MAS

In a centralized approach to machine learning ILP is considered an eager method [11]. However, in MAS the situation is more complex. Agents have two options for choosing the moments of learning. The first option consists in sharing new knowledge immediately, when it is generated from new training examples registered, whereas the second one consists in integrating learning with reasoning. We call them respectively *eager* and *lazy*.

Eager approach

If a new example causes changes in the previously induced rules, it possibly has impact on other agents' knowledge and should be reported. However, in a typical MAS it would be inefficient to exchange information each time something new is known. Therefore, we can intergrate learning with reasoning, so the whole process consists of two phases.

Lazy approach

Firstly, an agent asks other agents for some knowledge to perform a reasoning process. Secondly, it decides which agents can benefit from the knowledge posessed by others, and suggests corresponding exchange of knowledge to be performed.

In any of the approaches described above, the exchange of knowledge will lead to the transformation of the default theory. We define the following transformations:

1. $\{a(X) \to b(X), \exists Y \in E_b^- : a(Y)\} \mid\!\sim \dfrac{a(X) : b(X)}{b(X)}$
2. if $c(X) \to a(X)$ belongs to domain knowledge B then $\{a(X) \to b(X), c(X) \to \neg b(X)\} \mid\!\sim \dfrac{a(X) : b(X) \wedge \neg c(X)}{b(X)}$
3. $\{c_1(X) \to \neg b,$
   ...,
   $c_n(X) \to \neg b,$
   $e_1(X) \to b,$
   ...,
   $e_m(X) \to b,$

$$\left.\begin{array}{l}\dfrac{a(X) : b(X) \wedge \neg c_1(X) \wedge ... \wedge \neg c_n(X)}{b(X)}\end{array}\right\} \mid\!\sim$$
$$\dfrac{a(X) : \neg b(X) \wedge \neg e_1(X) \wedge ... \wedge e_m(X)}{\neg b(X)}$$

For the transformations above, we presume that observed examples cannot be invalidated in time. In the sequel we call the left side of the transofrmation $\mid\!\sim$ as premises, and the right side - conclusion.

The first transformation provides a mechanism for changing an implication into a default rule in case of observed negative examples. An agent can change the implication to a default rule, when it observes exceptions by itself or is taught about them by other agents. The transformation can be applied without knowing the details about the negative facts.

The second transformation describes the situation, when an exception to a strict rule is detected. As a result, the rule is transformed into a default rule with exception.

Within the set of premises of the third transformation we distinguish two subsets of induced rules, namely the ones supporting the learned concept, and the ones supporting its negation (i.e. $\{c_i(X) \rightarrow \neg b(X)\}_i$ and $\{e_j(X) \rightarrow b(X)\}_j$). The third transformation is used, when the following condition is satisfied:

$$\sum_{i=1}^{n} Sup(c_i(X) \rightarrow \neg b(X)) - \sum_{j=1}^{m} Sup(e_j(X) \rightarrow b(X)) > \epsilon$$

where $\epsilon > 0$ is a threshold for applying the transformation.

Example 5. Once again, let us consider the scenario from Example 3. If agent A_1 did not know the species of the non-flying birds, agent A_2 should still reconsider its database, so that it allows exceptions. It can use the first of the transformations described above to conclude that 'typically birds fly':

$$A_2 : \left\{bird(X) \rightarrow flies(X), tom \in E^-_{flies}, bird(tom)\right\} \mid\!\sim \frac{bird(X) : flies(X)}{flies(X)}$$

Example 6. Let us refer back to Example 2. In order to improve the system efficiency, after the answer to the query is decided, agent A_4 should facilitate the exchange of information leading to the exchange of knowledge. This is a delayed learning step in the system. It is easy to detect that the example provided by A_3 is an exception to the knowledge possessed by A_2. By analogy, A_2 holds examples which are exceptions to the knowledge of A_1. After the exceptions are exchanged, the system learns more precise rules:

$$A_1 : \{bird(X) \to flies(X), penguin(X) \to \neg flies(X)\} \mid\!\sim$$

$$\frac{bird(X) : flies(X), \neg penguin(X)}{flies(X)}$$

$$A_2 : \{penguin(X) \to \neg flies(X), superpenguin(X) \to flies(X)\} \mid\!\sim$$

$$\frac{penguin(X) : \neg flies(X), \neg superpenguin(X)}{\neg flies(X)}$$

The exchange of information between the agents is illustrated in Fig. 2.

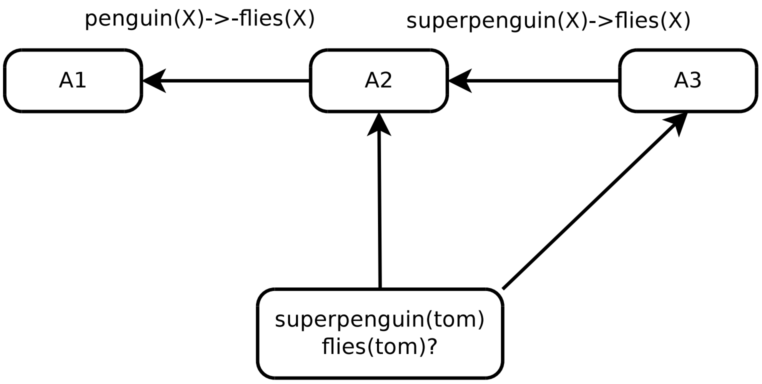


Fig. 2 Knowledge Exchange

Example 7. As more and more exceptions are acquired by an agent, the second transformation modifies the default rules locally by adding to it consecutive exceptions. At a given point it may happen that the agent's knowledge has to be revised, because the semantics of a rule can become out of date. If, for instance, initially an agent observes only non-flying birds:

$$\begin{aligned}\mathbf{A_1} : E^{+}_{flies} &= \{\} \\ E^{-}_{flies} &= \{1, 2\} \\ B &= \{penguin(1), ostrich(2), \\ &\quad penguin(X) \to bird(X) \\ &\quad ostrich(X) \to bird(X)\}\end{aligned}$$

it will induce a rule that birds typically do not fly.

$$\frac{bird(X) : \neg flies(X)}{\neg flies(X)}$$

However, as more and more exceptions to that rule (flying species of birds) are provided by other agents, it will finally become evident, that birds typically fly and that the examples initially observed should be exceptions, not the rule. Here, the third transformation will be used, so that the agent can update its database by the following rule:

$$\frac{bird(X) : flies(X) \wedge \neg penguin(X) \wedge \neg ostrich(X)}{flies(X)}$$

Now, we are ready to specify formally the algorithms for query answering, knowledge exchange and learning. Algorithm 2 describes actions performed by the agent seeking an answer to a query. It begins with comparing answers received from various agents. If the answers are consistent they can be used to generate the final answer to the query. In the opposite case, the agent compares generality/specificity of the answers. The most specific knowledge, preferably referring to the similar training example, should be used.

If there are contradictory answers of the same generality/specificity, support of the knowledge based on the number of training examples has to be compared. Knowledge with the biggest support will be applied. In case of rules, for which generality/specificity is undecided, the DDL confidence measure will be used in order to decide priority.

Algorithm 2: Query

```
begin
    $Answers \leftarrow askOtherAgents$;
    if $Consistent(Answers)$ then
        $Result \leftarrow Distinct(Answers)$;
    else
        if $Sources \leftarrow FindMostSpecificKnowledge(Query, Answers)$ then
            if $|Source| = 1$ then
                $Source = Sources$
            else
                $Source = SelectMaxSupport(Sources)$
        else
            $Source = SelectMaxConfidence(Sources)$
        $Result \leftarrow Answer(Source)$
        for $A_i, A_j \in Answers : answer(i) = \neg answer(j)$ and $rationale(i)$
        more general than $rationale(j)$ do
            pass exception from j to i;
end
```

Algorithm 3 compares answers and rationale given by various agents, and decides which is the most general answer with respect to the given query. If the predicates used by various agents belong to the same taxonomy, it can easily be decided which is most general. The taxonomy can be reperesented in the form of rules in agents' background knowledge, or in the form of additional ontology.

Algorithm 3: FindMostSpecificKnowledge

```
begin
    $Candidates \leftarrow \{i : \nexists j : pred(i) \rightarrow pred(j)\}$;
    Return i : priority(i) = maxpriority(Candidates);
end
```

8.6 Conclusions

In the paper we have concentrated on a novel approach to distributed and incremental learning of default theories of autnomous agents within a multi-agent system. The learned knowledge of the MAS is represented by the Distributed Default Logic formalism, which is used for expressing domain knowledge. We have defined a set of transformation rules, which allow the agents to reformulate the default rules, according the discovered elementary facts, and induced implications, so that the local and global default theory of MAS better describes the discovered knowledge. In the transformations the support values of induced implications are taken into account.

We have presented algorithms, in which the transformation rules are used. They enable the group of agents to enhance the global knowledge of the MAS within the process of cooperation of the agents by means of exchanging the locally acquired knowledge. In addition, the algorithms allow agents to cooperate in reasoning in order to answer queries.

If during the reasoning process the agents observe inconsistencies in answers and/or the global theory, the asking agent is obliged to tell agents possessing relevant examples to pass them on to agents holding more general knowldge. They in turn incorporate this new knowledge in the form of exceptions to default rules.

It is worth noting that the transformation rules do not take into account a possibility of invalidating of observed examples by newcomming facts. In the future we plan adapting the formalism, so that it would take into account this possibility as well.

References

1. Alonso, E., D'Inverno, M., Kudenko, D., Luck, M., Noble, J.: Learning in multi-agent systems. Knowledge Engineering Review 16, 277–284 (2001)
2. Antonelli, G.A.: A directly cautious theory of defeasible consequence for default logic via the notion of general extension. Artificial Intelligence 109, 71–109 (1999)
3. Benferhat, S., Dubois, D., Lagrue, S., Prade, H.: Towards Learning Default Rules by Identifying Big-Stepped Probabilities. In: IFSA World Congress (2002)
4. Bratko, I.: Applications of inductive logic programming. Communications of the ACM 38(11) (November 1995)

5. Dimopoulos, Y., Kakas, A.: Learning Non-Monotonic Logic Programs: Learning Exceptions. In: Lavrač, N., Wrobel, S. (eds.) ECML 1995. LNCS, vol. 912, pp. 122–137. Springer, Heidelberg (1995)
6. Duval, B., Nicolas, P.: Learning Default Theories. In: Hunter, A., Parsons, S. (eds.) ECSQARU 1999. LNCS (LNAI), vol. 1638, pp. 148–159. Springer, Heidelberg (1999)
7. Fonseca, N.A., Silva, F., Camacho, R.: Strategies to Parallelize ILP Systems. In: Kramer, S., Pfahringer, B. (eds.) ILP 2005. LNCS (LNAI), vol. 3625, pp. 136–153. Springer, Heidelberg (2005)
8. Governatori, G., Stranieri, A.: Towards the application of association rules for defeasible rules discovery. In: Verheij, B., Lodder, A., Loui, R.P., Muntjerwerff, A.J. (eds.) Legal Knowledge and Information Systems, pp. 63–75. IOS Press, Amsterdam (2001)
9. Inoue, K., Kudoh, Y.: Learning Extended Logic Programs. In: Proc. of the 15th IJCAI, vol. 1, pp. 176–181. Morgan Kaufmann (1997)
10. Johnston, B., Governatori, G.: An algorithm for the induction of defeasible logic theories from databases. In: Schewe, K.-D., Zhou, X. (eds.) Conference Research and Practice of Information Technology, Database Technology 2003, Australian Computer Science Association, ACS, vol. 17, pp. 75–83 (2003)
11. Kazakov, D., Kudenko, D.: Machine Learning and Inductive Logic Programming for Multi-Agent Systems. In: Luck, M., Mařík, V., Štěpánková, O., Trappl, R. (eds.) ACAI 2001 and EASSS 2001. LNCS (LNAI), vol. 2086, pp. 246–270. Springer, Heidelberg (2001)
12. Lukaszewicz, W.: Considerations on default logic: An alternative approach. Computational Intelligence 4(1), 1–16 (1988)
13. Muggleton, S.: Inductive Logic Programming. New Generation Computing 8(4), 295–318 (1991)
14. Muggleton, S., de Raedt, L.: Inductive Logic Programming: Theory and methods. The Journal of Logic Programming 19-20(suppl.1), 629–679 (1994)
15. Nienhuys-Cheng, S.-H., de Wolf, R. (eds.): Foundations of Inductive Logic Programming. LNCS(LNAI), vol. 1228. Springer, Heidelberg (1997)
16. Przymusiǹska, H., Przymusiǹski, T.: Stationary Default Extensions. Fundamenta Informaticae 21, 67–87 (1994)
17. Reiter, R.: A Logic for Default Reasoning. In: Artificial Intelligence, vol. 13, pp. 81–132 (1980)
18. Ryżko, D., Rybinski, H.: Distributed Default Logic for Multi-Agent System. In: Proc. WI/IAT (2003)
19. Sakama, C., Inoue, K.: Prioritized logic programming and its application to commonsense reasoning. In: AI, vol. 123(1-2), pp. 185–222 (2000)
20. Weiss, G., Dillenbourg, P.: What is `multi` in multiagent learning? In: Dillenbourg, P. (ed.) Collaborative Learning. Cognitive and Computational Approaches, pp. 64–80. Pergamon Press (1999)
21. Więch, P., Rybiński, H.: A Novel Approach to Default Reasoning for MAS. In: Szczuka, M., Kryszkiewicz, M., Ramanna, S., Jensen, R., Hu, Q. (eds.) RSCTC 2010. LNCS, vol. 6086, pp. 484–493. Springer, Heidelberg (2010)

Chapter 9
Rough Non-deterministic Information Analysis: Foundations and Its Perspective in Machine Learning

Hiroshi Sakai, Hitomi Okuma, and Michinori Nakata

Abstract. This chapter focuses on a mathematical framework for handling information incompleteness, which is deeply related to machine learning. Recently, the handling of the information incompleteness in data sets is recognized to be very important research area for machine learning. We have already proposed a framework *Rough Non-deterministic Information Analysis* (*RNIA*). This is a rough sets based framework for handling not only definite (or complete) information but also indefinite (or incomplete) information. This *RNIA* handles lots of aspects in tables with the information incompleteness, i.e., rough sets based issues, data dependencies, question-answering, rule generation, estimation of actual values, etc. Each aspect is extended from tables with complete information to tables with incomplete information according to the modal concepts. We survey this *RNIA*, and we describe the perspective of *RNIA* with respect to machine learning.

9.1 Introduction

Rough set theory offers a mathematical approach to vagueness and uncertainty, and the rough sets based concepts have been recognized to be very useful [19, 28, 34, 35, 36, 37, 38, 39, 41, 42, 68, 72, 76]. This theory usually handles tables

Hiroshi Sakai
Department of Basic Sciences, Faculty of Engineering, Kyushu Institute of Technology, Tobata, Kitakyushu 804, Japan
e-mail: sakai@mns.kyutech.ac.jp

Hitomi Okuma
Faculty of Education and Welfare Science, Oita University Dannoharu, Oita 870, Japan
e-mail: okuma@oita-u.ac.jp

Michinori Nakata
Faculty of Management and Information Science, Josai International University, Gumyo, Togane, Chiba 283, Japan
e-mail: nakatam@ieee.org

with deterministic information, which we call *Deterministic Information Systems* (*DISs*). Many applications of this theory to classification analysis [3, 23, 63, 70], data mining [11, 26], reduction [8, 18, 20, 64], rule generation [4, 12, 14, 69], machine learning [6] and incomplete and non-deterministic information systems [9, 15, 21, 22, 24, 25, 27, 29, 30, 31, 32, 33, 66, 67] have been investigated.

Non-deterministic Information Systems (*NISs*) and *Incomplete Information Systems* (*IISs*) have been proposed for handling information incompleteness in *DISs* [24, 25, 32, 33]. *NISs* have been recognized to be the most important framework for handling information incompleteness in tables, and several theoretical works has been reported [9, 15, 21, 22, 24, 25, 27, 29, 30, 31, 32, 33, 66, 67]. We follow this robust framework, and we have been developing algorithms and software tools, which can handle rough sets based concepts in *NISs*. We are simply calling this work *Rough Non-deterministic Information Analysis* (*RNIA*) [44,45,47-61].

In this chapter, we survey our previous work related to *RNIA*, and we also describe its perspective in machine learning. Throughout this chapter, we will employ lots of examples for knowing this *RNIA* intuitively, and we show related papers for knowing more details. We think that this chapter will take the role of linking each example to the formal definitions and details of *RNIA*.

This chapter is organized as follows: Section 2 recalls the foundations of rough sets in *DISs*. Section 3 and 4 introduce the frame work of *RNIA*, and survey several extended aspects from *DISs* to *NISs*. Especially in Section 5, we describe an aspect of rule generation in *NISs*. Section 6 proposes the next hot topics in *RNIA* as perspective, and we will consider the relation between rough sets and logic programs. Section 7 concludes this chapter.

9.2 Foundations of Rough Sets in DISs

This section recalls the foundations of rough sets and the manipulation algorithms for equivalence classes in *DISs*.

9.2.1 Some Definitions and Aspects in DISs

A *Deterministic Information System* (*DIS*) ψ is a quadruplet [36, 38]

$\psi=(OB, AT, \{VAL_A \mid A \in AT\}, f)$,

where OB is a finite set whose elements are called *objects*, AT is a finite set whose elements are called *attributes*, VAL_A is a finite set whose elements are called *attribute values* and f is such a mapping

$f : OB \times AT \rightarrow \cup_{A \in AT} VAL_A$.

We usually consider a table instead of this quadruplet ψ. A *DIS* ψ_{Table1} in Table 1 is an exemplary deterministic information system. We employ it for showing each aspect.

Table 1 An exemplary *DIS* ψ_{Table1} for the suitcase data sets. Here, VAL_{Color}=$\{red, blue, green\}$, VAL_{Size}=$\{small, medium, large\}$, VAL_{Weight}=$\{light, heavy\}$, VAL_{Price}=$\{high, low\}$.

Object	Color	Size	Weight	Price
x_1	red	small	light	low
x_2	red	medium	light	high
x_3	blue	medium	light	high
x_4	red	medium	heavy	low
x_5	red	large	heavy	high
x_6	blue	large	heavy	high

In each ψ and a subset $ATR \subseteq AT$, we employ a notation ATR=$\{A_1, \cdots, A_n\}$. Each index i at A_i is the tentative ordinal number in a set ATR, and is not the ordinal number in the original data set. For ATR and an object x, $(f(x,A_1), \cdots, f(x,A_n))$ is a *tuple* of x.

If $f(x,A_i)$=$f(y,A_i)$ holds for every $A_i \in ATR \subseteq AT$, we see there is a relation between x and y for ATR. This relation is an equivalence relation over OB [38]. Let $eq(ATR)$ denote a set of the equivalence classes with respect to ATR, and let $[x]_{ATR} \in eq(ATR)$ denote an equivalence class below:

$\{y \in OB | f(y,A_i)$=$f(x,A_i)$ for every $A_i \in ATR\}$.

In rough sets, we effectively employ equivalence classes.

According to ψ_{Table1}, let us consider four cases (A), (B), (C) and (D) of ATR.

(A) For ATR=$\{Size, Weight\}$,
a tuple of x_1 is $(small, light)$, $eq(\{Size, Weight\})$=$\{\{x_1\}, \{x_2, x_3\}, \{x_4\}, \{x_5, x_6\}\}$,
$[x_1]_{\{Size,Weight\}}$=$\{x_1\}$, $[x_2]_{\{Size,Weight\}}$=$\{x_2, x_3\}$.

(B) For ATR=$\{Color, Size, Weight\}$,
a tuple of x_1 is $(red, small, light)$,
$eq(\{Color, Size, Weight\})$=$\{\{x_1\}, \{x_2\}, \{x_3\}, \{x_4\}, \{x_5\}, \{x_6\}\}$,
$[x_i]_{\{Color,Size,Weight\}}$=$\{x_i\}$ (i=$1, 2, \cdots, 6$).

(C) For ATR=$\{Weight\}$,
a tuple of x_1 is $(light)$, $eq(\{Weight\})$=$\{\{x_1, x_2, x_3\}, \{x_4, x_5, x_6\}\}$,
$[x_1]_{\{Weight\}}$=$\{x_1, x_2, x_3\}$.

(D) For ATR=$\{Price\}$,
a tuple of x_1 is (low), $eq(\{Price\})$=$\{\{x_1, x_4\}, \{x_2, x_3, x_5, x_6\}\}$,
$[x_1]_{\{Price\}}$=$\{x_1, x_4\}$.

Now, let us sequentially consider five rough sets based aspects by using the above four cases (A), (B), (C) and (D).

(Aspect 1) *The Definability of a Set in a DIS* ψ*:* If a set $X \subseteq OB$ is the union of some equivalence classes in $eq(ATR)$, we say X is *definable* (for ATR) in ψ. Otherwise, we say X is *rough* (for ATR) in ψ.

In case (B), any set $X \subseteq OB$ is definable for ATR=$\{Color, Size, Weight\}$, because X=$\cup_{x \in X}[x]_{\{Color,Size,Weight\}}$ holds. However, X=$\{x_1, x_2\}$ is not definable for ATR=$\{Size, Weight\}$ in case (A). Both in cases neither (C) nor (D), X=$\{x_1, x_2\}$ is not definable.

(Aspect 2) *The Consistency of an Object:* Let us consider two disjoint sets $CON \subseteq AT$ which we call *condition attributes* and $DEC \subseteq AT$ which we call *decision attributes*. An object $x \in OB$ is *consistent* if $f(x,A)$=$f(y,A)$ holds for every $A \in CON$ implies $f(x,A)$=$f(y,A)$ holds for every $A \in DEC$.

Let CON be $\{Weight\}$ and DEC be $\{Price\}$. In this case, $[x_1]_{\{Weight\}}$=$\{x_1, x_2, x_3\}$ holds, but $f(x_1, Price) \neq f(x_3, Price)$ holds. Thus, object x_1 is not consistent. Similarly, all 6 objects are not consistent.

Rough set theory makes use of equivalence classes for solving problems. Here, let us show the most important proposition, which connects two equivalence classes $[x]_{CON}$ and $[x]_{DEC}$ with the consistency of an object x.

Proposition 1. *[38] For each DIS, (1) and (2) in the following are equivalent.*
(1) An object $x \in OB$ is consistent for CON and DEC.
(2) $[x]_{CON} \subseteq [x]_{DEC}$.

(Aspect 3) *The Degree of Dependency:* The *degree of dependency* for CON and DEC is a ratio

$deg(CON, DEC)$=$|\{x \in OB|\ x$ is consistent for CON and $DEC\ \}|/|OB|$.

Clearly, $deg(CON, DEC)$=1.0 holds if and only if every object $x \in OB$ is consistent.

For CON=$\{Weight\}$ and DEC=$\{Price\}$ in ψ_{Table1}, any object is not consistent. Therefore,

$deg(\{Weight\}, \{Price\})$= 0/6 = 0.0.

For CON=$\{Color, Size, Weight\}$ and DEC=$\{Price\}$ in ψ_{Table1}, any object is consistent. Therefore,

$deg(\{Color, Size, Weight\}, \{Price\})$= 6/6 = 1.0.

(Aspect 4) *Reduction of Condition Attributes:* Let us consider a consistent object x for CON and DEC. An attribute $A \in CON$ is *dispensable* in CON, if x is also consistent for $CON \setminus \{A\}$.

Object x_1 is consistent for CON=$\{Color, Size, Weight\}$ and DEC=$\{Price\}$. However, x_1 is also consistent for CON=$\{Size\}$. Namely, both *Color* and *Weight* are dispensable for x_1. If we take CON=$\{Size, Weight\}$, each object is consistent. Thus, *Color* is dispensable for each object.

(Aspect 5) *Rules and Criteria (Support, Accuracy and Coverage):* For any object $x \in OB$, let $imp(x, CON, DEC)$ denote a formula called an *implication*:

$\wedge_{A \in CON}\ [A, f(x,A)] \Rightarrow \wedge_{A \in DEC}[A, f(x,A)]$,

where a formula $[A, f(x,A)]$ implies that $f(x,A)$ is the value of the attribute A. This is called a *descriptor* [19, 24, 36, 63]. In most of work on rule generation, a rule is defined by an implication $\tau^x : imp(x, CON, DEC)$ satisfying some constraints.

A constraint, such that $deg(CON,DEC)$=1.0 holds for CON and DEC, has been proposed in [36, 37, 38].

In ψ_{Table1}, we know the following implication from object x_1

$[Size,small] \wedge [Weight,light] \Rightarrow [Price,low]$

is a rule, because $deg(\{Size,Weight\},\{Price\})$=1.0. Furthermore, we apply the reduction to this implication, and have the following *minimal* rule from object x_1,

$[Size,small] \Rightarrow [Price,low]$.

Another familiar constraint is defined by three values in the following:

$support(\tau^x)= |[x]_{CON} \cap [x]_{DEC}|/|OB|$,

$accuracy(\tau^x)=|[x]_{CON} \cap [x]_{DEC}|/|[x]_{CON}|$,

$coverage(\tau^x)=|[x]_{CON} \cap [x]_{DEC}|/|[x]_{DEC}|$.

Since $[x]_{CON}$, $[x]_{DEC}$ and $[x]_{CON} \cap [x]_{DEC}$ are also equivalence classes for attributes CON, DEC and $CON \cup DEC$, the following holds.

$support(\tau^y)=support(\tau^x)$, $accuracy(\tau^y)=accuracy(\tau^x)$ and

$coverage(\tau^y)=coverage(\tau^x)$ for each $y \in [x]_{CON} \cap [x]_{DEC}$.

Therefore, we may handle τ instead of τ^x in each ψ. Here, we clarify two standard rule generation tasks.

Specification of Rule Generation Tasks in a *DIS*

For threshold values α and β $(0 < \alpha, \beta \leq 1)$, find each implication τ satisfying $support(\tau) \geq \alpha$ and $accuracy(\tau) \geq \beta$. We say this is a *criterion based rule generation* in a *DIS*. Especially, if β=1.0, we say this is a *consistency based rule generation* in a *DIS*.

The *Apriori* algorithm [1, 2] proposed to search for such criterion based rules by Agrawal is now one of the most representative methods in data mining [5]. As for the consistency based rule generation, a *discernibility function* method [63] by Skowron is known well.

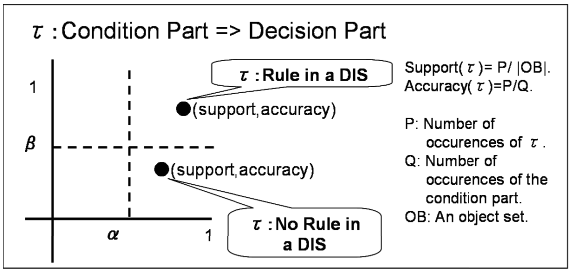


Fig. 1 A pair (*support*,*accuracy*) corresponding to the implication τ.

9.2.2 Manipulation Algorithms for Equivalence Relations and Data Dependency

For dealing with equivalence relations, we introduced a data structure. This is simple, but we can generate new equivalence relation from two equivalence relations.

In this structure, we identify the number i with an object x_i, so each equivalence class $[i]_{ATR}$ is a set of numbers, and each element is ordered in a class $[i]_{ATR}$. Two arrays $head[i]$ and $succ[i]$ $(1 \leq i \leq |OB|)$, which are correctly $head\ [x_i]$ and $succ[x_i]$, are defined as follows:

For $j \in [i]_{ATR}$,
 $head[j]$ is the first element of the equivalence class $[i]_{ATR}$,
 $succ[j]$ is the successor to j in $[i]_{ATR}$.
For the last $j \in [i]_{ATR}$,
 $head[j]$ is the first element of the equivalence class $[i]_{ATR}$,
 $succ[j]$=0.

In ψ_{Table1}, $eq(\{Price\})$=$\{\{x_1,x_4\},\{x_2,x_3,x_5,x_6\}\}$ holds, and we express it by
$head[1]$=1, $head[2]$=2, $head[3]$=2, $head[4]$=1, $head[5]$=2, $head[6]$=2,
$succ[1]$=4, $succ[2]$=3, $succ[3]$=5, $succ[4]$=0, $succ[5]$=6, $succ[6]$=0.
Since $[i]_{ATR}$=$\{head[i], succ[head[i]], \cdots, succ[\cdots succ[head[i]] \cdots]\}$ holds, $[3]_{\{Price\}}$ = $\{2,3,5,6\}$ is obtained by $head[3]$=2, $succ[2]$=3, $succ[3]$=5, $succ[5]$=6, $succ[6]$=0.

Algorithm 1 in the appendix generates $head[i]$ and $succ[i]$ $(1 \leq i \leq |OB|)$. In the worst case of Algorithm 1, it is necessary to compare two objects $((|OB|-1)+(|OB|-2)+\cdots+1)$ times. So, the worst computational complexity of Algorithm 1 is $O(|OB|^2)$.

Now, we consider to generating an equivalence relation
$eq(A \cup B)$=$\{M \subseteq OB | M = [i]_A \cap [i]_B$, $[i]_A \in eq(A)$ and $[i]_B \in eq(B)\}$
from two equivalence relations $eq(A)$ and $eq(B)$. Algorithm 2 in the appendix generates new equivalence relation $eq(A \cup B)$.

By Algorithm 1, we at first generate $eq(A_i)$ $(A_i \in AT)$, then we apply Algorithm 2 to generating $eq(ATR)$ $(ATR$=$\{A_1,A_2,\cdots,A_m\})$. In this way, we can handle each equivalence relation for any $ATR \subseteq AT$. Algorithm 2 is more effective for merging several kinds of attributes. Even though the order may be $O(|OB|^2)$ in the generation of $eq(\{A_1,A_2\})$, the order becomes to $O(|OB|)$ in the last application of Algorithm 2. Because, every equivalence class $[i]_{\{A_1,A_2,\cdots,A_k\}}$ converges to a set $\{i\}$ by adding attributes.

We can also apply $[x]_{CON}$ and $[x]_{DEC}$ to calculating the degree of dependency in a DIS. Algorithm 3 in the appendix calculates the degree, and Algorithm 4 calculates the subpart $(* :\ inclusion)$ in Algorithm 3.

The details of the manipulation in this sub section are in [45, 48, 51].

9.3 Foundations of Rough Non-deterministic Information Analysis

This section surveys a framework of *RNIA* (*Rough Non-deterministic Information Analysis*), *possible equivalence relations* and *data dependency* in *NISs*.

9.3.1 Some Definitions and Aspects in NISs

A *Non-deterministic Information System* (*NIS*) Φ is also a quadruplet [32, 37, 38]
Φ=$(OB, AT, \{VAL_A | A \in AT\}, g)$,
$g : OB \times AT \to P(\cup_{A \in AT} VAL_A)$ (a power set of $\cup_{A \in AT} VAL_A$).
Every set $g(x,A)$ is interpreted as that there is an actual value in this set but this value is not known [32, 37, 38]. Especially if the real value is not known at all, $g(x,A)$ is equal to VAL_A. This is called the *null value* interpretation [7] or missing value [15, 22, 66]. We usually consider a table instead of this quadruplet Φ. Let us consider an exemplary *NIS* Φ_{Table2} in Table 2.

Table 2 An exemplary *NIS* Φ_{Table2} for the suitcase data sets. Here, VAL_{Color}=$\{red, blue, green\}$, VAL_{Size}=$\{small, medium, large\}$, VAL_{Weight}=$\{light, heavy\}$, VAL_{Price}=$\{high, low\}$.

Object	Color	Size	Weight	Price
x_1	{red,blue,green}	{small}	{light,heavy}	{low}
x_2	{red}	{small,medium}	{light,heavy}	{high}
x_3	{red,blue}	{small,medium}	{light}	{high}
x_4	{red}	{medium}	{heavy}	{low,high}
x_5	{red}	{small,medium,large}	{heavy}	{high}
x_6	{blue,green}	{large}	{heavy}	{low,high}

In Φ_{Table2}, $g(x_1, Color)$=VAL_{Color} holds, and this means there is no information about this attribute value, namely we identify Φ_{Table2} with Table 3.

Table 3 A table with non-deterministic information and null values. The $*$ symbol means a null value, and we identify $*$ with a set of attribute value.

Object	Color	Size	Weight	Price
x_1	$*$	small	$*$	low
x_2	red	{small,medium}	$*$	high
x_3	{red,blue}	{small,medium}	light	high
x_4	red	medium	heavy	$*$
x_5	red	$*$	heavy	high
x_6	{blue,green}	large	heavy	$*$

In the previous work, non-deterministic information seems to be identified with a null value. However, each $g(x_3,Color)$, $g(x_6,Color)$, $g(x_2,Size)$ and $g(x_3,Size)$ is different from neither a null value nor a missing value. We have clarified the property of non-deterministic information, and we are proposing a new framework.

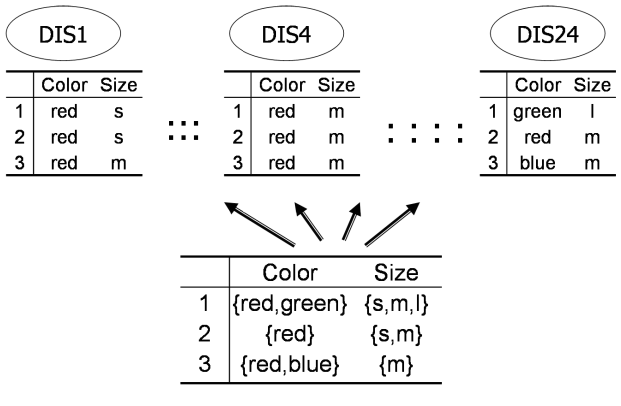
9.3.2 A Basic Chart and Two Modalities

Now, we introduce a *derived DIS* from a *NIS*, and show the basic chart in *RNIA*. Since each VAL_A ($A \in AT$) is finite, we can generate a *DIS* by replacing each non-deterministic information $g(x,A)$ with an element in $g(x,A)$. We named such a *DIS* a *derived DIS* from a *NIS*, and define the following.

$DD(\Phi)=\{\psi \mid \psi$ is a derived *DIS* from a *NIS* $\Phi\}$.

In Φ_{Table2}, there are 2304 (=$3^2 \times 2^8$) derived *DISs*, and $\psi_{Table_1} \in DD(\Phi_{Table2})$ holds. Due to the interpretation of non-deterministic information, we see an actual ψ^{actual} exists in this 2304 derived *DISs*. Like this, we always consider the *basic chart* for a *NIS* Φ and a set of derived *DISs* $DD(\Phi)$.

Fig. 2 is another example of the basic chart. For a *NIS* Φ_{Fig2}, a set $DD(\Phi_{Fig2})$ consists of 24 derived *DISs*. According to the basic chart, it is possible to define two modalities the *certainty* and the *possibility* for each aspect.



	Color	Size
1	red	s
2	red	s
3	red	m

	Color	Size
1	red	m
2	red	m
3	red	m

	Color	Size
1	green	l
2	red	m
3	blue	m

	Color	Size
1	{red,green}	{s,m,l}
2	{red}	{s,m}
3	{red,blue}	{m}

Fig. 2 An example of the basic chart for Φ_{Fig2} and a set of derived DISs $DD(\Phi_{Fig2})$.

(Certainty) If a formula α holds in every $\psi \in DD(\Phi)$, α also holds in ψ^{actual}. In this case, we say α *certainly holds* in ψ^{actual}.
(Possibility) If a formula α holds in some $\psi \in DD(\Phi)$, there exists such a possibility that α holds in ψ^{actual}. In this case, we say α *possibly holds* in ψ^{actual}.

Even if there exists the information incompleteness in a Φ, we can have the following decision making.

(1) If a formula α certainly holds, we think α holds under the uncertainty.
(2) If a formula α possibly holds, we think α may hold under the uncertainty.
(3) Otherwise, we think α does not hold under the uncertainty.

In *RNIA*, we follow the rough sets based aspects in *DISs*, and reconsider the certainty and the possibility of rough sets based aspects in *NISs*.

9.3.3 Computational Complexity in NISs

Here, we must pay attention to the computational complexity related to a *NIS*. For a *NIS* Φ, the number of derived *DISs* increases in the exponential order. Therefore, it will be hard to apply the *explicit method* such that we sequentially examine each concept in $\psi \in DD(\Phi)$.

In each aspect, we do not employ this explicit method. Instead of this explicit method, we propose methods depending upon equivalence classes in the derived *DISs*. Especially in rule generation, we had an algorithm which does not depend upon $DD(\Phi)$ at all.

9.3.4 Possible Equivalence Classes in NISs

For a *NIS*, we call an equivalence relation in a derived *DIS* a *possible equivalence relation* (*pe-relation*) in a *NIS*. A *pe*-relation defines a set $peq(ATR)$ of all *possible equivalence classes* (*pe-class*) in a *NIS*. For example in Fig. 2, we obtain
$peq(\{Color, Size\})=\{\{1,2,3\}\}$ in DIS_4,
$peq(\{Color, Size\})=\{\{1\},\{2\},\{3\}\}$ in DIS_{24}.
Both classes $\{1,2,3\}$ and $\{1\}$ are *pe*-classes with object 1. It is necessary to characterize each *pe*-class for handling rough sets based concepts in Section 2. We first define two sets *inf* and *sup* for each descriptor, which is given in (Aspect 5).

Definition 1. For a *NIS* with a function $g : OB \times AT \rightarrow P(\cup_{A \in AT} VAL_A)$ and a set of descriptors $[A_i, \zeta_i]$ ($A_i \in ATR \subseteq AT$), we define two sets *inf* and *sup*.
(1) For a descriptor $[A_i, \zeta_i]$, we define the following.
$inf([A_i, \zeta_i])=\{x \in OB | g(x, A_i)=\{\zeta_i\}\}$,
$sup([A_i, \zeta_i])=\{x \in OB | \zeta_i \in g(x, A_i)\}$.
(2) For a compound descriptor $[ATR, \zeta_{ATR}](=[\{A_1, \cdots, A_n\}, (\zeta_1, \cdots, \zeta_n)])$,
$inf([ATR, \zeta_{ATR}])=\{x \in OB | g(x, A_i)=\{\zeta_i\}$ for each $i\}$,
$sup([ATR, \zeta_{ATR}])=\{x \in OB | \zeta_i \in g(x, A_i)$ for each $i\}$.

We can directly obtain the next proposition from Definition 1.

Proposition 2. *[45, 50] For descriptors $[A_i, \zeta_i]$ ($A_i \in ATR \subseteq AT$), $ATR=\{A_1, \cdots, A_n\} \subseteq AT$ and a tuple $\zeta_{ATR}=(\zeta_1, \cdots, \zeta_n)$, we obtain the following.*
(1) $inf([ATR, \zeta_{ATR}])=\cap_i\, inf([A_i, \zeta_i])$.
(2) $sup([ATR, \zeta_{ATR}])=\cap_i\, sup([A_i, \zeta_i])$.

According to Proposition 2, we can easily calculate two sets inf and sup for each compound descriptor $[ATR,\zeta_{ATR}]$. For example in Fig. 2,

$inf([Color,red])=\{1\}$, $sup([Color,red])=\{1,2,3\}$,
$inf([Size,m])=\{3\}$, $sup([Size,m])=\{1,2,3\}$,
$inf([\{Color,Size\},(red,m)])=\{1\}\cap\{3\}=\emptyset$,
$sup([\{Color,Size\},(red,m)])=\{1,2,3\}\cap\{1,2,3\}=\{1,2,3\}$

are derived. These inf and sup in Definition 1 and Proposition 2 are key information for *RNIA*. The set sup is semantically equal to a set defined by the similarity relation *SIM* [21, 22]. In [21, 22], some theorems are presented based on the relation *SIM*, and our theoretical results are closely related to those theorems. However, the set sup causes new properties, which hold just in *NISs*.

Now, let us consider a relation between each *pe*-class and each compound descriptor $[ATR,\zeta_{ATR}](=[\{A_1,\cdots,A_n\},(\zeta_1,\cdots,\zeta_n)])$.

Definition 2. For a *NIS*, a compound descriptor $[ATR,\zeta_{ATR}]$ and a derived *DIS* ψ, let $Pe_{[ATR,\zeta_{ATR}],\psi}$ denote a *pe*-class defined by $[ATR,\zeta_{ATR}]$ and ψ.

In Fig. 2, $Pe_{[Color,red],DIS_1}=\{1,2,3\}$, $Pe_{[Size,s],DIS_1}=\{1,2\}$ and $Pe_{[Color,blue],DIS_1}=\emptyset$ hold. If we see a *DIS* ψ is a *NIS* with only singleton sets,

$Pe_{[ATR,\zeta_{ATR}],\psi}=inf([ATR,\zeta_{ATR}])=sup([ATR,\zeta_{ATR}])$

holds in ψ. However in every *NIS*, $Pe_{[ATR,\zeta_{ATR}],\psi}$ depends upon a derived *DIS* ψ, and generally

$inf([ATR,\zeta_{ATR}])\subseteq Pe_{[ATR,\zeta_{ATR}],\psi}\subseteq sup([ATR,\zeta_{ATR}])$

holds. Proposition 3 connects a *pe*-class $Pe_{[ATR,\zeta_{ATR}],\psi}$ with $inf([ATR,\zeta_{ATR}])$ and $sup([ATR,\zeta_{ATR}])$.

Proposition 3. *[50, 57] The conditions (1) and (2) in the following are equivalent.*
(1) X is a pe-class $Pe_{[ATR,\zeta_{ATR}],\psi}$.
(2) $inf([ATR,\zeta_{ATR}])\subseteq X\subseteq sup([ATR,\zeta_{ATR}])$.
Due to (1) and (2), for a set $M\subseteq sup([ATR,\zeta_{ATR}])\setminus inf([ATR,\zeta_{ATR}])$),

$X=Pe_{[ATR,\zeta_{ATR}],\psi}=inf([ATR,\zeta_{ATR}])\cup M$.

9.3.5 Some Extended Aspects to NISs

Now, we sequentially consider rough sets based aspects in *NISs*.

9.3.5.1 The Definability of a Set in NISs

We can extend (Aspect 1) in Section 2 to an aspect of a *NIS*.

(Certainly definable) A set X is *certainly definable*, if X is definable in each $\psi\in DD(\Phi)$.
(Possibly definable) A set X is *possibly definable*, if X is definable in some $\psi\in DD(\Phi)$.

Here, X=$\cup_{x \in X}\{x\}$ and $x \in [x]_{AT}$=$Pe_{[AT,\zeta_{AT}],\psi}$ hold. If each x and each $\zeta_{AT,x}$ (a tuple of x) satisfies $sup([AT,\zeta_{AT,x}]) \subseteq X$, any pe-class $[x]_{AT}$ is a subset of X, because sup is the maximum pe-class. Therefore, we conclude $\cup_{x \in X}[x]_{AT} \subseteq X$. Since $X \subseteq \cup_{x \in X}[x]_{AT}$ clearly holds, we conclude X=$\cup_{x \in X}[X]_{AT}$ for each pe-class. As for the possibility, we employ $inf([AT,\zeta_{AT,x}])$ instead of $sup([AT,\zeta_{AT,x}])$.

9.3.5.2 Pe-relations in NISs

Here, we refer to an algorithm to examine whether a set X is possibly definable or not. We have already investigated it, and published it in [48, 50, 54]. This program is implemented in a logic programming language Prolog. This tree search program employs inf and sup information for attributes ATR, and solves the definability of a set as a constraint satisfaction problem.

We can also apply the above algorithm to obtaining all pe-relations in a NIS. Namely, OB is definable in any derived DIS and any ATR, so this algorithm finds a solution, and we can obtain a $peq(ATR)$ as a side effect of the solution. The details are in [48, 50, 54]. The following is an execution of pe-relation generation from Φ_{Table2} on a windows 7 PC (3.3GHz, 64bitCPU).

```
?-trans. /* Translation data to the internal expression */
 File Name for Read Open: suitcase.pl.
 EXEC_TIME=0.0(sec)
 yes

?-pe. /* Generation of all pe-relations in each attribute */
 Original File Name: suitcase.pl.
 -- 1.pe----------
 [1][[1,2,3,4,5],[6]] 2
 [2][[1,2,4,5],[3,6]] 1
 [3][[1,2,4,5],[3],[6]] 1
 [4][[1,6],[2,3,4,5]] 2
 [5][[1],[2,3,4,5],[6]] 2
    :   :
 [9][[1,6],[2,4,5],[3]] 1
 POSSIBLE CASES 12
 -- 2.pe----------
 [1] [[1,2,3],[4],[5,6]] 1
 [2] [[1,2],[3,4],[5,6]] 1
    :   :
 [12] [[1],[2,3,4,5],[6]] 1
 POSSIBLE CASES 12
 -- 3.pe----------
 [1] [[1,2,3],[4,5,6]] 1
 [2] [[1,3],[2,4,5,6]] 1
 [3] [[1,2,4,5,6],[3]] 1
```

```
[4] [[1,4,5,6],[2,3]] 1
POSSIBLE CASES 4
-- 4.pe----------
[1] [[1,4,6],[2,3,5]] 1
[2] [[1,4],[2,3,5,6]] 1
[3] [[1,6],[2,3,4,5]] 1
[4] [[1],[2,3,4,5,6]] 1
POSSIBLE CASES 4
EXEC_TIME=0.0(sec)
yes
```

9.3.5.3 The Consistency of an Object in NISs

We can extend (Aspect 2) in Section 2 to an aspect of a *NIS*. Let *CON* be a set of condition attributes and *DEC* be a set of decision attributes.

(Certainly consistent) An object x is *certainly consistent* in a *NIS*, if x is consistent for *CON* and *DEC* in each $\psi \in DD(\Phi)$.
(Possibly consistent) An object x is *possibly consistent* in a *NIS*, if x is consistent for *CON* and *DEC* in some $\psi \in DD(\Phi)$.

The calculation of the consistency of an object x is shown in the next sub section about data dependency in *NISs*.

9.3.5.4 Data Dependency in NISs

As for the data dependency, we can extend (Aspect 3) in Section 2 to the *minimum data dependency* and the *maximum data dependency* in a *NIS*.

Definition 3. In a *NIS*, let us consider a set of condition attributes *CON* and a set of decision attributes *DEC*. For any derived *DIS* ψ, let $deg(CON,DEC,\psi)$ denote the data dependency $deg(CON,DEC)$ in ψ.
(1) Let $Min\_deg(CON,DEC)$ be $Min_{\psi}\{deg(CON,DEC,\psi)\}$, and we call it the *minimum degree* of *data dependency* for *CON* and *DEC*.
(2) Let $Max\_deg(CON,DEC)$ be $Max_{\psi}\{deg(CON,DEC,\psi)\}$, and we call it the *maximum degree* of *data dependency* for *CON* and *DEC*.

For these $Min\_deg(CON,DEC)$ and $Max\_deg(CON,DEC)$, we may sequentially examine $deg(CON,DEC,\psi)$ by using Algorithm 3 and 4 in Section 2. However, the number of derived *DISs* increases in the exponential order. Therefore, we employ the following steps instead of examining $deg(CON,DEC,\psi)$ sequentially.

(Step 1) We first generate all *pe*-relations $\{peq(CON)_i\}$ for *CON* by using the definability of a set *OB*.
(Step 2) We then generate all *pe*-relations $\{peq(DEC)_j\}$ for *DEC* by using the definability of a set *OB*.

(Step 3) The combinations of $\cup_{i,j}(peq(CON)_i, peq(DEC)_j)$ corresponds to all derived *DISs* for $CON \cup DEC$.
(Step 4) For each $(peq(CON)_i, peq(DEC)_j)$, we calculate $deg(CON, DEC, \psi_{i,j})$, and obtain the minimum and the maximum degrees.

Now, let us consider a case that CON=$\{Color, Size, Weight\}$ and DEC=$\{Price\}$ for Φ_{Table2}. In Section 3.5.2, we have already shown (Step 1) and (Step 2). In order to proceed to (Step 3), we employ Algorithm 1 and 2 in Section 2.

```
$ merge2w
Merging 1.pe ...
Merging 2.pe ...
Merging 3.pe ...
EXEC_TIME=0.000(sec)
```

The above is an execution of generating a set of $peq(\{Color, Size, Weight\})$. We specify the names of files to merge in a data file, and a program *mereg*2*w* in C simulates Algorithm 2. After this execution, we obtain a file 123.*pe*, and we know that there are 576 derived *DISs* for CON=$\{Color, Size, Weight\}$ and 20 different *pe*-relations. Namely, 576 cases are reduced to 20 cases.

In (Step 3), we have $peq(\{Color, Size, Weight\})_i$ $(1 \leq i \leq 20)$ and $peq(\{Price\})_j$ $(1 \leq j \leq 4)$. Therefore, we can calculate all 2304 degrees of dependencies by considering 80 combinations of $peq(\{Color, Size, Weight\})_i$ and $peq(\{Price\})_j$. We apply Algorithm 3 and 4 to 80 combinations. The following is an execution of data dependency in (Step 4).

```
$ depratio
File Name for Condition: 123.pe
File Name for Decision: 4.pe
--- Dependency Check ----------------
CRITERION 1(Num_of_Consistent_DISs/Num_of_All_DISs)
     Number of Derived DISs: 2304
     Number of Derived Consistent DISs: 1364
     Degree of Consistent DISs: 0.592
CRITERION 2(Total_Min_and_Max_Degrees)
     Minimum Degree of Dependency: 0.167
     Maximum Degree of Dependency: 1.000
--- Consistency Ratio for Every Object ---
Consistent ratio of object 1: 0.802(= 1848 / 2304)
Consistent ratio of object 2: 0.792(= 1824 / 2304)
Consistent ratio of object 3: 0.917(= 2112 / 2304)
Consistent ratio of object 4: 0.750(= 1728 / 2304)
Consistent ratio of object 5: 0.778(= 1792 / 2304)
Consistent ratio of object 6: 1.000(= 2304 / 2304)
EXEC_TIME=0.000(sec)
```

In the above execution, we know each object is consistent in 1364 derived *DISs*, and we know $Max\_deg(\{Color, Size, Weight\}, \{Price\})$=1.0 and $Min\_deg(\{Color, Size, Weight\}, \{Price\})$=0.167. Implementation of this procedure is discussed in [46, 49, 51, 54].

9.3.5.5 The Minimum and the Maximum of Criterion Values in NISs

At (Aspect 5) in Section 2, we have shown criteria $support(\tau^x)$ and $accuracy(\tau^x)$ in *DISs*. This $\tau^x(=\wedge_{A\in CON}[A, f(x,A)] \Rightarrow \wedge_{A\in DEC}[A, f(x,A)])$ is an implication from an object x. In a $NIS\ \Phi$, we usually consider $DD(\Phi)$ and each $\psi \in DD(\Phi)$. Here, the tuple of an object x in ψ_1 and the tuple of an object x in ψ_2 may be different. Namely, τ^x in ψ_1 may not exist in ψ_2. For example in Fig. 2, $\tau^1 : [Color, red] \Rightarrow [Size, s]$ in DIS_1 does not exist in DIS_4. If τ^x does not exist in ψ, we define $support(\tau^x)$=0.0 and $accuracy(\tau^x)$=0.0 in ψ. We also define

$DD(\tau^x)=\{\psi \in DD(\Phi) \mid support(\tau^x) > 0\}$.

Furthermore, if $DD(\tau^x)=DD(\Phi)$, we say τ^x is *definite*. Otherwise, we say τ^x is *indefinite*. In Fig. 2, there is no definite τ^x, and each τ^x is indefinite.

Definition 4. For a $NIS\ \Phi$, each $\psi \in DD(\Phi)$ and an implication τ^x, let $support(\tau^x, \psi)$ and $accuracy(\tau^x, \psi)$ be the support and accuracy values in ψ. We give the following definition.
(1) $minsupp(\tau^x)=Min_{\psi\in DD(\tau^x)}\{support(\tau^x, \psi)\}$.
(2) $maxsupp(\tau^x)=Max_{\psi\in DD(\tau^x)}\{support(\tau^x, \psi)\}$.
(3) $minacc(\tau^x)=Min_{\psi\in DD(\tau^x)}\{accuracy(\tau^x, \psi)\}$.
(4) $maxacc(\tau^x)=Max_{\psi\in DD(\tau^x)}\{accuracy(\tau^x, \psi)\}$.

In Definition 4, we may employ $DD(\Phi)$ instead of $DD(\tau^x)$. For a definite τ^x, $DD(\Phi)=DD(\tau^x)$ holds, so we may employ either $DD(\Phi)$ or $DD(\tau^x)$. However, if τ^x is indefinite, we directly obtain $minsupp(\tau^x)$=0.0 and $minacc(\tau^x)$=0.0, because there is a ψ where τ^x does not appear. Even though we may employ $DD(\Phi)$, however we think that $DD(\tau^x)$ is more appropriate than $DD(\Phi)$ in Definition 4. We have obtained the formula to calculate each criterion value in Definition 4. This calculation does not depend upon $|DD(\tau^x)|$.

Proposition 4. *[50, 57] It is possible to calculate the above four criterion values of* $\tau^x : [CON, \zeta] \Rightarrow [DEC, \eta]$.
Let us define OUTACC and INACC,

$OUTACC=[sup([CON,\zeta]) \setminus inf([CON,\zeta])] \setminus inf([DEC,\eta])$,

$INACC=[sup([CON,\zeta]) \setminus inf([CON,\zeta])] \cap sup([DEC,\eta])$.

If τ^x is definite, the following holds.

(D1) $minsupp(\tau^x)=|inf([CON,\zeta]) \cap inf([DEC,\eta])|/|OB|$,

(D2) $minacc(\tau^x)=\frac{|inf([CON,\zeta])\cap inf([DEC,\eta])|}{|inf([CON,\zeta])|+|OUTACC|}$,

(D3) $maxsupp(\tau^x)=|sup([CON,\zeta]) \cap sup([DEC,\eta])|/|OB|$,

(D4) $maxacc(\tau^x)=\frac{|inf([CON,\zeta])\cap sup([DEC,\eta])|+|INACC|}{|inf([CON,\zeta])|+|INACC|}$.

If τ^x is indefinite, the following holds.

(I1) $minsupp(\tau^x)=(|inf([CON,\zeta]) \cap inf([DEC,\eta])|+1)/|OB|$,

(I2) $minacc(\tau^x)=\frac{(|inf([CON,\zeta])\cap inf([DEC,\eta])|)+1}{|inf([CON,\zeta])\cup\{x\}|+|OUTACC\setminus\{x\}|}$,
(I3) $maxsupp(\tau^x)=|sup([CON,\zeta])\cap sup([DEC,\eta])|/|OB|$,
(I4) $maxacc(\tau^x)=\frac{|(inf([CON,\zeta])\cap sup([DEC,\eta]))\setminus\{x\}|+|INACC\setminus\{x\}|+1}{|inf([CON,\zeta])\cup\{x\}|+|INACC\setminus\{x\}|}$.

In Proposition 4, if τ^x is definite, $DD(\tau^x)$=$DD(\Phi)$ and $x \in inf([CON,\zeta]) \cap inf([DEC,\eta])$. However, if τ^x is indefinite, $DD(\tau^x) \subsetneq DD(\Phi)$ holds. In this case, we suppose the indefinite τ^x is selected as a candidate of τ. Namely, we think a set of definite τ^y ($y \neq x$) and τ^x. Since $x \notin inf([CON,\zeta]) \cap inf([DEC,\eta])$, we need to add +1 to both the numerator and the denominator of the formulas. In this way, we obtain Proposition 4.

Clearly, each calculation does not depend upon the size of $DD(\tau^x)$. We have also obtained the next proposition.

Proposition 5. *[59, 62] let us consider a NIS Φ and any τ^x.*
(1) There is a $\psi' \in DD(\tau^x)$ such that $support(\tau^x,\psi')$ and $accuracy(\tau^x,\psi')$ are both minimums. Namely, both $minsupp(\tau^x)$ and $minacc(\tau^x)$ occur in this ψ'. We employ a notation ψ_{min} for this ψ'.
(2) There is a $\psi'' \in DD(\tau^x)$ such that $support(\tau^x,\psi'')$ and $accuracy(\tau^x,\psi'')$ are both maximums. Namely, both $maxsupp(\tau^x)$ and $maxacc(\tau^x)$ occur in this ψ''. We employ a notation ψ_{max} for this ψ''.

In this section, we extended each rough set based concept in *DISs* to *NISs*. In *NISs*, each concept is extended to the certain concept and the possible concept. According to the previous research [15, 21, 22, 24, 25, 32, 33, 66, 67], we knew the necessity of handling *NISs*, and the survey in this section will be a solution for handling *NISs*.

9.4 An Aspect of Question-Answering and Decision Making in NISs

We specify the condition part $[CON,\zeta]$, and we can directly obtain the decision part $[DEC,\eta_i]$ with criterion values by Proposition 4. Namely, we consider each definite $\tau : [CON,\zeta] \Rightarrow [DEC,\eta_i]$ for a specified $[CON,\zeta]$, and we try to decide which $[DEC,\eta_i]$ is appropriate by using criterion values. We employ criterion values for confirming the validity of the decision making.

Let us consider the following actual execution of $[Color, red] \Rightarrow decision$ for Φ_{Table2}.

```
?-qa([[color,red]]).
--- Direct Question/Answering Mode ----------
qa3(1,[[color,red]],[2,4,5],[1,2,3,4,5],[price,low],
[1],[1,4,6]) OUTACC=[3], INACC=[1]
[1] [color,red] ==> [price,low]
MINSUPP=0.0, MINACC=0.0, MAXSUPP=0.333, MAXACC=0.5
```

```
qa3(2,[[color,red]],[2,4,5],[1,2,3,4,5],[price,high],
[2,3,5],[2,3,4,5,6]) OUTACC=[1], INACC=[3]
[2] [color,red] ==> [price,high]
MINSUPP=0.333, MINACC=0.5, MAXSUPP=0.667, MAXACC=1.0
EXEC_TIME=0.0(sec)
yes
```

Due to this execution, we know there are two implications,

τ_1: `[color,red] ==> [price,low],`

τ_2: `[color,red] ==> [price,high].`

Since the minimum values of $[Price, high]$ is more than the maximum values of $[Price, low]$, we probably make a decision of $[Price, high]$ under the condition of $[Color, red]$.

A derived *DIS* from a *NIS* in Table 4 causes $maxsupp(\tau_1)$ and $maxacc(\tau_1)$, and at the same time this derived *DIS* also causes $minsupp(\tau_2)$ and $minacc(\tau_2)$. The calculation by Proposition 4 implicitly specifies a set of derived *DISs* as a side effect.

As for the direct question-answering and decision making in *NISs*, the details are in [60, 62].

Table 4 This derived *DIS* causes both $maxsupp(\tau_1)$ and $maxacc(\tau_1)$ the maximum, and causes both $minsupp(\tau_2)$ and $minacc(\tau_2)$ the minimum.

Object	Color	Size	Weight	Price
x_1	red	any	any	low
x_2	red	any	any	high
x_3	blue	any	any	high
x_4	red	any	any	low
x_5	red	any	any	high
x_6	blue	any	any	low

9.5 Rule Generation in NISs

In Section 2, we have surveyed two types of rule generation in *DISs*. The one is the criterion based rule generation and the other is the consistency based rule generation. This section focuses on rule generation in *NISs*, and proposes an extended *Apriori* algorithm named *NIS-Apriori*. A *NIS-Apriori* based rule generation is applicable to several types of rule generation.

9.5.1 Rule Generation Tasks in a NIS

In Fig. 2, $\tau : [Color, red] \Rightarrow [Size, m]$ may occur in all objects. We employed the notation τ^x for expressing τ defined in an object x. In a *DIS*, $support(\tau^y)$=$support(\tau^x)$

and $accuracy(\tau^y)$=$accuracy(\tau^x)$ hold for each $y \in [x]$. Therefore, it is enough to consider τ. However, we need to remark the following in a *NIS*.

Remark 1. The same τ may occur from the different objects x and y, and there may be a case that τ^x satisfies the condition of rules but τ^y does not satisfy this condition. Therefore, we specify the object x in τ^x for each calculation, because $DD(\tau^x)$ depends upon x. However, we do not specify the object x for obtained rules τ. Namely, if τ^x for an object x satisfies the condition of rules, we see this τ is a rule.

In a *NIS* Φ, we have defined $DD(\Phi)$ and $DD(\tau^x)$ in Section 3. In Fig. 2, we have $|DD(\tau^1)| = 4$, $|DD(\tau^2)| = 12$, $|DD(\tau^3)| = 12$. If $DD(\tau^x)$=$DD(\Phi)$, we said this τ^x is *definite*. Otherwise, we said τ^x is *indefinite*. We give the next rule generation tasks in a *NIS*.

Specification of the Rule Generation Tasks in a ***NIS***
(The lower system) Find each implication τ such that $support(\tau^x) \geq \alpha$ and $accuracy(\tau^x) \geq \beta$ (for an object x) hold in each $\psi \in DD(\tau^x)$. We say this is a *criterion based certain rule generation* in a *NIS*. Especially, if β=1.0, we say this is a *consistency based certain rule generation* in a *NIS*.
(The upper system) Find each implication τ such that $support(\tau^x) \geq \alpha$ and $accuracy(\tau^x) \geq \beta$ (for an object x) hold in some $\psi \in DD(\tau^x)$. We say this is a *criterion based possible rule generation* in a *NIS*. Especially, if β=1.0, we say this is a *consistency based possible rule generation* in a *NIS*.

These two systems are natural extensions from rule generation tasks in a *DIS*, and we need to see that these two systems depend upon $DD(\tau^x)$. The number of derived *DISs* increases in the exponential order. However, we can solve this problem. Namely, we apply Proposition 4 and 5, and we obtain a result illustrated by Fig. 3.

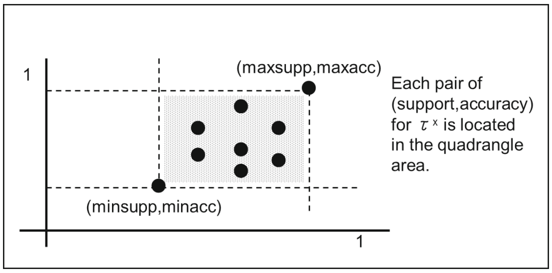


Fig. 3 A distribution of pairs (*support*,*accuracy*) for τ^x. There exists $\psi_{min} \in DD(\tau^x)$ which makes both $support(\tau^x)$ and $accuracy(\tau^x)$ the minimum. There exists $\psi_{max} \in DD(\tau^x)$ which makes both $support(\tau^x)$ and $accuracy(\tau^x)$ the maximum. We denote such quantities as *minsupp*, *minacc*, *maxsupp* and *maxacc*, respectively.

Therefore, we have the next equivalent specification.

Equivalent specification of the rule generation tasks in a NIS
(The lower system) Find each implication τ such that $minsupp(\tau^x) \geq \alpha$ and $minacc(\tau^x) \geq \beta$ for an object x (see Fig. 3).
(The upper system) Find each implication τ such that $maxsupp(\tau^x) \geq \alpha$ and $maxacc(\tau^x) \geq \beta$ for an object x (see Fig. 3).

For implementing this equivalent specification, we take the similar method as *Apriori* algorithm. We identify an item [1, 2] with a descriptor $[A, \zeta]$. We always assign $inf([ATR, \zeta_{ATR}])$ and $sup([ATR, \zeta_{ATR}])$ to each descriptor $[ATR, \zeta_{ATR}]$ by Definition 1. Since each τ^x is a conjunction of descriptors, we sequentially generate τ^x. In the lower system, we check $minsupp(\tau^x) \geq \alpha$ and $minacc(\tau^x) \geq \beta$. In the upper system, we check $maxsupp(\tau^x) \geq \alpha$ and $maxacc(\tau^x) \geq \beta$. We are calling the above steps *NIS-Apriori* algorithm. Clearly, *NIS-Apriori* does not depend upon $|DD(\tau^x)|$. The details are in [56, 57, 65].

9.5.2 Stability Factor of Rules in the Upper System

The upper system defines the possibility of rules, however this definition seems too weak. If τ_1 satisfies the constraints in only one $\psi \in DD(\tau_1^x)$, we see τ_1 as a rule in the upper system. On the other hand, If τ_2 satisfies the constraints in most $\psi \in DD(\tau_2^x)$, we also see τ_2 as a rule. We need to add another criterion to the upper system. Due to this reason, we have introduced the next *stability factor* [59, 62]
$SF(\tau)=|\{\psi \in DD(\tau) \mid \tau$ satisfies the constraints in $\psi\}|/|DD(\tau)|$
$DD(\tau)=\cup_x DD(\tau^x)$,
into the upper system. We can discriminate τ_2 from τ_1 by $SF(\tau_1)$ and $SF(\tau_2)$.

9.5.3 Current State of a Rule Generator in Prolog

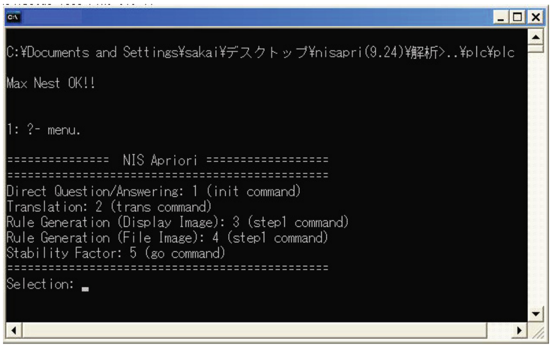
Fig.4 is the top page of the current rule generator. This is implemented in Prolog on a windows 7 PC (3.3GHz, 64bitCPU).

We are not planning to handle the large size data sets, but we are planning to handle data sets with large number of derived *DISs*. Since we can easily handle the list notation in Prolog, therefore Prolog seems suitable for this implementation.

In the lower system, the current program handles each definite τ^x, and we have not finished the program for each indefinite τ^x. In the upper system, the current program handles both.

9.5.4 An Example of Execution by a Rule Generator

Now, let us consider an exemplary NIS in Table 5, which is automatically generated by using a random number functionality.



```
C:¥Documents and Settings¥sakai¥デスクトップ¥nisapri(9.24)¥解析>..¥plc¥plc

Max Nest OK!!


1: ?- menu.

==============  NIS Apriori =================
============================================
Direct Question/Answering: 1 (init command)
Translation: 2 (trans command)
Rule Generation (Display Image): 3 (step1 command)
Rule Generation (File Image): 4 (step1 command)
Stability Factor: 5 (go command)
============================================
Selection:
```

Fig. 4 A menu of a *NIS-Apriori* based rule generator

Table 5 A Table of a *NIS* Φ_{Table5}

Object	A	B	C	D	E	F	G	H
1	{3}	{1,3,4}	{3}	{2}	{5}	{5}	{2,4}	{3}
2	{2}	{3,4}	{1,3,4}	{4}	{1,2}	{2,4,5}	{2}	{2}
3	{4,5}	{5}	{1,5}	{5}	{2}	{5}	{1,2,5}	{1}
4	{1}	{3}	{4}	{3}	{1,2,3}	{1}	{2,5}	{1,2}
5	{4}	{1}	{2,3,5}	{5}	{2,3,4}	{1,5}	{4}	{1}
6	{4}	{1}	{5}	{1}	{4}	{2,4,5}	{2}	{1,2,3}
7	{2}	{4}	{3}	{4}	{3}	{2,4,5}	{4}	{1,2,3}
8	{4}	{5}	{4}	{2,3,5}	{5}	{3}	{1,2,3}	{1,2,3}
9	{2}	{3}	{5}	{3}	{1,3,5}	{4}	{2}	{3}
10	{4}	{2}	{1}	{5}	{2}	{4,5}	{3}	{1}

The following is the actual data for Φ_{Table5}. The prototype system in Prolog can handle any data set in the following syntax.

```
object(10,8). /* #object=10, #attribute=8 */
support(0.2). /* a constraint: support is more than 0.2 */
accuracy(0.8). /* a constraint: accuracy is more than 0.8 */
decision(8). /* the decision attribute 8th */
attrib(1,a,5,[1,2,3,4,5]). /* an attribute 1, name, values */
attrib(1,b,5,[1,2,3,4,5]).
    :    :    :
attrib(1,h,5,[1,2,3,4,5]).
total_cases(7346640384,nointerval). /* the number of DD(Φ_{Table_5}) */
data(1,[3,[1,3,4],3,2,5,5,[2,4],3]). /* data */
data(2,[2,[3,4],[1,3,4],4,[1,2],[2,4,5],2,2]).
    :    :    :
data(10,[4,2,1,5,2,[4,5],3,1]).
```

According to the values of *support*, this data set is at first translated to the internal data. The following is a part of it:

```
lower(1,2,[a,2],[2,7,9],[2,7,9]).
lower(1,4,[a,4],[5,6,8,10],[3,5,6,8,10]).
lower(2,1,[b,1],[5,6],[1,5,6]).
lower(2,3,[b,3],[4,9],[1,2,4,9]).
upper(2,4,[b,4],[7],[1,2,7]).
```

The fourth and fifth arguments mean the minimum *pe*-class and the maximum *pe*-class for a descriptor. For each attribute, if attribute value of an object x is definite, x is added to the fourth and fifth arguments of the descriptor. If attribute value is indefinite, x is added to the fifth argument of the related descriptors.

The following is a process of `step2` command in rule generation. In reality, we sequentially employ `step1`, `step2`, $\cdots$, `step5` commands. `step1` generates rules whose condition part consist of a descriptor, and `step2` does rules whose condition part consist of a conjunction of two descriptors. `step3`, `step4` and `step5` are in the same manner. The computational complexity of these commands is almost the same as *Apriori* algorithm. A rule $\tau_1 : [d,5]\&[e,2] \Rightarrow [h,1]$ in the lower system satisfies $support(\tau_1^3) \geq 0.2$ and $accuracy(\tau_1^3) \geq 0.8$ in each of 7346640384 derived *DISs*. A rule $\tau_2 : [f,5]\&[g,2] \Rightarrow [h,3]$ in the upper system is supported by only 4% derived *DISs*, which is calculated by `sf` command. Namely, τ_2 will not be a reliable implication. The complexity of `sf` command depends upon pairs of *possible equivalence classes* (pe_{CON},pe_{DEC}) [59, 61], so this program may not be effective.

```
?-step2.
===== Lower System =========================================
[1] MINSUPP=0.2, MINACC=0.6666666667
[2] MINSUPP=0.1, MINACC=1.0
[3] MINSUPP=0.2, MINACC=1.0 [d,5]&[e,2]==>[h,1] [3,10]
The Rest Candidates: [[[1,4],[4,5],[8,1]]]
(Lower System Terminated)
===== Upper System =========================================
[1] MAXSUPP=0.0, MAXACC=0.0
      :    :    :
[150] MAXSUPP=0.2, MAXACC=1.0 [f,5]&[g,2]==>[h,3] [1,6]
The Rest Candidates:[[[1,2],[6,4],[8,2]],[[2,3],[7,2],[8,2]],
[[3,3],[6,5],[8,1]],[[3,3],[6,5],[8,2]],[[3,5],[7,2],[8,1]],
[[4,3],[7,2],[8,1]],[[4,3],[7,2],[8,2]],[[6,4],[7,2],[8,2]]]
(Next Candidates are Remained)
EXEC_TIME=0.0 (sec)
yes

?-sf([[f,5],[g,2]],[h,3]).
[1] PE_CON:[], PE_DEC:[1,9], Intersection:[]
[2] PE_CON:[], PE_DEC:[1,9,6], Intersection:[]
```

```
       :    :    :
 [128] PE_CON:[1,2,3,6], PE_DEC:[1,9,6,7,8], Intersection:[1,6]
 Number of DISs in This Case:1
 DENO=810, NUME=36
 SF=0.04444444444=(36/810)
 EXEC_TIME=1.0 (sec)
 yes
```

9.5.5 An Application to Other Types of Rule Generation

A *NIS-Apriori* based rule generator can handle criterion based certain and possible rules generation. This section considers how this rule generator is applied to some other types of rule generation.

9.5.5.1 Case 1: Criterion Based Rule Generation in a DIS

We can easily express a data set of a *DIS*. We employed a list notation for expressing non-deterministic information, for example,
`data(10,[4,2,1,5,2,[4,5],3,1]).`
The 6th attribute value [4,5] is non-deterministic, and other attribute values are deterministic (or definite).

In reality, if all attribute values are definite, the minimum and the maximum equivalence classes for each descriptors are the same set. The lower system and the upper system generate the same rules. Namely, *NIS-Apriori* can simulates criterion based rule generation in a *DIS*.

9.5.5.2 Case 2: Consistency Based Rule Generation in a DIS

If we specify $support(0.0)$, $accuracy(1.0)$ and a *DIS* in the syntax of a *NIS*, we obtain consistency based rules in a *DIS*. Namely, *NIS-Apriori* can take a role of the discernibility function method, and can simulates consistency based rule generation in a *DIS*.

9.5.5.3 Case 3: Rule Generation in a DIS with Missing Values

There are some important research on a *DIS* with missing values or *Incomplete Information Systems*. For example, *LERS* system [12, 13] by Grzymała-Busse and a framework of reduction based rule generation [21, 22] by Kryszkiewicz are well known. In both research, some interpretations are assumed for missing values, and rule generation methods are investigated.

In a *NIS-Apriori* based rule generator is also applicable to *DISs* with missing values. In most of tables with categorical data, each domain of attribute values is a finite set. Since any missing value is an element of this finite domain, we replace each missing value with this domain. Then, we can apply our rule generator to this adjusted *NIS*. In our framework, the interpretation of missing values seems clear, but instead we needed to face with the exponential order problem about the number of derived *DISs*. However, we have solved this exponential order problem by *NIS-Apriori* algorithm. Thus, we can also apply our rule generator to *DISs* with missing values.

9.5.5.4 Case 4: Consistency Based Rule Generation in a NIS

We at first survey consistency based rule generation by a discernibility function for Φ_{Table5}, then we show *NIS-Apriori* seems much more powerful than the discernibility function method.

In a discernibility function method, we fix a pair of a decision attribute and an attribute value ($[DEC, val_{DEC}]$), and define a target set $X \subseteq U$ (U is a set of all objects). Then, we try to obtain minimal set of descriptors ($\wedge_i[A_i, val_i]$) which discriminate X from $U - X$. Finally, an implication $\wedge_i[A_i, val_i] \Rightarrow [DEC, val_{DEC}]$ is a minimal consistency based rules. The details are in [53, 55].

The following is an example of execution for Φ_{Table5}. The decision part is specified by $[h, 1]$. In this program, we are using the ordinal number instead of an attribute name, namely $[h, 1]$ is expressed by $[8, 1]$ in this program. We know a target set is $\{3, 5, 10\}$ by `init` command, and the extended discernibility function depending upon an object 3 is displayed by `disfunc(3,M)` command. Since $1 \notin \{3, 5, 10\}$ holds, it is necessary to discriminate 1 from $\{3, 5, 10\}$. If we employ either a descriptor $[2, 5]$, $[4, 5]$ or $[5, 2]$, an object 1 is discriminated. In reality, there may be lots of minimal solutions of the discernibility function, so we interactively obtain solutions by `solall(3)` command. Here, we specified a descriptor $[2, 5]$ ($[b, 5]$), and we obtained two consistency based rules (rule 1) and (rule 2) with $[2, 5]$. If we select other descriptor, this program responds other rules.

```
?-init.
Rs File:data.rs.
DECLIST:<inf=[3,5,10]>
Certain Rules come from [3,5,10]
EXEC_TIME=0.0 (sec)
yes
?-disfunc(3,M).
M=[[1,[2,5],[4,5],[5,2]],[2,[2,5],[4,5]],[4,[2,5],[4,5],[6,5]],
[6,[2,5],[4,5],5,2]],[7,[2,5],[4,5],[5,2]],[8,[5,2],[6,5]],
[9,[2,5],[4,5],[5,2],[6,5]]]
 yes
?-solall(3).
Input a number of Related Descriptors to Start Exhaustive Search:3.
Exhaustive Search for less than 8 Cases !!
[Loop:1]
  Discernibility Function without Core Descriptors:
  [[1,[2,5],[4,5],[5,2]],[2,[2,5],[4,5]],[4,[2,5],[4,5],[6,5]],
  [6,[2,5],[4,5],[5,2]],[7,[2,5],[4,5],[5,2]],[8,[5,2],[6,5]],
  [9,[2,5],[4,5],[5,2],[6,5]]]
  Currently Selected Descriptors:[]
  Descriptors:[[2,5],[4,5],[5,2],[6,5]]
  Select a Descriptor:[2,5].
  Currently Selected Descriptors:[[2,5]]
  Revised Discernibility Function:[[8,[5,2],[6,5]]]
  Common Descriptors:[[5,2],[6,5]]
  [2,5]&[5,2]==>[8,1]   /* [b,5]&[e,2]==>[h,1] (rule 1) */
  [17496/17496(=324/324,54/54),Definite,Globally Consistent]
  [2,5]&[6,5]==>[8,1]   /* [b,5]&[f,5]==>[h,1] (rule 2) */
  [34992/34992(=648/648,54/54),Definite,Globally Consistent]
yes
```

We can easily obtain the same results by using a *NIS-Apriori* based rule generator. The following is an execution specified by *support*(0.0) and *accuracy*(1.0).

```
##### 1st Step ####################
===== Lower System ==========================================
[9] [a,3]==>[h,3] (0.1,1.0) [1]
[16] [b,2]==>[h,1] (0.1,1.0) [10]
The Rest Candidates:[[[1,2],[8,2]],[[1,2],[8,3]],[[1,4],[8,1]],
      :    :    :
[[7,2],[8,3]],[[7,3],[8,1]],[[7,4],[8,1]]]
(Next Candidates are Remained)
===== Upper System ==========================================
[1] [a,1]==>[h,1] (0.1,1.0) [4]
      :    :    :
[104] [g,5]==>[h,2] (0.1,1.0) [4]
The Rest Candidates:[[[1,2],[8,1]],[[1,2],[8,2]],[[1,2],[8,3]],
      :    :    :
EXEC_TIME=0.0 (sec)
yes

##### 2nd Step ####################
===== Lower System ==========================================
[12] [a,2]&[c,5]==>[h,3] (0.1,1.0) [9]
      :    :    :
[129] [b,5]&[e,2]==>[h,1] (0.1,1.0) [3]
[134] [b,5]&[f,5]==>[h,1] (0.1,1.0) [3]
      :    :    :
[235] [d,5]&[e,2]==>[h,1] (0.2,1.0) [3,10]
[240] [d,5]&[f,5]==>[h,1] (0.1,1.0) [3]
      :    :    :
```

In this execution, we obtained (rule 1) as [129] and (rule 2) as [134]. Furthermore, we obtained consistency based rules [235] and [240] related to an object 3. We have previously realized a program depending upon a discernibility function, however we now think *NIS-Apriori* based rule generator is much more useful.

We have also applied to some data sets, for example mammographic.csv (the object size is 150, the attribute size is 6, the number of derived *DISs* is about 10^{46}) in UCI repository [10] and examined *NIS-Apriori* in Prolog works well.

9.6 Perspective of RNIA in Machine Learning

This section considers the next research related to *RNIA*, and shows the perspective of *RNIA* with respect to machine learning.

9.6.1 Handling of Inexact Data

We are now advancing from *NISs* to *Incomplete Information Databases* (*IIDs*) [24, 25] by Lipski. In *NISs*, each attribute value is implicitly a discrete value, and we could easily define $DD(\Phi)$. However, in an *IID* Γ, intervals are employed for handling continuous values like in Table 6. As for intervals, theoretically there may be uncountable number of subsets, therefore the definition of $DD(\Gamma)$ will not be easy. In reality, Lipski coped with mathematical foundations of the question answering systems in *IIDs* and proposed some software solutions. Therefore, $DD(\Gamma)$ was not defined in [24, 25].

Table 6 An exemplary example of Incomplete Information Database Γ_{Table6} [24].

OB	Age	Dept#	Hireyear	Sal
x_1	[60,70]	$\{1,\cdots,5\}$	$\{70,\cdots,75\}$	$\{10000\}$
x_2	[52,56]	$\{2\}$	$\{72,\cdots,76\}$	(0,20000]
x_3	$\{30\}$	$\{3\}$	$\{70,71\}$	$(0,\infty)$
x_4	$(0,\infty)$	$\{2,3\}$	$\{70,\cdots,74\}$	$\{22000\}$
x_5	$\{32\}$	$\{4\}$	$\{75\}$	$(0,\infty)$

We are considering the concept of the *resolution* of data, and we are now coping with the definition of $DD(\Gamma)$ and rule generation from Γ [61].

As for the numerical data, we have also proposed a method by using the *numerical pattern* [52, 58]. If we handle numerical values, the number of attribute values may be too large. In Table 7, we may need to consider 1001 descriptors for AVR, namely $[AVR, 0.000]$, $[AVR, 0.001]$, $\cdots$, $[AVR, 1.000]$.

Now, we consider information incompleteness for numerical values. Information incompleteness is a relative concept. For example, let us consider number π. The value 3.14 will be enough for students, but it will be too simple for researcher. This example will also be related to granularity and granular computing [40, 71].

Table 7 Players' Batting Data in Baseball Games, AVG: Batting Average, SF&SH: Sacrifice Flies and Hits, SB: Stolen Bases, OBP: On-Base Percentage, SLG: Slugging Percentage [58].

Player	AVG	SF&SH	SB	OBP	SLG
p_1	0.322	0	03	0.397	0.553
p_2	0.312	1	07	0.391	0.430
p_3	0.309	0	03	0.390	0.557
p_4	0.300	0	01	0.307	0.556
p_5	0.273	0	05	0.326	0.467

We introduced two symbols @ and #, which represent numeric from 0 to 9. A *numerical pattern* is a sequence of @ and #, for example @@@, @@#, @##, @@.@ and @#.#. Here, @ denotes a significant figure and # denotes a figure, which we do not care. For example, AVG values of p_3 and p_4 are 0.309 and 0.300, respectively. These two values are different according to a numerical pattern 0.@@@, but these two values are the same according to a numerical pattern 0.@@#. By means of introducing such numerical patterns, we can explicitly define the meaningful figures in numerical values.

In *Apriori* algorithm, at first we give the value α for *support*, then we pick up candidates of descriptors. Therefore, if we employ the numerical pattern, we do not have to do the discretization of numerical values as the pre-processing.

We are now planning to handle other types of data like intervals [16, 17, 43, 74], and are now reforming the unified framework of rule generation with non-deterministic information, intervals and the numerical patterns.

9.6.2 Learning a DIS from a NIS by Constraints

In the basic chart in Fig. 2, we considered $DD(\Phi_{Fig.2})$ and defined the certainty and the possibility. Namely, the issue was to characterize the worst case and the best case. Now, we focus on another aspect. We add some constraints, like rules in the lower system or data dependency to a *NIS* Φ, then some attribute values are restricted by the constraints. As a result, we can reduce the information incompleteness in *NISs*. The purpose becomes to obtain $\psi^{actual} \in DD(\Phi)$.

Let us consider Φ_{Table2} in Section 3.1 and Section 4. If we agree with the next implication

τ_1: `[color,red] ==> [price,low]`

and we see this is a constraint in Φ_{Table2}, as a side effect we have a derived *DIS* $\psi_{Table4} \in DD(\Phi_{Table2})$. If we agree with the other implication

τ_2: `[color,red] ==> [price,high],`

we have the next Table 8 as a restricted Φ_{Table2}.

Table 8 A restricted Φ_{Table2} by the constraint τ_2.

Object	Color	Size	Weight	Price
x_1	{blue,green}	{small}	{light,heavy}	{low}
x_2	{red}	{small,medium}	{light,heavy}	{high}
x_3	{red}	{small,medium}	{light}	{high}
x_4	{red}	{medium}	{heavy}	{high}
x_5	{red}	{small,medium,large}	{heavy}	{high}
x_6	{blue,green}	{large}	{heavy}	{low}

Furthermore in Table 8, let us consider another constraint, i.e., the data dependency from *Size* to *Weight*. In order to keep the data dependency, some attribute values are automatically fixed. Since the tuples of x_4 and x_6 are definite, we need to select attribute values which are consistent to x_4 and x_6. At first, we need to select

$small \in g(x_3, Size)$ and $light \in g(x_1, Weight)$.

Then, we need to select

$\{medium, large\} \subseteq g(x_5, Size)$ and

either a pair $(small, light)$ or $(medium, heavy)$

from object x_2. Like this, we obtain more restricted Table 9.

Table 9 A restricted Φ_{Table8} by the dependency from *Size* to *Weight*.

Object	Color	Size	Weight	Price
x_1	{blue,green}	{small}	{light}	{low}
x_2	{red}	{small}	{light}	{high}
x_3	{red}	{small}	{light}	{high}
x_4	{red}	{medium}	{heavy}	{high}
x_5	{red}	{medium,large}	{heavy}	{high}
x_6	{blue,green}	{large}	{heavy}	{low}

Such procedure is now just an idea, and we need to cope with this procedure algorithmically. Probably, such procedure will be a learning of a *NIS* like backpropergation in the neural net [73].

9.6.3 Table Data and Logical Data in Machine Learning

In this sub section, let us consider two frameworks, namely rule generation in a table data and rule generation in a logical data. In this paper, we followed the rough sets based framework, and developed rule generation in a table data. This will correspond to machine learning in a table data.

However, there is another framework of machine learning, namely *Inductive Logic Programming* (*ILP*) [75]. In *ILP*, hypothesis is generated from positive examples, negative examples and background knowledge.

We are familiar with the framework of logic programming, and most programs in this paper are implemented in Prolog. We have formerly followed logic programming, and tried to define Prolog with rough sets based concepts [44, 47].

The biggest difference of two frameworks is in the existence of variables. Rough set theory does not handle variables, and the table data consists of constant symbols. However, there exist variables in *ILP* and logic programming. Since variables are quantified universally in *ILP*, the rules in the *ILP* cover more general cases, and it seems difficult to obtain such general rules. On the other hand, there is no variable in rough sets, and rules do not contain variables. Each rule consists of descriptors (or proposition). Such rules may be less general than rules in the *ILP*, but it seems easier to obtain rules. The combination rough sets and *ILP* (*Rough ILP*) will be another research area.

9.7 Concluding Remarks

This paper surveyed the foundations of *Rough Non-deterministic Information Analysis* (*RNIA*) including *DISs* and *NISs*. *RNIA* will be a good framework for handling the information incompleteness. We focused on rough sets based aspects, question-answering in *NISs* and rule generation in *NISs*.

In the next steps, we need to introduce inexact types of data into *NISs*, and we need to think the algorithmic aspect of learning a *DIS* from a *NIS*. We also need to consider the relation between rough sets based rule generation and rule generation in a framework of logic.

Acknowledgements. The authors would be grateful to Professor Lakhmi Jain, Professor Sheela Ramanna and Professor Robert Howlett for their kind suggestion. The authors would thank Professor Dominik Ślęzak for his discussion about *RNIA*.

This work is supported by the Grant-in-Aid for Scientific Research (C) (No. 16500176, No.18500214, No.22500204), Japan Society for the Promotion of Science.

References

1. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules in large databases. In: Bocca, J.B., Jarke, M., Zaniolo, C. (eds.) VLDB 1994, pp. 487–499. Morgan Kaufmann (1994)
2. Agrawal, R., Mannila, H., Srikant, R., Toivonen, H., Verkamo, A.I.: Fast discovery of association rules. In: Advances in Knowledge Discovery and Data Mining, pp. 307–328. AAAI/MIT Press (1996)
3. Bazan, J., Nguyen, H.S., Nguyen, S.H., Synak, P., Wróblewski, J.: Rough set algorithms in classification problem. In: Rough Set Methods and Applications. STUDFUZZ, vol. 56, pp. 49–88. Springer (2000)
4. Blaszczynski, J., Greco, S., Słowiński, R.: Multi-criteria classification - a new scheme for application of dominance-based decision rules. European Journal of Operational Research 181(3), 1030–1044 (2007)
5. Ceglar, A., Roddick, J.F.: Association mining. ACM Computing Survey 38(2) (2006)

6. Chmielewski, M.R., Grzymała-Busse, J.W.: Global discretization of continuous attributes as preprocessing for machine learning. International Journal of Approximate Reasoning 15(4), 319–331 (1996)
7. Codd, E.F.: A relational model of data for large shared data banks. Communication of the ACM 13(6), 377–387 (1970)
8. Cornelis, C., Jensen, R., Martín, G.H., Ślęzak, D.: Attribute selection with fuzzy decision reducts. Information Sciences 180(2), 209–224 (2010)
9. Demri, S., Orłowska, E.: Incomplete Information: Structure, Inference, Complexity. Monographs in Theoretical Computer Science. An EATCS Series. Springer (2002)
10. Frank, A., Asuncion, A.: UCI Machine Learning Repository, School of Information and Computer Science, University of California, Irvine, CA (2010), http://mlearn.ics.uci.edu/MLRepository.html
11. Greco, S., Matarazzo, B., Słowiński, R.: Granular computing and data mining for ordered data: The dominance-based rough set approach. In: Meyers, R.A. (ed.) Encyclopedia of Complexity and Systems Science, pp. 4283–4305. Springer (2009)
12. Grzymała-Busse, J.W.: A new version of the rule induction system LERS. Fundamenta Informaticae 31(1), 27–39 (1997)
13. Grzymała-Busse, J.W., Werbrouck, P.: On the best search method in the lem1 and lem2 algorithms. In: Orłowska, E. (ed.) Incomplete Information: Rough Set Analysis. STUDFUZZ, vol. 13, pp. 75–91. Springer (1998)
14. Grzymała-Busse, J.W., Stefanowski, J.: Three discretization methods for rule induction. International Journal of Intelligent Systems 16(1), 29–38 (2001)
15. Grzymała-Busse, J.W.: Data with missing attribute values: Generalization of indiscernibility relation and rule induction. Transactions on Rough Sets 1, 78–95 (2004)
16. Huynh, V.N., Nakamori, Y., Ono, H., Lawry, J., Kreinovich, V., Nguyen, H.T. (eds.): Interval / Probabilistic Uncertainty and Non-Classical Logics. AISC, vol. 46. Springer (2008)
17. Huynh, V.N., Nakamori, Y., Hu, C., Kreinovich, V.: On decision making under interval uncertainty: A new justification of hurwicz optimism-pessimism approach and its use in group decision making. In: ISMVL 2009, pp. 214–220. IEEE Computer Society (2009)
18. Inuiguchi, M., Yoshioka, Y., Kusunoki, Y.: Variable-precision dominance-based rough set approach and attribute reduction. International Journal of Approximate Reasoning 50(8), 1199–1214 (2009)
19. Komorowski, J., Pawlak, Z., Polkowski, L., Skowron, A.: Rough sets: a tutorial. In: Pal, S.K., Skowron, A. (eds.) Rough Fuzzy Hybridization: A New Method for Decision Making, pp. 3–98. Springer (1999)
20. Kryszkiewicz, M., Rybinski, H.: Computation of reducts of composed information systems. Fundamenta Informaticae 27(2-3), 183–195 (1996)
21. Kryszkiewicz, M.: Rough set approach to incomplete information systems. Information Sciences 112(1-4), 39–49 (1998)
22. Kryszkiewicz, M.: Rules in incomplete information systems. Information Sciences 113(3-4), 271–292 (1999)
23. Leung, Y., Fischer, M.M., Wu, W.Z., Mi, J.S.: A rough set approach for the discovery of classification rules in interval-valued information systems. International Journal of Approximate Reasoning 47(2), 233–246 (2008)
24. Lipski, W.: On semantic issues connected with incomplete information databases. ACM Transactions on Database Systems 4(3), 262–296 (1979)
25. Lipski, W.: On databases with incomplete information. Journal of the ACM 28(1), 41–70 (1981)

26. Murai, T., Resconi, G., Nakata, M., Sato, Y.: Operations of zooming in and out on possible worlds for semantic fields. In: Damiani, L.J.E., Howlett, R.J., Ichalkaranje, N. (eds.): Knowledge-Based Intelligent Information Engineering Systems and Allied Technologies. KES 2004, Frontiers in Artificial Intelligence and Applications, vol. 82, pp. 1083–1087. IOS Press (2002)
27. Nakamura, A.: A rough logic based on incomplete information and its application. International Journal of Approximate Reasoning 15(4), 367–378 (1996)
28. Nakamura, A., Tsumoto, S., Tanaka, H., Kobayashi, S.: Rough set theory and its applications. Journal of Japanese Society for Artificial Intelligence 11(2), 209–215 (1996)
29. Nakata, M., Sakai, H.: Rough-set-based approaches to data containing incomplete information: possibility-based cases. In: Nakamatsu, K., Abe, J.M. (eds.) Advances in Logic Based Intelligent Systems, Frontiers in Artificial Intelligence and Applications, vol. 132, pp. 234–241. IOS Press (2005)
30. Nakata, M., Sakai, H.: Lower and upper approximations in data tables containing possibilistic information. Transactions on Rough Sets 7, 170–189 (2007)
31. Nakata, M., Sakai, H.: Applying Rough Sets to Information Tables Containing Possibilistic Values. Transactions on Computational Science 2, 180–204 (2008)
32. Orłowska, E., Pawlak, Z.: Representation of nondeterministic information. Theoretical Computer Science 29(1-2), 27–39 (1984)
33. Orłowska, E. (ed.): Incomplete Information: Rough Set Analysis. STUDFUZZ, vol. 13. Springer (1998)
34. Orłowska, E.: Introduction: What you always wanted to know about rough sets. In: Orłowska, E. (ed.) Incomplete Information: Rough Set Analysis. STUDFUZZ, vol. 13, pp. 1–20. Springer (1998)
35. Orłowska, E.: A roadmap of information logics and information algebras inspired by rough sets. In: Plenary Workshop in Rough Sets, Fuzzy Sets, Data Mining and Granular Computing (2005)
36. Pawlak, Z.: Information systems theoretical foundations. Information Systems 6(3), 205–218 (1981)
37. Pawlak, Z.: Systemy Informacyjne: Podstawy teoretyczne. Wydawnictwa Naukowo-Techniczne Publishers (1983)
38. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning About Data. Kluwer Academic Publishers (1991)
39. Pawlak, Z.: Some Issues on Rough Sets. In: Peters, J.F., Skowron, A., Grzymała-Busse, J.W., Kostek, B.z., Świniarski, R.W., Szczuka, M.S. (eds.) Transactions on Rough Sets I. LNCS, vol. 3100, pp. 1–58. Springer, Heidelberg (2004)
40. Pedrycz, W., Skowron, A., Kreinovich, V. (eds.): Handbook of Granular Computing. Wiley (2008)
41. Polkowski, L., Skowron, A. (eds.): Rough Sets in Knowledge Discovery 1: Methodology and Applications. STUDFUZZ, vol. 18. Springer (1998)
42. Polkowski, L., Skowron, A. (eds.): Rough Sets in Knowledge Discovery 2: Applications, Case Studies and Software Systems. STUDFUZZ, vol. 19. Physica-Verlag (1998)
43. Quinlan, J.R.: Improved use of continuous attributes in C4.5. The Journal of Artificial Intelligence Research 4, 77–90 (1996)
44. Sakai, H.: On a Framework for logic programming with incomplete information. Fundamenta Informaticae 19(3/4), 223–234 (1993)
45. Sakai, H., Okuma, A.: An Algorithm for Finding Equivalence Relations from Tables with Non-Deterministic Information. In: Zhong, N., Skowron, A., Ohsuga, S. (eds.) RSFDGrC 1999. LNCS (LNAI), vol. 1711, pp. 64–73. Springer, Heidelberg (1999)

46. Sakai, H., Okuma, A.: An Algorithm for Checking Dependencies of Attributes in a Table with Non-Deterministic Information: A Rough Sets Based Approach. In: Mizoguchi, R., Slaney, J.K. (eds.) PRICAI 2000. LNCS(LNAI), vol. 1886, pp. 219–229. Springer, Heidelberg (2000)
47. Sakai, H., Okuma, A.: On a theorem prover for variational logic programs with functors setu and sets. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 8(1), 73–92 (2000)
48. Sakai, H.: Effective procedures for handling possible equivalence relations in non-deterministic information systems. Fundamenta Informaticae 48(4), 343–362 (2001)
49. Sakai, H.: Effective procedures for data dependencies in information systems. In: Inuiguchi, M., Tsumoto, S., Hirano, S. (eds.) Rough Set Theory and Granular Computing. STUDFUZZ, vol. 125, pp. 167–176. Springer (2003)
50. Sakai, H., Okuma, A.: Basic Algorithms and Tools for Rough Non-Deterministic Information Analysis. In: Peters, J.F., Skowron, A., Grzymała-Busse, J.W., Kostek, B.z., Świniarski, R.W., Szczuka, M.S. (eds.) Transactions on Rough Sets I. LNCS, vol. 3100, pp. 209–231. Springer, Heidelberg (2004)
51. Sakai, H.: Possible Equivalence Relations and their Application to Hypothesis Generation in Non-Deterministic Information Systems. In: Peters, J.F., Skowron, A., Dubois, D., Grzymała-Busse, J.W., Inuiguchi, M., Polkowski, L. (eds.) Transactions on Rough Sets II. LNCS, vol. 3135, pp. 82–106. Springer, Heidelberg (2004)
52. Sakai, H., Murai, T., Nakata, M.: On a Tool for Rough Non-Deterministic Information Analysis and its Perspective for Handling Numerical Data. In: Torra, V., Narukawa, Y., Miyamoto, S. (eds.) MDAI 2005. LNCS (LNAI), vol. 3558, pp. 203–214. Springer, Heidelberg (2005)
53. Sakai, H., Nakata, M.: Discernibility Functions and Minimal Rules in Non-Deterministic Information Systems. In: Ślęzak, D., Wang, G., Szczuka, M.S., Düntsch, I., Yao, Y. (eds.) RSFDGrC 2005. LNCS (LNAI), vol. 3641, pp. 254–264. Springer, Heidelberg (2005)
54. Sakai, H.: On a Rough Sets Based Data Mining Tool in Prolog: An Overview. In: Umeda, M., Wolf, A., Bartenstein, O., Geske, U., Seipel, D., Takata, O. (eds.) INAP 2005. LNCS (LNAI), vol. 4369, pp. 48–65. Springer, Heidelberg (2006)
55. Sakai, H., Nakata, M.: An application of discernibility functions to generating minimal rules in non-deterministic information systems. Journal of Advanced Computational Intelligence and Intelligent Informatics 10(5), 695–702 (2006)
56. Sakai, H., Nakata, M.: On Possible Rules and Apriori Algorithm in Non-Deterministic Information Systems. In: Greco, S., Hata, Y., Hirano, S., Inuiguchi, M., Miyamoto, S., Nguyen, H.S., Słowiński, R. (eds.) RSCTC 2006. LNCS (LNAI), vol. 4259, pp. 264–273. Springer, Heidelberg (2006)
57. Sakai, H., Ishibashi, R., Koba, K., Nakata, M.: Rules and Apriori Algorithm in Non-Deterministic Information Systems. In: Peters, J.F., Skowron, A., Rybiński, H. (eds.) Transactions on Rough Sets IX. LNCS, vol. 5390, pp. 328–350. Springer, Heidelberg (2008)
58. Sakai, H., Koba, K., Nakata, M.: Rough sets based rule generation from data with categorical and numerical values. Journal of Advanced Computational Intelligence and Intelligent Informatics 12(5), 426–434 (2008)
59. Sakai, H., Hayashi, K., Nakata, M., Ślęzak, D.: The Lower System, the Upper System and Rules with Stability Factor in Non-Deterministic Information Systems. In: Sakai, H., Chakraborty, M.K., Hassanien, A.E., Ślęzak, D., Zhu, W. (eds.) RSFDGrC 2009. LNCS(LNAI), vol. 5908, pp. 313–320. Springer, Heidelberg (2009)

60. Sakai, H., Hayashi, K., Kimura, H., Nakata, M.: An Aspect of Decision Making in Rough Non-Deterministic Information Analysis. In: Nakamatsu, K., Phillips-Wren, G., Jain, L.C., Howlett, R.J. (eds.) New Advances in Intelligent Decision Technologies. SCI, vol. 199, pp. 527–536. Springer, Heidelberg (2009)
61. Sakai, H., Nakata, M., Ślęzak, D.: Rule Generation in Lipski's Incomplete Information Databases. In: Szczuka, M., Kryszkiewicz, M., Ramanna, S., Jensen, R., Hu, Q. (eds.) RSCTC 2010. LNCS(LNAI), vol. 6086, pp. 376–385. Springer, Heidelberg (2010)
62. Sakai, H., Okuma, H., Nakata, M., Ślęzak, D.: Stable rule extraction and decision making in rough non-deterministic information analysis. International Journal of Hybrid Intelligent Systems 8(1), 41–57 (2011)
63. Skowron, A., Rauszer, C.: The discernibility matrices and functions in information systems. In: Słowiński, R. (ed.) Intelligent Decision Support - Handbook of Advances and Applications of the Rough Set Theory, pp. 331–362. Kluwer Academic Publishers (1992)
64. Ślęzak, D.: Rough sets and functional dependencies in data: Foundations of association reducts. Transactions on Computational Science 5, 182–205 (2009)
65. Ślęzak, D., Sakai, H.: Automatic Extraction of Decision Rules from Non-Deterministic Data Systems: Theoretical Foundations and SQL-Based Implementation. In: Ślęzak, D., Kim, T.-h., Zhang, Y., Ma, J., Chung, K.-i. (eds.) DTA 2009. CCIS, vol. 64, pp. 151–162. Springer, Heidelberg (2009)
66. Stefanowski, J., Tsoukiàs, A.: On the Extension of Rough Sets Under Incomplete Information. In: Zhong, N., Skowron, A., Ohsuga, S. (eds.) RSFDGrC 1999. LNCS (LNAI), vol. 1711, pp. 73–82. Springer, Heidelberg (1999)
67. Stefanowski, J., Tsoukiàs, A.: Incomplete information tables and rough classification. Computational Intelligence 17(3), 545–566 (2001)
68. Tsumoto, S.: Knowledge discovery in clinical databases and evaluation of discovered knowledge in outpatient clinic. Information Sciences 124(1-4), 125–137 (2000)
69. Yang, X., Yu, D., Yang, J., Wei, L.: Dominance-based rough set approach to incomplete interval-valued information system. Data & Knowledge Engineering 68(11), 1331–1347 (2009)
70. Yao, Y(Y.Y.), Liau, C.-J., Zhong, N.: Granular Computing Based on Rough Sets, Quotient Space Theory, and Belief Functions. In: Zhong, N., Raś, Z.W., Tsumoto, S., Suzuki, E. (eds.) ISMIS 2003. LNCS (LNAI), vol. 2871, pp. 152–159. Springer, Heidelberg (2003)
71. Zadeh, L.A.: Toward a theory of fuzzy information granulation and its centrality in human reasoning and fuzzy logic. Fuzzy Sets and Systems 90(2), 111–127 (1997)
72. Ziarko, W.: Variable precision rough set model. Journal of Computer and System Sciences 46(1), 39–59 (1993)
73. Backpropagation, http://en.wikipedia.org/wiki/Backpropagation
74. Confidence Interval, http://en.wikipedia.org/wiki/Confidence_interval
75. ILP, http://en.wikipedia.org/wiki/Inductive_logic_programming
76. Rough Set Software, Bulletin of Int'l Rough Set Society 2(1), 15–46 (1998)

Appendix

Algorithm 1
Input: A *DIS* with a function f and a set $ATR \subseteq AT$.
Output: Two arrays $head[i]$ and $succ[i]$ $(1 \leq i \leq |OB|)$.

```
begin
    for i:=1 to |OB| do begin head[i]:=i; succ[i]:=0 end;
    for i:=1 to (|OB|-1) do
       if head[i]=i then
          begin pre:=i;
                for j:=i+1 to |OB| do
                   if ($f(x_j,a)$=$f(x_i,a)$ for $\forall a \in ATR$) then
                   begin head[j]:=i; succ[pre]:=j; pre:=j end
          end
end.
```

Algorithm 2
Input: Two arrays for attributes A and B respectively, i.e.,
$head_A[i]$, $succ_A[i]$, $head_B[i]$ and $succ_B[i]$ $(1 \leq i \leq |OB|)$.
Output: Two arrays for $A \cup B$, $head_{A\cup B}[i]$ and $succ_{A\cup B}[i]$ $(1 \leq i \leq |OB|)$.

```
begin
    for i:=1 to |OB| do
       begin head$_{A\cup B}$[i]:=i; succ$_{A\cup B}$[i]:=0 end;
    for i:=1 to |OB| do
       if head$_{A\cup B}$[i]=i then
          begin pre:=i; point:=succ$_A$[i];
                while point$\neq$ 0 do
                   begin
                      if head$_B$[point]=head$_B$[i] then
                         begin succ$_{A\cup B}$[pre]:=point;
                               head$_{A\cup B}$[point]:=i; pre:=point
                         end;
                      point:=succ$_A$[point]
                   end
          end
end.
```

Algorithm 3
Input: Two arrays for *CON* and *DEC*, respectively.
Output: The degree of dependency from *CON* to *DEC*.

```
begin
    count:=0;
    for i:=1 to |OB| do
       if head$_{CON}$[i]=i then
      if [i]$_{CON}$$\subseteq$[i]$_{DEC}$ (*: inclusion) then count:=count+|[i]$_{CON}$|;
  degree:=count/|OB|
end.
```

Algorithm 4

Input: Any object x_i and two arrays for *CON* and *DEC*, respectively.
Output: $[i]_{CON} \subseteq [i]_{DEC}$ or not.

```
begin
  mark:=0; point:=head$_{CON}$[i];
  while point$\neq$0 do
     if head$_{DEC}$[point]=head$_{DEC}$[i] then
      point:=succ$_{CON}$[point] else begin mark:=1; point:=0 end;
  if mark=0 then [i]$_{CON}$$\subseteq$[i]$_{DEC}$ else [i]$_{CON}$$\not\subseteq$[i]$_{DEC}$
end.
```

Chapter 10
Introduction to Perception Based Computing

Andrzej Skowron and Piotr Wasilewski

Abstract. We present a general scheme of interaction and we discuss the role of interactions in modeling of perception processes. We also discuss the role of information systems in interactive computing used to build perception modeling. In particular, we illustrate use of information systems for representation of actions or plans, their (changing in time) pre and post conditions. These information systems create a starting point for perception modeling, i.e., modeling of the process of understanding of sensory measurements.

Keywords: interactive computing, interactive information systems, interactive tables, rough sets, granular computing, wisdom technology.

10.1 Introduction

In this chapter, we consider, perception as the process of understanding of sensory information. Perceiving units will be called agents. Agents are preforming computations on objects called (information) granules (see, e.g., [94, 95, 96, 4, 49, 64, 25, 70, 76]).

The need for perception based computing appears, for example, in problems of analysis of complex processes that result from the interaction of many component processes and from control over such processes. A component process control is aimed at achieving the desired patterns of the system behaviors. This task is a challenge for areas such as multi-agent systems or autonomous systems [28, 79, 56]. Perceived properties of complex processes are often complex vague concepts, about which only partial information is available. Also information about the satisfiability of such concepts determines activating complex actions. It is worth noting that actions can be initiated at different levels of the hierarchy of concepts from a given

Andrzej Skowron · Piotr Wasilewski
Institute of Mathematics, University of Warsaw Banacha 2, 02-097 Warsaw, Poland
e-mail: skowron@mimuw.edu.pl,piotr@mimuw.edu.pl

ontology and that a prediction of action activation at higher levels of the hierarchy is usually conditioned by the perception of properties depending on the history of computations in which information about actions conducted also on lower levels of the hierarchy and their results is stored. Discovery of search strategies for new essential properties at higher levels of hierarchy (e.g., used for activation of compound actions on these higher levels) becomes a challenge and is crucial for understanding of perception. The values of these attributes depend on the history of computing (with registered information about the actions performed on the actual and the lower levels and their results). These new features determine the perception of satisfiability degrees of complex concepts mentioned above conditioning execution of actions on the considered level of hierarchy. The difficulties of analysis and synthesis of perceptual computations follow from the nature of interactive computations, in which it becomes necessary to take into account interactions between processes during performed steps of computing (called intrastep interactions [21]) and not only interactions taking place after completing of computation steps (called interstep interactions [21]). These difficulties follow also from partial information about component processes, from possible interactions between them, and also from requirements on avoidance of central control. Hence, computations on granules performed by agents should be interactive. This requirement is fundamental for modeling of complex systems [17]. For example, in [37] this is expressed by the following sentence:

> *[...] interaction is a critical issue in the understanding of complex systems of any sorts: as such, it has emerged in several well-established scientific areas other than computer science, like biology, physics, social and organizational sciences.*

We discuss the role of information systems in modeling of perception processes. Using a general scheme of interactions, we distinguish basic information on interactions in time between agent and its environment which should be represented for proper interaction modeling. Next, we define a special class of decision tables in which such information can be represented. These decision tables create a starting point for modeling of perception processes.

The chapter has the following organization. In Section 10.2, we discuss motivation for Perception Based Computing (PBC) and Section 10.3 overviews the basic ideas and positions about perception. Section 10.4 presents interactive information systems together with some concepts from hierarchical modeling in rough set theory. Section 10.5 is devoted to elements of interactive computing and rough set analysis of interactive computing. Section 10.6 describes attributes representing actions and shows examples illustrating how planning can be described and analyzed by means of interactive information systems. Section 10.7, contains an introduction to granule and interaction semantics which creates a step toward discussion on approximate reasoning rules on interactions and interactive computations (e.g., specifying beliefs of agents and their adaptive changes).

10.2 Motivation for Perception Based Computing

Perception Based Computing (PBC) methods are needed for solving problems of data mining (DM) and knowledge discovery in databases (KDD) with dynamically evolving complex data (e.g., stream data sources, sensory data). Another challenge, making PBC methods indispensable, is a growth of the size and complexity of data sources (e.g., Web sources, neuro-imaging data, data from network interactions). These challenges, in particular, discovery of complex concepts such as behavioral patterns, hardly can be met by classical methods [55]. They can be met by KDD systems which dialogue with experts or users during the discovery process [88] or by adaptive learning systems changing themselves during the learning process as the response to evolving data.

Another area where PBC methods are needed is the multi-agent systems field. Behavior steering and coordination of multi-agent coalitions acting and cooperating in open, unpredictable environments call for interactive algorithms [18], i.e. algorithms interacting with the environment during performing particular steps of computations or changing themselves during the process of computation. Next challenge of this type comes from human - robot interaction. The problem of human control over autonomously coordinating swarms of robots is the central challenge in this field which should be solved before human - robot teams can be taken out of laboratories and put to practical use.

Coordination and control are essentially perception based thus PBC methods are indispensable for designing and behavior description of cognitive systems and for understanding interactions in layered granular networks [49] where granules can be interpreted both as data patterns and agents (e.g., robots or mobile sensors). Granules in such networks which are additionally self-organizing can be also understood as cores in pertinent multi-core computing engines in structurally and run-time reconfigurable hardware, what makes PBCs useful in computer engineering as well as an essential part of cognitive informatics.

Presented approach is aimed at developing methods based on the generalized information systems (a special kind of data tables) and the rough set approach for representing partial information on interactions in layered granular networks [25, 75]. The idea of the representation of interactions using information systems has some roots in such approaches as rough sets introduced by Zdzisław Pawlak [42], the information flow by Jon Barwise [6] or Chu spaces [5, 11]. Information systems are used to represent granules of different complexity and the interactions among them [75]. Rough sets are used for vague concept approximation [46], for example, in the approximation of ontologies given by experts.

Note that the fusion of information may lead to new information systems with structural objects [75, 76, 70] or to nets of information systems linked by different constraints. For example, a family of parameterized sensors may model a situation in which the sensors are enabled by the judgment module for recording features of video at different moments of time in probing the environment. This makes it possible to collect the necessary features of the environment for an activating of the relevant higher level action. Parameters may be related, e.g., to positions of moving camera. This is closely related to the approach to perception presented in [36] (page 1) (see also Figure 1):

... perceiving is a way of acting. Perception is not something that happens to us, or in us. It is something we do. Think of blind person tap-tapping his or her way around a cluttered space, perceiving the space by touch, not all at once, but through time, by skillful probing and movement. This is, or at least ought to be, our paradigm of what perceiving is. The world makes itself available to the perceiver through physical movement and interaction.

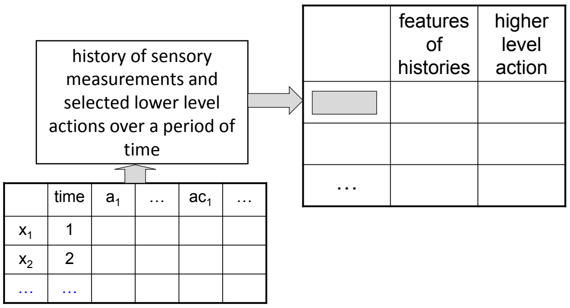

history of sensory measurements and selected lower level actions over a period of time
features of histories
higher level action
...
time
a_1
...
ac_1
...
x_1
1
x_2
2
...
...

Fig. 1 Action in perception

The discussed above example on a family of parameterized sensors suggests that the sensory attributes may be fused using some parameters such as time of enabling or position of sensors. Certainly, for performing more compound actions it is necessary to use a net of such parameterized sensors in which sensory attributes are linked by relevant constraints [36], [65]. Hierarchical modeling may also lead to nets of information systems constructed over information systems corresponding to sensory attributes. Nodes in these networks may be linked using different information such as behavioral patterns or local theories induced from information systems in nodes as well as their changes when information systems are updated. In the former case, the reader may recognize some analogy to theory of information flow [6].

We proposed to build foundations for Perception based Computing (PBC) on the basis of Interactive Granular Computing (IRGC), in particular on Interactive Rough Granular Computing. A step toward this goal is presented in [75, 76]. PBC can be considered in a more general framework of Wisdom Technology (Wistech) [23, 25, 24] based on a metaequation

$$wisdom = knowledge + adaptive\ judgment + interactions. \tag{1}$$

In the above metaequation there is mentioned a special kind of reasoning called as *adaptive judgment*. There are many important issues belonging to adaptive judgment such as searching for relevant approximation spaces including inducing new features, feature selection rule induction, discovery of measures of inclusion and strategies strategies for conflict resolution, adaptation of measures based on the minimum description length, adaptive reasoning about changes, perception (action and sensory) attributes selection, adaptation of quality measures during computations performed by agents, adaptation of object structures, adaptation of strategies for knowledge representation and interaction with knowledge bases, ontology acquisition and approximation, adaptive strategies for beliefs changes (e.g., changes of approximate reasoning rules on interactions), discovery of language for cooperation or competition, and adaptive strategies for language evolution. In general, adaptive judgment is a mixture of deductive and inductive reasoning methods for reasoning about interactive granular computations and on controlling such computations by adaptive strategies for. The mentioned mixture of deductive and inductive reasoning creates many challenges. This is closely related to opinion expressed by Leslie Valiant `http://people.seas.harvard.edu/~valiant/researchinterests.htm`

A fundamental question for artificial intelligence is to characterize the computational building blocks that are necessary for cognition. A specific challenge is to build on the success of machine learning so as to cover broader issues in intelligence. This requires, in particular a reconciliation between two contradictory characteristics – the apparent logical nature of reasoning and the statistical nature of learning.

10.3 Perception [15, 3]

Perception is one of the main forms of interaction of an agent with the environment. Moreover, this form is indispensable in the case of interactive systems. Without perception every action made by agent in the environment would be blind, without it agent would not be able to adapt its behavior to changing conditions of the environment or to modify dynamically its course of actions as a response to results of agent's actions in the environment. It is so because of that perceiving of conducted actions results is an essential part of feedback mechanism and makes adaptive change of a course of actions possible.

In contemporary psychology, perception is understood as a process of perceiving of external or internal environment of an organism leading to understanding of sensations/sensory data. This process consists of two stages: receiving sensations and perception itself. But what does it mean sensation and then understanding of sensation?

Modern experimental psychology was originated in research on sensations within approach called *psychophysics* as started by Ernst Heinrich Weber and Gustav Theodor Fechner. In psychophysics, sensation was defined as realizing or becoming aware by subject of properties of an object or an event that occurs when receptors of some type are stimulated. Then most psychologists were concentrated on the second stage, perception. In terms of sensations, perception is now understood as process of organization and interpretation of sensations. Sensation organization means grouping or dissecting them according to their similarity and discernibility, or to compare sensations and specifying relations between them. Sensation interpretation mean recognition of objects on the basis of sensations or attaching a unique meaning to sensations patterns that can be understood in different ways.

To solve the problem of understanding of ambiguous sensation patterns Herman von Helmholtz founded perception on the idea of unconscious inference and probability principle. Helmholtz claimed that perception results from incomplete data by means of unconscious inferences: making assumptions and conclusions based on previous experiences. Inferences according to him are made about the scene that most likely caused the retinal image or event which itself is ambiguous. This approach was called constructivism since perception results are constructions explaining ambiguous sensations. Constructivism highlighted the role of human mind in perception process.

Contrary to that, next approach, structuralism proposed by Wilhelm Wundt was founded on the philosophy of British empiricism: perception results from the association of basic sensory atoms in memory on the basis of their repeated, prior joint occurrences. Thus, perception is totally based on learning.

The next approach, *gestaltism*, proposed and developed, among others, by Max Wertheimer, Wolfgang Khler and Kurt Koffka rejected primacy of learning in perception. Gestaltism was developed in close relation to Edmund Husserls phenomenology stressing the primacy of whole in the perception process. Perceiving of the whole, which cannot be reduced to its parts, results from combination of some innate structures in human brain with objective properties of the perceived stimulus (sensations). Results of this combination are emergent properties of the whole which are not features of its components. Gestaltists proposed also other laws describing perceptions including the law of Prgnanz, dealing with ambiguous sensations but which, differently from von Helmholtz approach, is not based on the likelihood principle: ambiguous sensations of several geometrically possible organizations will be interpreted as the best, simplest and most stable shape.

The reaction to Gestalt was the new approach called the *New Look*. It emphasized the rationalizing role of the perceiving agent and influence of knowledge on the perception. In series of experiments Jerome Bruner and Cecile Goodman shows that identical sensual stimuli produce different perceptual results depending on subjects social status. The New Look pointed out the predominance of cognitive and emotional aspects of perception, subjects past experience (what was showed also previously) but also their expectations and emotional reactions to the stimulus (what was new idea).

While the previous approaches either underlined the role of an organism in perception process (psychophysics, structuralism, gestaltism, New Look) or balanced between an organism and its environment (constructivism), then the next approach, *ecological optics* proposed by James Gibson [13, 14], was opposite. In Gibsons opinion, perception is based on the relationship between the organism and the environment, but in this setting the environment is the element that plays a main role. The environment is characterized by *affordances*, the particular properties corresponding to what the environment allows the perceiving agent to do. Thus affordances represent opportunities offered by the environment for interaction between sensorimotor abilities of agent and the real granules existing in the world.

The next approach which gained presently the widest acceptance in the field of perception research was proposed by David Marr [30]. This approach is computational in nature and perception is analyzed form information processing perspective. Marr, started from Gibsons position, adopted Chomskys distinction between competence and performance. Marr understood competence in the context of perception as defining what is computed and why. Performance defines algorithms that will be used to execute of computation. On the basis of this distinction, Marr proposed that every perception system, and more generally, every information processing systems can be analyzed on three different levels. On the first, computational level, which precedes the second, competence defines tasks of the perception system. On the second, algorithmic level, performance specifies procedures that will be executed to achieve the tasks. And finally on the third level, underlying first two, it is specifying how these procedures will be implement in an organism or a machine. In the case of human perception, the third level is neurophysiologic, here neurophysiology of nervous cells that are attained to perception should be known, to connect two higher levels to the organic system which process sensory information perceiving the world.

The approach to perception considered in this chapter is wider than studied in the visual perception domain, however it follows Marrs's ideas [30]. Perception here is treated as action-oriented perception [2, 36] and driven by actions. Goals of initiated action help with selection of an appropriate perceptual interpretation among many ones attached to given information provided by senses/sensors. For example, by analogy to visual perception one can attempt to construct softbots acting in Web on the basis of perception characterized by their sensory measurements together with ability to perform reasoning leading from these measurements to conclusions about satisfiability of complex vague concepts used as guards for actions. The guards can be approximated on the basis of measurements performed by sensory attributes only rather than defined exactly. Satisfiability degrees for guards are results of reasoning called as the adaptive judgment. The approximations are induced using hierarchical modeling. Note that, e.g., injecting domain knowledge into a relational database engine [83] will also require approximate modeling of situation perception by domain experts or system users.

10.4 Interactive Information Systems

In this paper, we describe and analyze interactive computing using rough sets. In particular, in this section, we discuss some aspects of information systems in perception processes by agents.

Rough set theory, created by Zdzisaw Pawlak [39, 40, 41], is an approach to granular computing based on ability to discern between objects. Objects and granules are represented by means of information systems called also information tables. *An information system* is a triple $\mathscr{A} = (U, At, \{V_a\}_{a \in At})$, where U is a set of *objects*, At is a set of *attributes*, and each V_a is a *value domain* of an attribute $a \in At$, where $a : U \longrightarrow \mathscr{P}(V_a)$ ($\mathscr{P}(V_a)$ is a the power set of V_a). If $a(x) \neq \emptyset$ for all $x \in U$ and $a \in At$, then $\mathscr{A}$ is *total*. If $card(a(x)) = 1$ for every $x \in U$ and $a \in At$, then $\mathscr{A}$ is *de-*

terministic, otherwise $\mathscr{A}$ is *nondeterministic*. Rough sets were originally introduced for information systems with deterministic attributes, or more exactly for systems where attributes are total functions into their value domains i.e., $a : U \longrightarrow V_a$ for every $a \in At$ (that are naturally identified with deterministic attributes). However, in various applications of rough sets it turns out that it can be useful to admit non-total information systems with nondeterministic attributes. Such systems can represent incomplete information which appears often in data mining and machine learning when searching for new patterns or constructing classifiers starting from sample sets of objects. Such searching can result in creating new nondeterministic attributes with values that are relational structures. Incompleteness of information in rough set theory can be also reflected by partial descriptions relative to specific sets of attributes. Let $(U, At, \{V_a\}_{a \in At})$ be an information system, partial information about a given object $x \in U$ is represented by the A-signature of x, $InfA(x) = \{(a, a(x)) : a \in A\}$ where $A \subseteq At$. Signatures can be treated as vectors of attribute values possessed by objects from information systems, so they are descriptions of objects by means of attributes from a given set of attributes. Signatures represent elementary granules of information understood as sets of objects with the same descriptions.

Another assumption about original information systems was previously accepted, namely that, as mappings, attributes are surjections, i.e. every attribute value is possessed by at least one object, what reflects a closed-world data base point of view. This assumption was implicitly broken in information systems with real-valued attributes and explicitly by introducing *sensory* and *perception* attributes [72, 75]. Attributes which are injections can be viewed as open for interactions in the sense that it is admitted that a new value, not assigned previously to objects, can appear. Thus attributes which are surjections or injections are also called *closed* or *open attributes*, respectively [75].

In the case of perception attributes another partition of attributes is essential. One can differentiate between *atomic* and *constructible* attributes. *Atomic attributes* are basic in the sense that their values depend only on some external factors, with respect to a given information system and are independent from the values of other attributes of this system. Atomic attributes can be closed as well as open. *Constructible attributes* are complex attributes which are inductively defined from atomic attributes of a given information system: if b is a constructible attribute, then for any some object x and already defined atomic attributes $a_1, a_2, \ldots, a_k$:

$$b(x) = F(a_1(x),\ a_2(x), \ldots,\ a_k(x)), \tag{2}$$

where $F : V_{a_1} \times V_{a_2} \times \ldots \times V_{a_K} \longrightarrow V_b$ and values form V_b are constructed on the basis of values from V_i for $i = 1, \ldots, k$.

Sensory attributes represent sensor measurements. They are open atomic attributes which values are results of measurements conducted by sensors thus they depend only on the environment and are independent from values of other attributes. In Figure 2, we illustrate the basic features of sensory attributes. *Perception attributes* are sensory attributes or constructible attributes defined on the basis of sensory ones. The latter are also called *complex perception attributes*. Complex

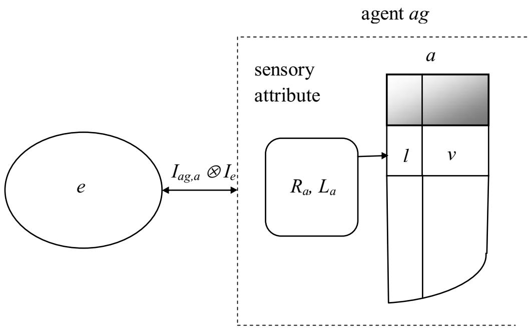


Fig. 2 Sensory attribute. e denotes the environment, $\mathscr{R}_a, L_a$ - relational structure of sensory attribute a and set of formulas assigned to a, respectively. We assume that the result of the interaction of sensory attribute a with the environment results in selection of a formula from a language L_a assigned to a. l is a label of the environment state currently perceived by a, v is the index such that the formula $\alpha_v \in L_a$ was selected in interaction of a with the environment. In the shadowed area the results of past interaction are stored, the interaction of a with the environment e is not changing e (the changes of e are caused by dynamics of the environment only). In the agent state only a row with label l and v was added and represents the result of sensory attribute a measurement.

perception attributes represent higher order result of perception, e.g. some identified patterns or created perceptual granules. For formal descriptions of sensory and perception attributes the reader is referred to [75] (section 7).

Constructible attributes can take as their values relational structures defined over its value domains. Hence, we consider a generalization of traditionally used information systems [39, 40, 41] by considering together with value set V_a a relational structure over V_a. Note that also objects in such information systems may have complex structure. To explain this we present an illustrative example. Let B be a family of atomic attributes. We define a value set

$$V_B = \bigcup_{B' \subseteq B} \prod_{b \in B'} V_b \tag{3}$$

This set consists of sequences of values, i.e. value vectors, which are subsequences of sequences from $\prod_{b \in B} V_b$. Now, a new attribute b_0 is defined in the way that for any object x, $b_0(x) = (V_{b'}, r_i)$, where $V_{b'} \subseteq V_B$ and $r_i \subseteq V_{b'} \times V_{b'}$ for $i = 1, \ldots, n$. Therefore attribute b_0 is a function of the form $b_0 : U \longrightarrow \{(V_a, r_i)\}_{i \in I}$. Since not every value vector from family V_B has to be an information signature, then attributes constructed in that way are open attributes.

Relational structures corresponding to attributes can be fused. Let $\{(V_{a_i}, \tau_{a_i})\}$ be a family of tolerance spaces, i.e. relational structures where V_{a_i} is a value domain of an attribute a_i and $\tau_{a_i} \subseteq V_{a_i} \times V_{a_i}$ is a tolerance relation (relation that is reflexive and symmetric) for $i = 1, \ldots, k$. Their fusion is a relational structure over $V_{a_1} \times \ldots \times V_{a_k}$ consisting of a relation $\tau \subseteq (V_{a_1} \times \ldots \times V_{a_k})^2$ such that for any

$(v_1, \ldots, v_k), (v'_1, \ldots, v'_k) \in V_{a_1} \times \ldots \times V_{a_k}$ we assume $(v_1, \ldots, v_k)\tau(v'_1, \ldots, v'_k)$ if and only if $v_i \; \tau_{a_i} \; v'_i$ for $i = 1, \ldots, k$. Note that τ is also a tolerance relation. Intuitively, a vector $(v_1, \ldots, v_k)$ represents a set of objects possessing values $v_1, \ldots, v_k$ for attributes $a_1, \ldots, a_k$ respectively. Thus some vectors from $V_{a_1} \times \ldots \times V_{a_k}$ (not necessarily all) represent granules consisting of objects (some vectors from $V_{a_1} \times \ldots \times V_{a_k}$ correspond to the empty set). Therefore a relation τ corresponds to a relation between granules. Constructible attributes defined by means of relational structures can be used to represent some structural properties of objects, for example time windows in information systems where objects are time points. In hierarchical modeling, object signatures at a given level of hierarchy can be used for constructing structural objects on the next level of hierarchy.

For solving classification problems, *decision information systems* (called also *decision tables*) were distinguished [41, 46]. They are information systems of the form $\mathscr{A} = (U, C \cup D, \{Val_a\}_{a \in C \cup D})$ where a family of attributes is divided into two disjoint classes $C, D \subseteq At$, elements of which are called *condition* and *decision attributes* respectively. These systems are used in analysis of decision rules [41, 46]. *Action attributes* are decision attributes in decisions information systems where condition attributes contain also sensory attributes and, possibly, complex perception attributes. The basic features of action attributes are illustrated in Figure 3. For more exact characterization of action attributes see section 9 below. *Interactive*

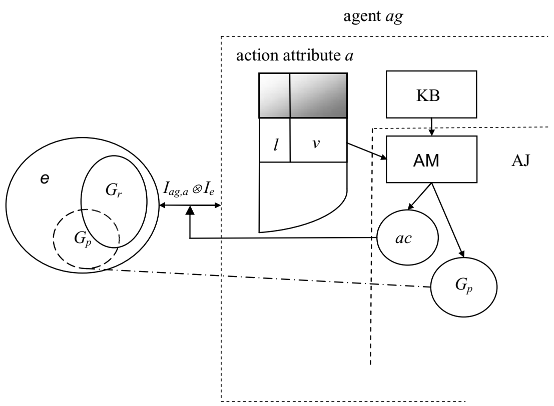


Fig. 3 Action attribute. On the basis of the current information about the current state of the agent *ag* and the state of the environment *e*, the action attribute *a* is selecting an action *ac* and predicts changes of the environment caused by *ac* which are represented by granule G_p. l, v have meaning as in Figure 2. AJ denotes the adaptive judgment module with the action submodule denoted by AM. The action attribute *a* is selecting an action *ac* to be performed (using module AM, knowledge base KB contents, and the measurement results stored by sensory attributes). Changes in *e* caused by *ac* in the form of granule G_p are predicted too. The selected action *ac* determines the interaction $I_{ag,a}$ with the environment from the side of the agent *ag*. Note that reaction of the environment may be unpredictable and the granule G_r representing change of *e* as the result of $I_{ag,a} \otimes I_e$ (on the component of the environment) may be different from predicted described by granule G_p.

information systems are decision systems where condition attributes contain sensory attributes together with some complex perception attributes and decision attributes contain action attributes. Instead of term *interactive decision tables* we will use term *interactive tables*.

10.5 Interactive Computing

In this section, we briefly present an idea of interactive computing. In our view, planning is an essential part of interactive computing. Interactive algorithms are adaptive in the sense that they can change themselves during computation performing, the next state of the algorithm can be only foreseen with some probability, since every step can be changed as a response to the environment influence during performing of that step. However, they should be differentiated from probabilistic algorithms: once a step of probabilistic algorithm is drawn then performing it cannot be changed regardless of the environment influence. In this sense, probabilistic algorithms are inflexible and not adaptive. Therefore, interactive algorithm need to react even during performance of a previously specified step. This leave an open space for planning: even a very simple interactive algorithm is more like an adaptive strategy than a drawing steps algorithm which beside this is rigid.

In the process of interactive computation both an agent and an environment are involved. A system performing interactive computing consists of an agent and the agent's environment, more exactly, a part of the environment that is perceived by an agent. The global states of such system are defined as pairs $(s_{ag}(t), s_e(t))$, where $s_{ag}(t)$ and $s_e(t)$ are states of a given agent ag and the environment e at time t, respectively. Figure 4 illustrates how, in the case of interactive computations, the transition relation $\longrightarrow$ between global states is performed, i.e., when $(s_{ag}(t), s_e(t)) \longrightarrow (s_{ag}(t+\Delta), s_e(t+\Delta))$ holds, where Δ is a time necessary for performing the transition. $A(t)$, $E(t)$ denote the set of attributes available by agent ag at the moment of time t and the set of attributes (sensors) influenced by environment e at time t, respectively. $Inf_{A(t)}(s_{ag}(t), s_e(t))$ is the signature [75] of $(s_{ag}(t), s_e(t))$ relative to the set of attributes $A(t)$ and $Inf_{E(t)}(s_{ag}(t), s_e(t))$ is the signature of $(s_{ag}(t), s_e(t))$ relative to the set of attributes $E(t)$, i.e. signature $Inf_{A(t)}(s_{ag}(t), s_e(t))$ describe a state of agent ag at the moment of time t, while signature $Inf_{E(t)}(s_{ag}(t), s_e(t))$ describe a part of environment e that is perceived by agent ag at time t. These signatures are arguments of strategies Sel_Int_{ag}, Sel_Int_e selecting interactions I_{ag} and I_e of agent ag with environment e and environment e with agent ag, respectively. I_{ag} represents the planned influence of agent ag on environment e (and on the agent ag itself), i.e. an action of agent ag while I_e represents an influence of environment e on agent ag (and on the environment) which results will be perceived by ag. This includes also predicted results of agent action on environment e (and on the agent ag itself) as well as a perception of change of e caused by the previous global state. $I_{ag} \otimes I_e$ denotes the result of the interaction product $\otimes$ on I_{ag} and I_e. Since set $E(t)$ can be insufficient for describing environment e therefore agent ag can have very incomplete information about I_e as well as the result $(I_{ag} \otimes I_e)(s_{ag}(t+\delta), s_e(t+\delta))$ only, where

δ denotes the delay necessary for computing the signatures and selection of interactions (for reasoning simplicity we assume that these delays for *ag* and *e* are the same). Thus, information about $s_{ag}(t+\Delta)$ and $s_e(t+\Delta)$ perceived by *ag* can be very incomplete too. Usually, agent *ag* can only estimate of $s_{ag}(t+\Delta)$ and $s_e(t+\Delta)$ during planning selection of interaction I_{ag}. These predictions then can be compared with the perception of global state $(s_{ag}(t+\Delta), s_e(t+\Delta))$ by means of attributes $A(t+\Delta)$. Interaction $I_{ag} \otimes I_e$ can change the content of both the agent state and the environment state. The current set of attributes $A(t)$ is a part of the agent state $s_{ag}(t)$ and can be changed, for example, by adding new attributes discovered using I_{ag}. As we mention at the beginning of this section, interactiveness of agent *ag* is formally reflected by the fact that the description of strategy Sel_Int_{ag} is stored in the current state of agent $s_{ag}(t)$. This strategy itself can be modified as the result of interaction, and generally, sets of attributes as well as strategies for selecting interactions can be adopted in time.

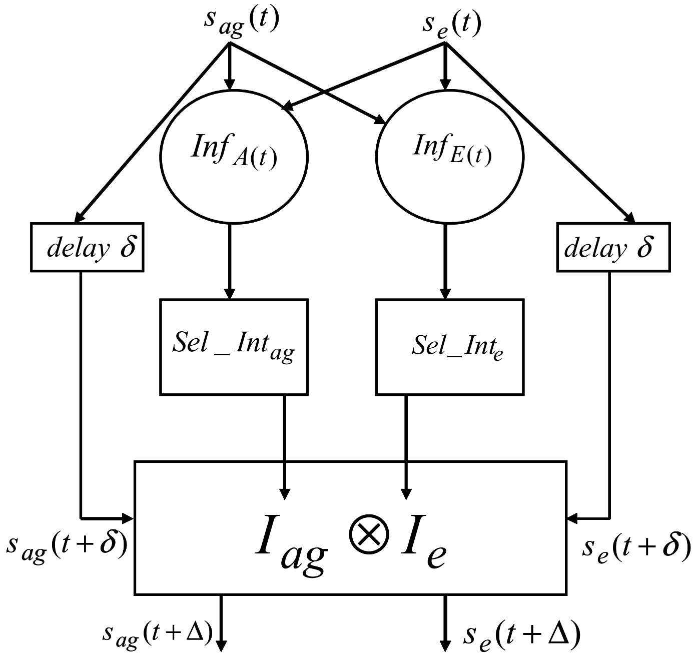


Fig. 4 Transition from global state $(s_{ag}(t), s_e(t))$ to global state $(s_{ag}(t+\Delta), s_e(t+\Delta))$.

Interactive computing is not performed solely by agents in the sense that also environments are essentially involved in the computation process. Therefore, more exactly is to say that a given agent *ag* observes computation than that *ag* perform it, and that an interactive computation is performed commonly by the agent and its environment, namely, this part of the environment which is perceived by the agent and affects the agent. However, agent *ag* affected by its environment does not passively respond to the environment, but it applies strategy Sel_Int_{ag} selecting interactions with its environment. This has another consequence, namely that agent *ag* does not observes the whole environment, only part of it determined by the activated sensors of *ag*. Hence, in particular, values of not all attributes in a given row of information systems describing *ag* are known. An agent posses only partial information about the environment. It should be noted that another interaction also can take place, namely internal interaction inside an agent between its components: an agent can observe this interaction using its sensors, in this case internal sensors, like in animals signals of a somatosensory system. And, as in the case of the environment, an agent usually posses only partial information about its internal states.

Interactive computations in our approach are sequences of signatures of global states connected by transition relation. More formally, *a computation observed by an agent ag in interaction with its environment e* is any sequence

$$sig_1, \ \dots \ sig_n, \ \dots \tag{4}$$

fulfilling the following conditions: for some t, Δ and for any i, sig_i is the signature of global state $(s_{ag}(t+i), s_e(t+i))$ relative to the attribute set $A(t+i)$ available by *ag* at a moment of time $t + i\Delta$ and

$$(s_{ag}(t+i\Delta), s_e(t+i\Delta)) \longrightarrow (s_{ag}(t+(i+1)\Delta), s_e(t+(i+1)\Delta)).$$

Length of a computation Comp observed by *ag* is a number of elements of sequence minus one, when *Comp* is a finite or is a cardinality of *Comp*, when it is infinite sequence.

10.6 Action Attributes and Plans

Actions are responses of an agent to the influence of its environment. By actions an agent affects its environment while the environment affects the agent and this influence is perceived by the agent through a perception process. Thus action is one of the forms of interaction between an agent and its environment. Every action is aimed at reaching some specified goal. This goal can be viewed as a part of a global state of an agent and its environment (more exactly, the part of an environment perceived by the agent). One of the main problems connected with a performance of an action is making decision whether the goal was reached or not, or a dynamical version of this decision: when the goal is reached. Thus every action planning should result also in specification of tools for making such decisions. It is done by a specification of an expected state of the environment and possibly the agent too. A specification of time necessary for reaching the goal, i.e. finishing the action, namely time after which a decision whether action was successful or not is needed too.

In our approach to interactions, strategies Sel_Int_{ag}, Sel_Int_e are responsible for planning. Strategy Sel_Int_{ag} on the basis of signatures $Inf_{A(t)}(s_{ag}(t), s_e(t))$ and $Inf_{E(t)}(s_a(t), s_e(t))$ selects interaction I_{ag}, i.e. proposed action while strategy

Sel_Int_e on the basis of $Inf_{A(t)}(s_{ag}(t), s_e(t))$ and $Inf_{E(t)}(s_{ag}(t), s_e(t))$ selects interaction I_e which represents a predicted environment influence on agent and on the environment itself. Since both strategies operate on the same signatures, then for the sake of simplicity we can represent their result by one strategy Sel_Int i.e.

$$Sel\_Int(x,y) = (Sel\_Int_{ag}(x), Sel\_Int_e(y)) \tag{5}$$

where x and y range over signatures relative to the attribute sets A and E respectively, thus, for example, x and y can be substituted with signatures $Inf_{A(t)}(s_{ag}(t), s_e(t))$ and $Inf_{E(t)}(s_{ag}(t), s_e(t))$ respectively. Strategy Sel_Int returns selected action ac_1 together with objective o_1, i.e. the expected result of performing that action and the estimated time Δ_1 needed for that performance. Action ac_1 is a part of I_{ag}, I_{ag} can also influence internal states of agent ag, while objective o_1 and time Δ_1 are contained in I_e. Both interactions I_{ag} and I_e can dynamically affect each other and the result of such interfering interaction is given by product $I_{ag} \otimes I_e$. Thus the product $I_{ag} \otimes I_e$ returns global state $(s_{ag}(t+\Delta_1), s\_e(t+\Delta_1))$. It can happen that this global state differs from the expected result of completing action ac_1 after time Δ_1. In such case one can apply a quality criterion defined by a quality measure evaluated on computations observed by the agent in order to decide whether a goal is reached or not, or to evaluate action performance quality. To define and compute such quality criterion is necessary for planning, i.e. for discovering a selecting interactions strategy but it is very difficult problem. Similarly, the main task in reinforcement learning [81, 78] is to learn the approximation of function Q which real value $Q(s,a)$ describes the reward for executing action a in state s. For solving this task probabilistic models are used. However, for compound real-life problems it may be hard to build such models for such compound concept as $Q(s,a)$ [87]. The relationships of reinforcement learning and rough sets are discussed in [50, 51, 52, 53]. For relationships of reinforcement learning, rough sets, and hierarchical learning the reader is referred, e.g., to [34, 8, 7, 35].

Let $A = (U, C \cup D, \{Val_a\}_{a \in C \cup D})$ be an interactive information system representing agent ag. This system is illustrated by Figure 5. Objects of the system correspond to global states of the form $(s_{ag}(t), s_e(t))$ thus we assume that the rows in this interactive table are labeled by state names indexed by time moments from the specified time interval $[g,h]$ in discrete time T (a set T is denumerable) and $U = \{s_t : t \in T\}$ where s_t is the name of global state $(s_{ag}(t), s_e(t))$, i.e. objects of the system are names of global states at time moments from the interval $[g,h]$. Concerning attributes, for every moment $t \in U$, $A(t) \subseteq C \cup D$, i.e. some of attributes from attribute set $A(t)$ represent information relevant to internal states of ag and some to actions of ag, while $E(t) \subseteq C$, i.e. attributes from $E(t)$ are entirely condition attributes. Note that perception attributes of agent ag including sensory ones are contained in $E(t)$. It should be highlighted that objects in interactive information systems are just names or labels of global states about which only partial information is available to the agent. In interactive tables only information relative to attributes from sets A and E available to the agent in particular moments of time is stored. Atomic attributes correspond to sensors or internal, build-in-agent states and when values of these attributes have to be calculated they come from outside of the information table, namely from sensors, or they are specified by agent internal states. Values of constructible attributes, including actions and complex perception attributes, are calculated on the basis of values of atomic attributes or other constructible attributes, namely on the basis of information stored inside an interactive table. For the sake of notational simplicity, for an attribute a, instead of $a(s_t)$ we can write just $a(t)$ but keeping in mind remarks made above.

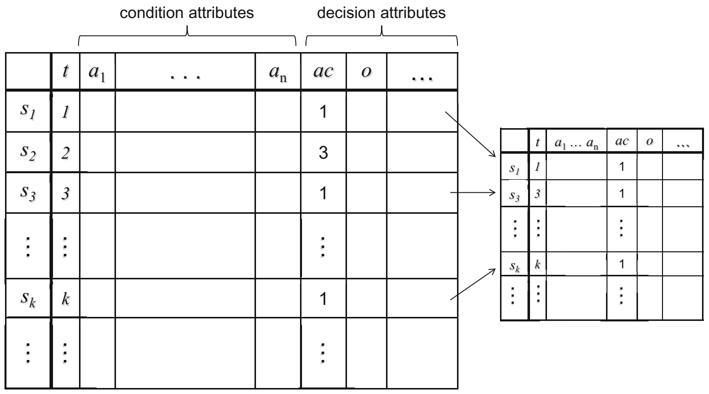


Fig. 5 An interactive information system representing a particular agent and creation of an interactive information system representing a particular action.

In order to represent actions we introduce a new attribute, denoted by ac, which values are actions denoted by natural numbers, we assume also that ac is an injection of the form $ac : U \longrightarrow \mathbb{N}$, where $\mathbb{N}$ is the set of natural numbers. By this assumption we mean that new actions can appear in the flow of time. We also admit notation if $ac(t) = n$, then *n uniquely identifies the action ac_n*. $ac(s_t)$ represents action selected by the agent in time t. In order to represent action objectives we introduce a new attribute denoted by o which values are ordered pairs of the form (v_n, Δ_n), i.e. $o(s_t) = (v_n, \Delta_n)$, where $n \in \mathbb{N}$ and v_n is a value vector from the set

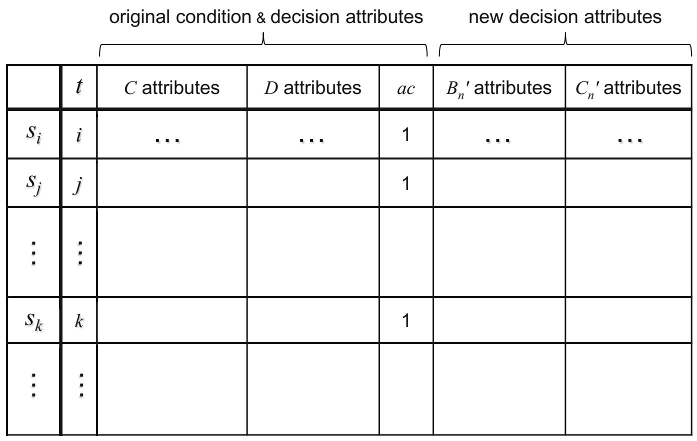
$$V_C = \bigcup_{B' \subseteq C} \prod_{b \in B'} V_b \tag{6}$$

representing expected result of an action ac_n and $\Delta_n \in T$ is an estimated time needed for performing action ac_n. Therefore $o(s_t) = (v_{ac(t)}, \Delta_{ac(t)})$ for every $t \in U$. It follows that $ac, o \in D$, i.e. attributes ac and o are decision attributes. In addition, we assume that interactive table $\mathscr{A}$ is consistent. It means that attributes ac and o take values in such way that to every condition signature it is attached only one action, one values vector and one moment of time.

In order to analyze actions we create new interactive information systems specified for every action separately. This system is illustrated by Fig. 3. Note that every value vector v_n contains also information about attributes used for prediction of action results, i.e. for every v_n such that $o(s_t) = (v_n, \Delta_n)$ there is $B' \subseteq C$ such that $v_n = Inf_{B'}(s_t) = \{(a, a(st)) : a \in B'\}$. Let us denote such B' by $B_{n,t}$. For the sake of simplicity we assume that in two different moments of time for every action, values vectors with respect to the same family of attributes are predicted, i.e. for every $n \in N$ and $i, j \in U$, if $i \neq j$, then $B_{n,i} = B_{n,j}$. Thus for every action attribute ac_n we have family B_n of attributes such that all predicted values vectors are B_n-signatures. Note that $B_n \subseteq C$ but we would like to add attributes from family B_n to the new information system as decision attributes, thus we create a new family B'_n that is disjoint with B_n has the same cardinality as B_n and consists of copies of attributes from family B_n, i.e. for every $a' \in B'_n$ there is a B_n such that for every $t \in U$, $a'(t) = a(t)$. We would like also to compare the condition attributes signatures taken at moments of action selecting action with signatures with respect to the same attributes taken at moments of finishing action execution after estimated time, e.g. to compare signature $Inf_{C(t)}$ with signature $Inf_{C(t+n)}$. For the sake of such comparison we will add C-attributes as decision attributes and analogously, as in the case of B_n-attributes, we create a new family C'_n containing copies of attributes from C possessing some additional property that for every $c' \in C'_n$ there is $c \in C$ such that $c'(t) = c(t+n)$. For action ac_n we define a new interactive information system:

$$\mathscr{A}_n = (U_n, C \cup D_n, \{Val_a\}_{a \in C \cup D_n}) \tag{7}$$

where $U_n = \{t \in U : ac(t) = n\}$, $D_n = D \cup B'_n \cup C'_n$ and attributes families C and D are taken from interactive information system $\mathscr{A}$. Figure 6 illustrates information system $\mathscr{A}_n$.


original condition & decision attributes
new decision attributes

	t	C attributes	D attributes	ac	B_n' attributes	C_n' attributes
s_i	i	...	...	1	...	...
s_j	j			1		
⋮	⋮					
s_k	k			1		
⋮	⋮					

Fig. 6 An interactive information system representing a particular action.

Note that from assumption about consistency of interactive table $\mathscr{A}$ it follows that also interactive table $(U_n, C \cup D \cup B'_n, \{Val_a\}_{a \in C \cup D \cup \{ac\} \cup B'_n})$ is consistent. However, interactive table $\mathscr{A}_n$ does not have to be consistent. It depends on unpredictable results of interaction. It can happen that the same information C-signature taken in two different time moments is mapped to two different C'_n-signatures, i.e. it can happen that there are $i, j \in U_n$ such that $i \neq j$ and $Inf_{C(i)} = Inf_{C(j)}$ but $Inf_{C'_n(i)} \neq Inf_{C'_n(j)}$.

10.7 Towards Granule Semantics

In many areas (e.g., biology, sociology, MAS, robotics, pattern recognition, machine learning or data mining, simulations of complex phenomena, or semantic search engines), the challenge is to discover (induce) compound granules from some elementary ones, representing imperfect knowledge about analyzed objects, and concepts, or/and phenomena in such a way that the compound granules (e.g., clusters of highly structural objects in data mining, new features obtained by feature construction in machine learning or coalitions in MAS) satisfy the given, often vague, target specification to a satisfactory degree. This idea has been coined, e.g., in rough mereology [57]. The hardness of this challenge is caused by the fact that the searching spaces for relevant granules are so huge that efficient searching is often intractable by using the existing methods and the current technology. We propose to support the searching process, e.g., by interactions with domain experts (see, e.g., [7], [80]). This can be done by acquiring from them the relevant ontology and next by making it "understandable" to the system by using rough set methods for domain ontology approximation in the language of the system. Target complex granules are induced using interactive computations on granules (see [7], [75], [73]). Such computations progress through interactions among granules from different levels of layered networks of granules and also by interactions with often unpredictable environments. Interactions between granules are of different complexity, possibly taken from different levels of the hierarchy [75], [73]. Better understanding the nature of the interactions is one of the main goals of the Perception Based Computing. In interactive computations on granules, interactions can be partially controlled [75]. Developing strategies for the discovery of controlling schemes of interactions among granules relevant to the target goal of computations is also one of the main tasks for PBC.

From a mathematical point of view, granules can be represented (exactly or at least to a degree) by sets often from quite high levels of the powerset hierarchy. The power set hierarchy, called also simply set hierarchy, S_X for any set X is defined by transfinite induction[1] as follows: $S_X^0 = X$, $S_X^{n+1} = \mathscr{P}(X \cup S_X^n)$, where n is a natural number, and finally:

$$S_X = \bigcup_{n \in \mathbb{N}} S_X^n,$$

[1] For simplicity, we consider only a special case.

where $\mathbb{N}$ is the set of natural numbers. In other words

$$S_X = \mathscr{P}^{\omega}(X), \tag{8}$$

where

$$\mathscr{P}^{\omega}(X) = X \cup \mathscr{P}(X) \cup \mathscr{P}^2(X) \cup \mathscr{P}^3(X) \cup \ldots$$

and $\mathscr{P}^2(X) = \mathscr{P}\mathscr{P}(X)$. For example, the von Neumann hierarchy being a construction of natural numbers on the basis of ordinals:

$$\emptyset,\ \{\emptyset\},\ \{\emptyset,\{\emptyset\}\},\ \{\emptyset,\{\emptyset\},\{\emptyset,\{\emptyset\}\}\},\ \ldots$$

consists of elements from every level of the set hierarchy. Let us note that in a set hierarchy every level S_X^n consists also all sets from the preceding levels in the hierarchy what particularly is illustrated by the von Neuman hierarchy. In the case of information systems we can adjust a little the definition of set hierarchy. For an information system $\mathscr{A} = (U, At, \{V_a\}_{a \in At})$ we adjust only the basis of induction i.e. the first set from the hierarchy: $S^0_{\mathscr{A}} = U \cup \bigcup_{a \in At}\{V_a\}_{a \in At}$. Definitions of the sets from the next levels of the hierarchy remain unchanged.

Granules can have an elementary structure (such as elementary neighborhoods of objects see: [63], [84, 91]) or a complex structure (such as cognitive agents [90], autonomous agents, teams of agents [82, 89], complex patterns or classifiers in data mining, [80, 1]). Granules of higher order are represented on higher levels of layered networks. For example, granules representing agents can have a complex structure consisting of many components responsible for, e.g., perceiving the environment, planning actions, or sending messages to other agents [75]. Coalitions of agents can be represented as a special kind of granules in layered granular networks. Note that autonomous agents can use complex vague concepts as guards of actions performed during the interaction with the environment [75]. For the approximation of such concepts the rough set approach can be used (see, e.g. [40, 41, 46, 91]). Interactions among granules and approximations of granules are two basic concepts related to interactive computations on granules which will be studied within the project. In layered granular networks we represent the (hierarchical) structure of granules as well as links between interacting granules. Granular layered networks will be built over information systems representing granules and their properties.

We put minimal conditions on the nature of granules: granules are labeled by names and have semantics constructed on the basis of set hierarchy specifying types of particular granules. They can be partially specified. We assume that granules consist of parts. Every part has a name, contains some value and posses a type of contained value. We assume that in addition to values specified by a type of the part, the part can also contain information that its value is unknown.

Let us consider some examples. More formal approach will be presented elsewhere.

Granules can represent some events like in the case of unknown value for a given attribute in some time moment. Consider a granule G_a for attribute a with unknown a-value of type τ at time $t = 27s$. It has a name l_a and consists of three parts, first part with name a is of type τ and containing specification for the value itself saying at the moment that value is unknown, and second with a name t, is of type *seconds* and containing specification of numerical value of time 27 and third part with a name *active* is of binary type and containing specification that at the moment its value is unknown. Such granules represents events or situations and can be viewed as *observational granules* and called also *observables*.

Granules can also represent results of interactions, for example granules interactions. Let G_1 and G_2 be granules interacting in the context of granule G, i.e. G contains information about factors which possibly can influence interaction G_1 and G_2, information not contained in G_1 and G_2. A result of interaction $G_1 \otimes_G G_2$:

$$\frac{G_1 \otimes_G G_2}{H}$$

where $\otimes_G$ denotes interaction in the context of granule G and H is a granule representing typical result of interaction of G_1 and G_2 in the context of G.

Note that usually the agent *ag* has only a partial information about interacting granules and the interaction context. Hence, *ag* can use only approximate reasoning rules about interactions which can be discovered (induced) from experimental data and domain knowledge.

Example 1 (Measuring of attribute value). Granules can represent also actions. In the case of ascertaining unknown value of attribute a at time t discussed above, a process of looking for that value can be activated. For example, if a is a sensory attribute, then a given sensor can be activated in order to perform measurement leading to finding an attribute value for a after some time Δ[2]. However, in this case we should differentiate between a granule representing activation of the measurement process (as a result of decision making process) and a granule representing the process of measurement itself. Let $G_a^{\mathtt{act}}$ be a granule of the first type. It is constructed on the basis of G_a and consists of three parts: first two are the same like in G_a and the third one is of binary type with a name *active* where value 1 is representing the fact that an appropriate attribute is looking for its value. Such granules representing results of decision making processes can be called *decision granules* being particular examples of *action granules*.

Let $G_{\mathtt{control}}$ be a granule representing a control module of agent *ag*. Starting of measuring process can be viewed as a result of interaction between control module and a decision made previously that particular measurement should be done. Then interaction between decision granule $G_a^{\mathtt{act}}$ and control granule $G_{\mathtt{control}}$ activates measuring process:

$$\frac{G_a^{\mathtt{act}} \otimes G_{\mathtt{control}}}{H_a},$$

[2] It is worthy to note that some granules can be equipped in their own sensors perceiving other granules.

where H_a is a granule corresponding to physical sensor responsible for measuring the value of attribute a, so this granule corresponds to the process of measurement. Thus H_a is an example of *action granules*.

Let G_e be a granule representing the environment. Any measurement is a result of interaction between measuring process and the environment, so it can be represented as granule being a result of granule interaction:

$$\frac{H_a \otimes G_e}{(g,(H_a,G_e))},$$

where $(g,(H_a,G_e))$ is a granule representing the result of measurement consisting of two parts. First part contains g - a value representing the result of sensor measurement. The second part contains an information about origins of value g, namely that it is a result of measurement process represented by granule H_a and leaving unchanged environment granule G_e what means that the process of measurement was entirely passive, what not always is the case, like in quantum mechanics.

Then interaction between $(g,(H_a,G_e))$ and G_a leads to fusion of g and G_a, i.e. interaction between value g itself and G_a:

$$\frac{(g,(H_a,G_e)) \otimes G_a}{g \otimes G_a}$$

while interaction between fusion of g and G_a and a granule `clock` gives a new granule:

$$\frac{g \otimes G_a \otimes \texttt{clock}}{G'_a}$$

where `clock` is a granule representing a clock of agent ag and G'_a has the same structure like G_a but a part of G'_a (value attribute part of G'_a) contains value g instead of information about unknown value, while numerical value contained in t part is $t + \Delta$ where t is a value contained in t part of G_a and Δ is estimated time of measuring of value of a including a time for making decision about conducting this measurement.

Example 2 (Performing action i). Let $G^{\texttt{act}}_{ac}$ be a granule representing decision that i-th action at time t should be activated. $G^{\texttt{act}}_{ac}$ has analogical structure to $G^{\texttt{act}}_{a}$, however it consist of four parts. First part with name *ac* is of type ρ and contains a specification of action i, second part with name *time* is of type *moment* and contains a specification of starting time t of performing action i, third part is of binary type with name *active* where value 1 is representing the fact that an appropriate action should be performed, fourth part with name *real* with possibly containing a granule representing (partially) the environment but at the moment contains specification that its value is unknown. Such granules are examples of *decision granules*. Then interaction between decision granule $G^{\texttt{act}}_{ac}$ and control granule $G_{\texttt{control}}$ activates the process of performing action:

$$\frac{G_{ac}^{\mathtt{act}} \otimes G_{\mathtt{control}}}{G_i},$$

where G_i is a granule representing the process of performing action i starting at time t.

Interaction between the process of performing a given action and the environment leads to change of the environment being a result of completed action, which analogously like in the case of measuring, is represented by complex granule consisting of two parts:

$$\frac{G_i \otimes G_e}{(G^0, G_e')},$$

where G^0 is a granule representing (most often partially) the state of the environment after performing action i and G_e' is a granule representing the expected state of the environment after performing action i.

Interaction between granules G^0, $G_{ac}^{\mathtt{act}}$ and granule `clock` leads to a granule representing result of performed action i:

$$\frac{G^0 \otimes G_{ac}^{\mathtt{act}} \otimes \mathtt{clock}}{H_i},$$

where H_i is a granule representing the result of performing action i. H_i has the same structure as decision granule $G_{ac}^{\mathtt{act}}$ but in the case of H_i part *active* has unknown value while part *real* contains granule G^0 and part *time* contains $t + \Delta$, where Δ is a time needed for performing action i specified according to interaction with granule `clock`.

Let `KB` be a granule representing Knowledge Base of agent *ag*. Then granule representing expected state of the environment after performing of action i results from interactions between granule representing the process of performing action i and granule `KB`:

$$\frac{G_i \otimes \mathtt{KB}}{G_e'}.$$

Granules can be described by *attribute logic* (see e.g. [46], [75]). Formulas of attribute logic are created by means of standard logical connectives from atomic formulas of the form $a = v$, where a is an attribute or a part of a granule, and v is a value specified for a. For example granule H_i can be described by a formula:

$$\alpha := ac = i \,\wedge\, time = t + \Delta \,\wedge\, real = G_0 \,\wedge\, active = ?,$$

where "?" stands for "unknown value". Turning it out granules can give semantics for attribute logic. For example, an interpretation of the formula α, denoted by $||\alpha||$ is the family of all granules satisfying specification presented by α. Using attribute logic formulas we can also specify granules of higher order, being, e.g., coalitions of granules For example, formula

$$\beta := active = 1 \vee ac = i$$

specifies a coalition consisting of all decision granules and granules describing somehow action *i*. One of the main goals of attribute logic is to express knowledge contained in knowledge base of a given agent. Such knowledge is necessary for action planning or for control of action performance. One of the main task in action planning and action decision making is to determine expected results of performed action, i.e., to predict how the environment or an agent will be changed as the result of an action performing. Such predictions can be made by reasoning about change. Attribute logic together with granule semantics is needed to construct such reasonings.

10.8 Conclusions

In the chapter, we discussed the basic issues for perception modeling. Among them are models of objects involved in perception, called as granules and interactions between them determining interactive computations on granules, issues of representation of interactions as well as (adaptive) approximate reasoning rules about interactions and interactive computations. We restricted our considerations to rather intuitive presentation of the main ideas. More formal approach will be presented in on of our next paper. In particular, we presented several illustrative examples, of granules interactions. In the paper, we considered low level information systems in perception modeling. It was shown that these information systems in the form of decision tables correspond to recordings by agent in time the perceived values of sensory attributes, the action active at a given moment of time, predicted values of sensory attributes after performing of action as well as the sensory measurements corresponding to the finished action at a given moment of time. Deeper analysis of perception based computing will require considering construction of hierarchical information systems. For example, the (semi)optimal selected action at a given moment of time may be predicted on the basis of high level features of histories recorded in discussed information systems. This directly refers to the main idea of perception discussed in [35]. One of the main challenge of perception is related to discovery of relevant features of such histories. This problem will be discussed in our next paper.

Acknowledgements: The research has been supported by the individual research project realized within Homing Plus programme, edition 3/2011, of Foundation for Polish Science, co-financed from European Union, Regional Development Fund, and by the grants 2011/01/D/ST6/06981 from the Polish National Science Centre and NN516 077837 from the Ministry of Science and Higher Education of the Republic of Poland.

References

1. Abbasian, H., Drummond, C., Japkowicz, N., Matwin, S.: Robustness of Classifiers to Changing Environments. In: Farzindar, A., Kešelj, V. (eds.) Canadian AI 2010. LNCS, vol. 6085, pp. 232–243. Springer, Heidelberg (2010)
2. Arbib, M.A.: The Metaphorical Brain 2: Neural Networks and Beyond. Willey & Sons (1989)
3. Bara, B.G.: Cognitive Science. A Developmental Approach to the Simulation of the Mind. Lawrence Erlbaum Associates, Hove (1995)
4. Bargiela, A., Pedrycz, W.: Granular Computing: An Introduction. Kluwer Academic Publishers (2002)
5. Barr, B.: $*$-Autonomous categories. Lecture Notes in Mathematics, vol. 752. Springer (1979)
6. Barwise, J., Seligman, J.: Information Flow: The Logic of Distributed Systems. Cambridge University Press (1997)
7. Bazan, J.G.: Hierarchical Classifiers for Complex Spatio-Temporal Concepts. In: Peters, J.F., Skowron, A., Rybiński, H. (eds.) Transactions on Rough Sets IX. LNCS, vol. 5390, pp. 474–750. Springer, Heidelberg (2008)
8. Bazan, J.G., Skowron, A.: On-Line Elimination of Non-Relevant Parts of Complex Objects in Behavioral Pattern Identification. In: Pal, S.K., Bandyopadhyay, S., Biswas, S. (eds.) PReMI 2005. LNCS, vol. 3776, pp. 720–725. Springer, Heidelberg (2005)
9. Blass, A., Gurevich, Y.: Abstract State Machines Capture Parallel Algorithms. ACM Transactions on Computational Logic 4(4), 578–651 (2003)
10. Bower, J.M., Bolouri, H. (eds.): Computational Modeling of Genetic and Biochemical Networks. MIT Press (2001)
11. http://chu.stanford.edu/
12. De Raedt, L.: Logical and Relational Learning (2008)
13. Gibson, J.J.: The perception of the visual world. Houghton Mifflin (1950)
14. Gibson, J.J.: The ecological approach to visual perception. Lawrence Earlbaum Assoc. Publishers (1979)
15. Goldstein, E.B. (ed.): Blackwell Handbook of Perception. Blackwell Publishers Ltd. (2001)
16. Goldin, D.Q., Smolka, S.A., Attie, P.C., Sonderegger, E.L.: Turing Machines, Transition Systems, and Interaction. Information and Computation 194(2), 101–128 (2004)
17. Goldin, D., Smolka, S., Wegner, P.: Interactive Computation: The New Paradigm. Springer, Heidelberg (2010)
18. Goldin, D.Q., Wegner, P.: Principles of interactive computation. In: Goldin, D., Smolka, S., Wegner, P. (eds.) Interactive Computation: The New Paradigm, pp. 25–37. Springer (2006)
19. Gurevich, Y.: Evolving Algebra 1993: Lipari Guide. In: Bögger, E. (ed.) Specification and Validation Methods, pp. 9–36. Oxford University Press (1995)
20. Gurevich, Y.: Sequential Abstract State Machines Capture Sequential Algorithms. ACM Trans. on Computational Logic 1(1), 77–111 (2000)
21. Gurevich, Y.: Interactive Algorithms 2005. In: Goldin, D., Smolka, S., Wegner, P. (eds.) Interactive Computation: The New Paradigm, pp. 165–181. Springer (2006)
22. Hastie, T., Tibshirani, R., Friedman, J.H.: The Elements of Statistical Learning: Data Mining, Inference, and Prediction, 2nd edn. Springer (2008)
23. Jankowski, A., Skowron, A.: A wistech paradigm for intelligent systems. In: Peters et.al, [54], pp. 94–132

24. Jankowski, A., Skowron, A.: Logic for artificial intelligence: The Rasiowa - Pawlak school perspective. In: Ehrenfeucht, A., Marek, V., Srebrny, M. (eds.) Andrzej Mostowski and Foundational Studies, pp. 106–143. IOS Press, Amsterdam (2008)
25. Jankowski, A., Skowron, A.: Wisdom technology: A rough-granular approach. In: Marciniak, M., Mykowiecka, A. (eds.) Bolc Festschrift. LNCS, vol. 5070, pp. 3–41. Springer, Heidelberg (2009)
26. Keefe, R.: Theories of Vagueness. Cambridge Studies in Philosophy (2000)
27. Kleijn, J., Koutny, M.: A Petri net model for membrane systems with dynamic structure. Natural Computing 8, 781–796 (2009)
28. Liu, J.: Autonomous Agents and Multi-Agent Systems: Explorations in Learning, self-Organization and Adaptive Computation. World Scientific Publishing (2001)
29. Liu, H., Motoda, H.: Feature Extraction, Construction and Selection: A Data Mining Perspective. Springer (1998)
30. Marr, D.: Vision: A Computational Investigation into the Human Representation and Processing of Visual Information. Freeman (1982)
31. Michalski, R.S.: Pattern recognition as knowledge-guided computer induction. Technical Report No. 927. Department of Computer Science. University of Illinois, Urbana-Champaign (1978)
32. Milner, R.: Elements of interaction. Communications of ACM 36, 78–89 (1993)
33. Newell, A.: Unified theories of cognition. Harvard University Press (1990)
34. Nguyen, S.H., Bazan, J.G., Skowron, A., Nguyen, H.S.: Layered Learning for Concept Synthesis. In: Peters, J.F., Skowron, A., Grzymała-Busse, J.W., Kostek, B.z., Świniarski, R.W., Szczuka, M.S. (eds.) Transactions on Rough Sets I. LNCS, vol. 3100, pp. 187–208. Springer, Heidelberg (2004)
35. Nguyen, T.T., Skowron, A.: Rough-granular computing in human-centric infor-mation processing. In: Bargiela, A., Pedrycz, W. (eds.) Human-Centric Information Processing Through Granular Modelling. SCI, vol. 182, pp. 1–30. Springer (2009)
36. Noë, A.: Action in Perception. MIT Press (2004)
37. Omicini, A., Ricci, A., Viroli, M.: The Multidisciplinary Patterns of Interaction from Sciences to Computer Science. In: Goldin, D., Smolka, S., Wegner, P. (eds.) Interactive Computation: The New Paradigme, pp. 359–414. Springer, Heidelberg (2010)
38. Păun, G.: Membrane Computing. An Introduction. Springer (2002)
39. Pawlak, Z.: Information Systems – theoretical foundation. Information Systems 6, 205–218 (1981)
40. Pawlak, Z.: Rough sets. International Journal of Computing and Information Sciences 18, 341–356 (1982)
41. Pawlak, Z.: Rough sets. Theoretical Aspects of Reasoning About Data. Kluwer Academic Publishers (1991)
42. Pawlak, Z.: Concurrent versus sequential the rough sets perspective. Bulletin of the EATCS 48, 178–190 (1992)
43. Pawlak, Z.: Elementary rough set granules: toward a rough set processor. In: Pal, S.K., Polkowski, L., Skowron, A. (eds.) Rough-Neural Computing: Techniques for Computing with Words, pp. 5–13. Springer (2003)
44. Pawlak, Z.: Some Issues on Rough Sets. In: Peters, J.F., Skowron, A., Grzymała-Busse, J.W., Kostek, B.z., Świniarski, R.W., Szczuka, M.S. (eds.) Transactions on Rough Sets I. LNCS, vol. 3100, pp. 1–58. Springer, Heidelberg (2004)
45. Pawlak, Z., Skowron, A.: A rough set approach for decision rules generation. In: Thirteenth International Joint Conference on Artificial Intelligence IJCAI 1993, pp. 114–119. Morgan Kaufmann (1993)

46. Pawlak, Z., Skowron, A.: Rudiments of rough sets. Information Science 177, 3–27 (2007)
47. Pawlak, Z., Skowron, A.: Rough sets: Some extensions. Information Science 177, 28–40 (2007)
48. Pawlak, Z., Skowron, A.: Rough sets and boolean reasoning. Information Science 177, 41–73 (2007)
49. Pedrycz, W., Skowron, A., Kreinovich, V. (eds.): Handbook of Granular Computing. John Wiley & Sons (2008)
50. Peters, J.F., Henry, C.D., Ramanna, S.: Reinforcement learning in swarms that learn. In: Barthes, J.-P., Jain, L., Skowron, A., Sun, R., Morizet-Mahoudeaux, P., Liu, J., Zhong, N. (eds.) Proceedings of the 2005 IEEE/WIC/ACM International Conference on Intelligent Agent Technology, Compiegne, France, September 19-22, pp. 400–406. IEEE Computer Society Press, Los Alamitos (2005)
51. Peters, J.F., Henry, C.D., Ramanna, S.: Reinforcement learning with pattern-based rewards. In: Hamza, M.H. (ed.) Computational Intelligence, Proceedings of the Fourth International IASTED Conference on Computational Intelligence, July 4-6, pp. 267–272. IASTED/ACTA Press, Calgary (2005)
52. Peters, J.F., Lockery, D., Ramanna, S.: Monte-Carlo off-policy reinforcement learning: A rough set approach. In: Nedjah, N., de Mourelle, M.L., Abraham, A., Köppen, M. (eds.) 5th International Conference on Hybrid Intelligent Systems (HIS 2005), Rio de Janeiro, Brazil, November 6-9, pp. 187–192 (2005)
53. Peters, J.F., Henry, C., Ramanna, S.: Rough ethograms: A study of intelligent system behavior. In: Kłopotek, M.A., Wierzchoń, S., Trojanowski, K. (eds.) Intelligent Information Systems. AISC, vol. 31, pp. 117–126. Springer, Heidelberg (2005)
54. Peters, J.F., Skowron, A., Düntsch, I., Grzymała-Busse, J.W., Orłowska, E., Polkowski, L. (eds.): Transactions on Rough Sets VI. LNCS, vol. 4374. Springer, Heidelberg (2007)
55. Poggio, T., Smale, S.: The mathematics of learning: Dealing with data. Notices of the AMS 50(5), 537–544 (2003)
56. Polkowski, L., Skowron, A., Żytkow, J.: Rough foundations for rough sets. In: Proceedings of The Third International Workshop on Rough Sets and Soft Computing (RSSC 1994), November 10-12, pp. 142–149. San Jose State University, CA (1994); see also: Rough foundations for rough sets. In: Lin, T.Y., Wildberger, A.M. (eds.): Soft Computing, Simulation Councils, Inc., San Diego, pp. 55–58 (1995)
57. Polkowski, L., Skowron, A.: Rough mereology: a new paradigm for approximate reasoning. International Journal of Approximate Reasoning 15, 333–365 (1997)
58. Read, S.: Thinking about logic: An introduction to the philosophy of logic. Oxford University Press (1995)
59. Ray, D.: A Game-Theoretic Perspective on Coalition Formation. Oxford University Press (2007)
60. Shehory, O., Kraus, S.: Methods for task allocation via agent coalition formation. Artificial Intelligence 101, 165–200 (1998)
61. Simons, H.: The Science of the Artificial, 2nd edn. MIT Press (1982)
62. Skowron, A.: Rough sets and vague concepts. Fundamenta Informaticae 64(1-4), 417–431 (2005)
63. Skowron, A., Stepaniuk, J.: Tolerance approximation spaces. Fundamenta Informaticae 27, 245–253 (1996)
64. Skowron, A., Stepaniuk, J.: Informational granules and rough-neural computing. In: Pal, S.K., Polkowski, L., Skowron, A. (eds.) Rough-Neural Computing: Techniques for Computing with Words, pp. 43–84. Springer (2003)

65. Skowron, A., Stepaniuk, J.: Hierarchical modelling in searching for complex patterns: constrained sums of information systems. Journal of Experimentaland Theoretical Artificial Intelligence 17(1-2), 83–102 (2005)
66. Skowron, A., Stepaniuk, J.: Approximation Spaces in Rough Granular Computing. Fundamenta Informaticae 100, 141–157 (2010)
67. Skowron, A., Stepaniuk, J.: Rough Granular Computing Based on Approximation Spaces (submitted)
68. Skowron, A., Suraj, Z.: Rough sets and concurrency. Bulletin of the Polish Academy of Sciences 41, 237–254 (1993)
69. Skowron, A., Suraj, Z.: Discovery of concurrent data models from experimental tables: A rough set approach. In: Proceedings of First International Conference on Knowledge Discovery and Data Mining, pp. 288–293. AAAI Press (1995)
70. Skowron, A., Szczuka, M.: Toward interactive computations: A rough-granular approach. In: Koronacki, J., Wierzchon, S., Ras, Z., Kacprzyk, J. (eds.) Commemorative Volume to Honor Ryszard Michalski, pp. 1–20. Springer (2009)
71. Skowron, A., Wang, H., Wojna, A., Bazan, J.G.: A Hierarchical Approach to Multimodal Classification. In: Ślęzak, D., Yao, J., Peters, J.F., Ziarko, W.P., Hu, X. (eds.) RSFDGrC 2005. LNCS (LNAI), vol. 3642, pp. 119–127. Springer, Heidelberg (2005)
72. Skowron, A., Wasilewski, P.: Information Systems in Modeling Interactive Computations on Granules. In: Szczuka, M., Kryszkiewicz, M., Ramanna, S., Jensen, R., Hu, Q. (eds.) RSCTC 2010. LNCS, vol. 6086, pp. 730–739. Springer, Heidelberg (2010)
73. Skowron, A., Wasilewski, P.: An Introduction to Perception Based Computing. In: Kim, T.-H., Lee, Y.-H., Kang, B.-H., Ślęzak, D. (eds.) FGIT 2010. LNCS, vol. 6485, pp. 12–25. Springer, Heidelberg (2010)
74. Skowron, A., Wasilewski, P.: Information systems in interactive computing. In: Wakulicz-Deja, A. (ed.) Proceedings of Desision System Support Conference, Instituite of Informatics, Zakopane, December 1–4, pp. 33–45. Silesian University (2010)
75. Skowron, A., Wasilewski, P.: Information systems in modeling interactive computations on granules. Theoretical Computer Science 412(42), 5939–5959 (2011)
76. Skowron, A., Wasilewski, P.: Toward interactive rough–granular computing. Control & Cybernetics 40(2), 1–23 (2011)
77. Skowron, A., Stepaniuk, J., Peters, J.F.: Towards discovery of relevant patterns from parametrized schemes of information granule construction. In: Inuiguchi, M., Hirano, S., Tsumoto, S. (eds.) Rough Set Theory and Granular Computing, pp. 97–108. Springer (2003)
78. Skowron, A., Stepaniuk, J., Peters, J., Swiniarski, R.: Calculi of approximation spaces. Fundamenta Informaticae 72(1-3), 363–378 (2006)
79. Słowiński, R., Vanderpooten, D.: Similarity relation as a basis for rough approximations. In: Wang, P. (ed.) Advances in Machine Intelligence and Soft Computing, vol. 4, pp. 17–33. Duke University Press, Duke (1997)
80. Stepaniuk, J.: Rough-Granular Computing in Knowledge Discovery and Data Mining. Springer (2008)
81. Sutton, R.S., Barto, A.G.: Reinforcement Learning: An Introduction. The MIT Press (1998)
82. Sycara, K., Norman, T.J., Giampapa, J.A., Kollingbaum, M., Burnett, C., Masato, D., McCallum, M., Strub, M.H.: Agent Support for Policy-driven Collaborative Mission Planning. The Computer Journal 53, 528–540 (2010)
83. Ślęzak, D., Toppin, G.: Injecting domain knowledge into a granular database engine – A position paper. In: CIKM 2010 (2010)

84. Ślęzak, D., Wasilewski, P.: Granular Sets Foundations and Case Study of Tolerance Spaces. In: ICSR 2006. LNCS (LNAI), vol. 4039, pp. 461–469. Springer, Heidelberg (2007)
85. Thagard, P.: Mind: Introduction to Cognitive Science, 2nd edn. MIT Press (2005)
86. Turing, A.: Computing machinery and intelligence. Mind 59, 33–60 (1950)
87. Vapnik, V.: Statisctical Learning Theory. John Wiley & Sons (1998)
88. Vapnik, V.: Learning Has Just Started an interview with Prof. Vladimir Vapnik by R. Gilad- Bachrach (2008), http://learningtheory.org
89. Wang, H., Lewis, M., Chien, S., Scerri, P., Velagapudi, P., Sycara, K., Kane, B.: Teams organization and performance in multi-human/multi-robot teams. In: 2010 IEEE International Conference on Systems, Man, and Cybernetics, pp. 1617–1623 (2010)
90. Wang, Y., Kinsner, W., Zhang, D.: Contemporary Cybernetics and its Faces of Cognitive Informatics and Computational Intelligence. IEEE Trans. on System, Man, and Cybernetics (B) 39(4), 1–11 (2009)
91. Wasilewski, P., Ślęzak, D.: Foundations of Rough Sets from Vagueness Perspective. In: Hassanien, A.E., Suraj, Z., Ślęzak, D., Lingras, D.P. (eds.) Rough Computing. Theories, Technologies and Applications. Information Science Refer., pp. 1–37 (2008)
92. Wegner, P.: Why interactions is more powerful than algorithms. Communications of ACM 40, 81–91 (1997)
93. Wegner, P.: Interactive foundation of computing. Theoretical Computer Science 192, 315–351 (1998)
94. Zadeh, L.A.: Fuzzy sets and information granularity. In: Gupta, M., Ragade, R., Yager, R. (eds.) Advances in Fuzzy Set Theory and Applications, pp. 3–18. North-Holland Publishing Co. (1979)
95. Zadeh, L.A.: From computing with numbers to computing with words – from manipulation of measurements to manipulation of perceptions. IEEE Transactions on Circuits and Systems 45, 105–119 (1999)
96. Zadeh, L.A.: A new direction in AI-toward a computational theory of perceptions. AI Magazine 22(1), 73–84 (2001)

Chapter 11
Overlapping, Rare Examples and Class Decomposition in Learning Classifiers from Imbalanced Data

Jerzy Stefanowski

Abstract. This paper deals with inducing classifiers from imbalanced data, where one class (a minority class) is under-represented in comparison to the remaining classes (majority classes). The minority class is usually of primary interest and it is required to recognize its members as accurately as possible. Class imbalance constitutes a difficulty for most algorithms learning classifiers as they are biased toward the majority classes. The first part of this study is devoted to discussing main properties of data that cause this difficulty. Following the review of earlier, related research several types of artificial, imbalanced data sets affected by critical factors have been generated. The decision trees and rule based classifiers have been generated from these data sets. Results of first experiments show that too small number of examples from the minority class is not the main source of difficulties. These results confirm the initial hypothesis saying the degradation of classification performance is more related to the minority class decomposition into small sub-parts. Another critical factor concerns presence of a relatively large number of borderline examples from the minority class in the overlapping region between classes, in particular for non-linear decision boundaries. The novel observation is showing the impact of rare examples from the minority class located inside the majority class. The experiments make visible that stepwise increasing the number of borderline and rare examples in the minority class has larger influence on the considered classifiers than increasing the decomposition of this class. The second part of this paper is devoted to studying an improvement of classifiers by pre-processing of such data with re-sampling methods. Next experiments examine the influence of the identified critical data factors on performance of 4 different pre-processing re-sampling methods: two versions of random over-sampling, focused under-sampling NCR and the hybrid method SPIDER. Results show that if data is sufficiently disturbed by borderline and rare examples SPIDER and partly NCR work better than over-sampling.

Jerzy Stefanowski
Institute of Computing Science, Poznań University of Technology, ul. Piotrowo 2, 60–965 Poznań, Poland
e-mail: jerzy.stefanowski@cs.put.poznan.pl

11.1 Introduction

Supervised learning of classifiers from examples is one of the main tasks in machine learning and data mining. Many approaches based on different principles have been introduced in last decades, for reviews, see e.g. [34, 41]. However, their usefulness for obtaining high predictive accuracy in real life data depends on different factors, including also difficulties of the learning problem and its data characteristics. *Class imbalance* is one of the sources of these difficulties.

A data set is considered to be imbalanced if one of target classes contains much smaller number of examples than the other classes. The under-represented class is called the *minority class*, while the remaining classes are referred to as *majority classes*.

Many real life problems are characterized by a highly imbalanced distribution of examples in classes. Typical examples are rare medical diagnosis [26], recognition oil spills in satellite images [36], detecting specific astronomical objects in sky surveys [45] or technical diagnostics of equipment failures. Moreover, in fraud detection, either in card transactions [17] or in telephone calls [5] the number of legitimate transactions is much higher than the number of fraudulent ones. Similar situations occur either in direct marketing where the response rate class is usually very small in most marketing campaigns [39] or information filtering where some important categories contain few messages only [38]. Other practical problems are also discussed in [11, 18, 19, 62].

If imbalance in the class distribution is extensive, i.e. some classes are *strongly under-represented*, then the typical learning methods do not work properly. An even class distribution is often assumed (also non explicitly) and the classifiers are "somehow biased" to focus searching on the more frequent classes while "missing" examples from the minority class. As a result constructed classifiers are also biased toward recognition of the majority classes and they usually have difficulties (or even are unable) to classify correctly new objects from the minority class. In [38] authors described an information retrieval system, where the minority class (being of a primary importance) contained only 0.2% of all examples. Although the classifiers achieved the overall accuracy close to 100%, they were useless because they failed to deliver requested documents from this class. Similar degradation of classifier's performance for the minority class was also reported for other imbalanced problems, see e.g. [9, 26, 29, 35, 43, 62].

Learning from imbalanced data is considered by some researchers as one of the most challenging topics in machine learning and data mining [65]. It has received growing research interest in the last decade and several specialized methods have already been proposed, see [11, 12, 18, 62] for a review. These methods are usually categorized in two groups:

- The first group includes classifier-independent methods that rely on transforming the original data to change the distribution of classes, e.g., by re-sampling.
- The other group involves modifications of either a learning phase of the algorithm, classification strategies, construction of specialized ensembles or adaptation of cost sensitive learning.

This paper concerns the first group as these methods are more universal and they can be used in a pre-processing stage before applying many learning algorithms. While the other group includes many quite specialized methods based on different principles. For instance, many authors changed search strategies, evaluation criteria or parameters in the internal optimization of the algorithm, see e.g [23, 27, 31, 61, 62, 63]. A survey of special changes in ensembles is given in [21], while adaptations to cost sensitive learning are reviewed in [18].

Before focusing our interest on some of pre-processing methods, we want to ask a more general question about the nature of class imbalance problem and to study the key properties of data distribution which make learning classifiers so difficult. A small number of examples in the minority class is not the only source of difficulties for classifiers. Recent works also suggest that there are other factors that contribute to difficulties. The most well known studies with artificial data are the works of Japkowicz [29, 30], who showed that simple class imbalance ratio was not the main difficulty. The degradation of performance was also related to other factors, mainly to decomposition of the minority class into many sub-clusters with very few examples. The rare sub-concepts correspond to, so called, *small disjuncts*, which lead to classification errors more often than examples from larger parts of the class [20]. Other researchers also explored the effect of *overlapping* between imbalanced classes – more recent experiments on artificial data with different degrees of overlapping also showed that overlapping was more important than the overall imbalance ratio [24, 47].

However, the authors of the above mentioned papers considered these factors independently each other. It could be worth to investigate them occurring together in the data also in presence of other factors. In earlier studies Stefanowski and his cooperators have noticed that many imbalanced data (e.g. coming from UCI repository [2] and used in many papers on new approaches to class imbalance) contain also minority class examples located inside the majority class [40, 42]. They could treated as *outliers* (in particular, if they are single examples surrounded by many examples from majority classes) or *rare cases* (if they are not single ones). They should not be considered as noise as they are too rare and too precious for the minority class. According to the best knowledge this kind of rare examples has not been examined in studies with imbalanced data. Furthermore, it could be interesting to consider the role of changing decision boundary between classes from linear to non-linear shapes. Let us remind that rather simpler shapes were previously studied [24, 29].

To sum up, studying the role of these factors in class imbalance is still an open research problem. Therefore, the main aim of this study is to experimentally examine which of these factors are more critical for the performance of the classifier. Carrying out such experiments requires preparing a new collection of artificial data sets which are affected by the above mentioned factors. Proposing such data sets is another sub-aim of this paper.

Then, assuming that performance of classifiers could be deteriorated by these data factors one could examine competence of pre-processing methods to deal with particular factors.

In this paper we are particularly interested in *focused* (also called *informed*) *re-sampling* methods, which modify the class distribution taking into account local characteristics of examples. Representative such methods are SMOTE for selective over-sampling of the minority class [9], one side sampling [35] and NCR for removing examples from the majority classes [37] or hybrid SPIDER method [54].

Therefore, an experimental comparison of chosen focused re-sampling methods and simpler random replication of the minority class methods applied to previously generated data sets and establishing competence of these methods for dealing with particular data factors are the next aims of this paper.

The following paper contains many new experimental results (in particular in studying the role of data factors). However, its also summarizes some results coming from co-operation with other colleagues, in particular concerning SPIDER methods and already published in [42, 55].

The paper is organized as follows. Section 2 describes main evaluation measures used for imbalanced data. The review of related research with data factors in imbalanced data is given in Section 3. Then, the generation of artificial data sets is presented in Section 4. The next section contains results of experiments study of the influence of data critical factors on the tree, rule based and k-NN classifiers. In Section 6 the most related focused pre-processing methods, including SPIDER, are briefly presented. Their comparative experimental evaluation is summarized in Section 7. The paper concludes with a discussion in Section 8.

11.2 Evaluation Measures for Learning Classifiers from Imbalanced Data

Imbalanced data constitutes a problem not only when inducing a classifier, but also when evaluating its performance. The overall classification accuracy is not the only and the best criterion characterizing performance of a classifier in case of class imbalance [62].

As the overall classification accuracy is biased towards the majority classes, in most of the studies on imbalanced data, measures defined for two-class classification are considered, where typically the class label of the minority class is called positive and the class label of the majority class is negative [18]. Even if data contains more majority classes the classifier performance on these classes could be aggregated into one negative class. Therefore, the performance of the classifiers is presented in a confusion matrix as in Table 1.

Table 1 Confusion matrix for performance evaluation

	Predicted Positive	**Predicted Negative**
True Positive	TP	FN
True Negative	FP	TN

From the confusion matrix, apart from other more elaborated measures (see e.g. reviews as [18]), one can construct simple metrics concerning recognition of the positive (minority) and negative (majority) classes:

$$TruePositiveRate = TP/(TP+FN)$$

$$TrueNegativeRate = TN/(TN+FP)$$

$$FalsePositiveRate = FP/(TN+FP)$$

$$Precision = TP/(TP+FP)$$

True Positive Rate is called *Sensitivity* (also Recall) while True Negative Rate is referred to *Specificity*. As the improvement of recognizing the minority class is associated with changes of recognizing other majority classes, aggregated measures are considered to characterize the performance of the classifiers. First of all, several authors use the *ROC (Receiver Operating Characteristics) curve* analysis [16]. A ROC curve is a graphical plot of a true positive rate (sensitivity) as a function of a false positive rate (1 − specificity) along different threshold values characterizing overall performance of a studied classifier. The quality of the classifier performance is reflected by the area under a ROC curve (so called AUC measure) [11, 62]. AUC varies between 0 and 1. Its larger values indicate better classifier performance. Although AUC is a very popular tool, some researchers have showed that it has some limitations as in the case of highly skewed data sets it could lead to an overoptimistic estimation of the algorithm's performance. Thus, other proposals include Precision Recall Curves [15] or other special cost curves (see their review in [18]).

One can also use simpler measures to characterize classifiers, in particular if they have a purely deterministic prediction (see discussions on applicability of ROC analysis in [61]). Kubat and Matwin [35] proposed to use the geometric mean of sensitivity an specificity defined as a:

$$G\text{-}Mean = \sqrt{Sensitivity \cdot Specificity},$$

This measure relates to a single point on the ROC curve and it key idea is to maximise the recognition of each of minority and majority classes while keeping these accuracies balanced. An important, useful property of the G-Mean is that it is independent of the distribution of examples between classes [24]. An alternative criterion aggregating precision and recall is F measure; for discussion of its properties see e.g. [18].

11.3 Earlier Studies with Data Factors in Class Imbalance

In this section we discuss earlier, related works on studying data properties which influence learning classifiers from imbalanced data sets.

First, one can notice that the majority of researchers proposing new approaches to handle class imbalance validated them in experiments conducted mainly on real-life data sets (usually imbalanced data coming from the UCI Machine Learning Repository [2]). Moreover, other comparative studies of various basic classifiers or pre-processing methods are carried out in a similar way on such data sets, see e.g. such comprehensive comparative studies [3, 59]. However, working with artificial data sets, where it is possible to change their characteristics in the controlled way, is more valuable if someone attempts to study more precisely the impact of a chosen factor characterizing distributions of examples.

Several experimental studies have already showed that the performance of standard classifiers decreased in imbalanced data. However, some researchers hypothesized, that the *class imbalance ratio* (i.e. too low cardinality of the minority class referred to the total number of examples or to the majority class) is not necessarily the only, or main, problem causing this performance decrease and dealing only with it may be insufficient for improving classification results. In other words, besides this imbalanced ratio, data could be accompanied with other factors, which in turn cause the degradation of classification performance.

Japkowicz and her co-operators focused on *within-class imbalance*, i.e. target concepts (classes) were decomposed into sub-concepts [30]. To check the influence of increasing the level of decomposition she and her co-authors carried our many experiments with artificially generated data. Let us describe their construction just to have a reference point for our further solutions. Three parameters were controlled: the size of the training set, the imbalance ratio, and so called *degree of concept complexity* (understood as decomposition of the class into a number of subclasses). Two classes the minority vs. the majority class were considered only and each of data sets was generated in one-dimension interval. This input interval was divided into a number of subintervals of the same size (up to five), each associated with a different class label. The examples were uniformly distributed within subintervals. The degree of complexity corresponds to the number of alternating subintervals. For these assumption 27 data sets were generated with various combinations of the above mentioned three parameters. Following similar assumptions they also generated additional data sets in five-dimensional space, where an alternance of classes was modelled by separate clusters.

Then, C4.5 tree and multi layered perceptron (MLP) with back propagation algorithms were run over these data sets. Their results showed that imbalance ratio did not cause the degradation of classifiers' performance so much as increasing degree of complexity. The worst classification results were obtained for the highest decomposition of classes (e.g. into 5 parts) in particular existing with too small number of examples. Their main result is that "the true nature of the class imbalance problem (...) is only if the size of the small class is very small with respect to the concept complexity; i.e. it contains very small subclusters". On the other hand, this also means that in much larger data where subclusters could be represented by a reasonable number of examples, the imbalance ratio alone will not decrease so much the classification performance [30].

According to Japkowicz, if the such within-class imbalanced sub-concepts contain quite a small number of minority class examples it is associated with the problem of *small disjuncts* while building classifiers – which was originally introduced by Holte in standard (balanced) learning of symbolic classifiers [20]. Briefly speaking, a classifier learns a concept by generating disjunct forms (e.g. rules of tree) to describe it. Small disjuncts are these parts of the learned classifier which cover a too small number of examples [20, 62]. It has been observed in the empirical studies that small disjuncts contribute to the classification error more than larger disjuncts. In case of fragmented concepts (in particular in the minority class) the presence of small disjunct arises [18].

As a practical consequence special approaches to handle the problem of small disjuncts of imbalanced concepts were proposed in [29, 30]. They are based on specialized over-sampling of minority class, i.e. a required number of examples are randomly replicated until balancing majority and minority class in the certain degree. By appropriate increasing the amount of the smaller class the classifiers learned from such modified data are less sensitive to original rare concepts. An example of using cluster analysis to identify the sub-concept and their random over-sampling is described in Section 11.6. The impact of small disjuncts was also further studied by other researchers, see e.g. [28, 46]. In particular, additional experiments with applying other classifiers on the artificial data constructed in the above mention way showed that decision trees were the most sensitive to the small disjuncts, then the next was MLP and support vector machines were the less sensitive[1].

Recently some researchers have also focused on different factors characterizing data distributions. Prati et al studied the role of *overlapping* between minority and majority classes [47]. They generated artificial data sets where the minority and the majority class were represented by two clusters in five dimensional space (examples where generated around centroids following Gaussian distribution). Two parameters were changed: the imbalance ratio, and the distance between centroids – so classes could be moved from clear separation to high overlapping. Using C4.5 classifier and AUC criterion they showed that increasing the overlapping ratio was more responsible for decreasing AUC results than decreasing cardinality of the minority class (for some data AUC decreased from 0.99 to 0.5). Another observation, consistent with intuition, was that for clearly separate and distant clusters the classification measures did not decrease even with high under-representation of the minority class.

Then, influence of increasing overlapping was more precisely examined in [24]. Garcia et al. generated two-dimensional data sets with two classes separated by a line orthogonal to one of the axis. Depending on the amount of overlapping examples of the majority class were uniformly generated inside the minority class part in a stepwise way moving from the decision boundary until covering completely minority class. Garcia et al. assumed a fixed size of data and changed the overlapping amount for a given imbalance ratio and vice versa. Results of experiments with 6 different classifiers showed that increasing overlapping more degraded their performance (with respect to TPR and TPN) than changing the imbalance ratio. Moreover, in the other experiments they fixed the amount of overlapping and changed the distribution of the minority examples by increasing their number in the overlapping area. In this way they achieved balance between classes in this boundary, and then

[1] These results are also summarized in the report [8]

the minority class dominated the majority one. Again the results of experiments confirmed that increasing such a local imbalance ratio and the size of the overlapping area were more influential than changing the overall imbalance ratio. However, these factors influenced in a different way performance of particular classifier. For instance k nearest neighbor classifiers (K-NN) was the most sensitive to changes in the local imbalance region. Naive Bayesian, MLP and J4.8 were better working in the dense overlapping region. These conclusions have been later verified in additional experiments (more focusing on performance of K-NN and other evaluation measures), see [25].

Prati et al. have recently come back to studying the overlapping in class imbalance [4]. Comparing to their previous work [47] where they identified that performance degradation was not solely caused by class imbalances, but it was strongly related to the degree of class overlapping, in the new work, they decided to investigate the usefulness of five different re-sampling methods on the same difficult artificial data sets. The chosen methods were: popular random-over sampling, random under-sampling, Nearest Cleaning Rule (NCR) [37], SMOTE and SMOTE + ENN [9]. We will briefly describe them in further section 6. Now we can say that the main conclusion was that appropriate balancing training data usually led to a performance improvement of C4.5 classifiers for highly imbalanced data sets with highly overlapped classes. However, the improvements depends on the particular method and the overlapping degree. For the highest degree of overlapping it was not clear which method was the best (NCR worked there quite well). Results for other overlapping showed that over-sampling methods in general, and Smote-based methods in particular, were more effective than under-sampling. Moreover, the Smote-based methods were able to achieve the best performance even for the most skewed distributions. Then, the data cleaning step used in the Smote + ENN seemed to be especially suitable in situations having a higher degree of overlapping. The quite good performance of SMOTE over-sampling integrated with data cleaning (as edited nearest rule ENN or Tomek Links [58]) was also confirmed in other experiments on many UCI real data sets [3].

Prati et al. have also noticed in their conclusions that ”it is also worthwhile to consider the generation of artificial data sets where the distribution of examples of the minority class is separated into several small clusters” [4].

Finally, quite a few researchers noticed that the other factor which could influence degradation of classifiers performance on imbalanced data could be *noisy examples* [1, 60]. Traditionally noise in supervised learning is understood as an (random) *error in labeling examples* (i.e. an example is assigned to wrong class) or erroneous values of attributes describing some examples [7]. These researchers wrote that existing methods for handling imbalanced data sets were studied under an assumption saying that the input data are noise-free or noise in the data sets is not significant. They claimed that real-world data are rarely perfect and can often suffer from corruptions that may impact decision of models created from these data. An investigation of both class and attribute noise in case of typical machine learning (i.e. balanced data) was conducted by many researchers which conclusion that the presence of noise can be harmful to a classifier, in particular when it is applied to previously unseen or testing examples. In case of imbalanced data authors of [1, 60] proposed to identify noisy examples and remove them from the input data. However, their experiments were either conducted over UCI typical data sets or specific software data [60]. Moreover, they used special classification filters with identify so called *mislabeled examples* [7] by sophisticated ensembles. In these approaches the meaning of a mislabeled example has a broader sense as besides random errors it also includes outliers and other nontypical class representatives [22]. In this paper, a different view on such examples is presented.

11.4 Generation of New Artificial Data Sets

First, we will discuss assumptions for preparing new data sets in our experimental study.

Let us summarize that authors of related works indicated the role of following factors characterizing data distribution:

- decomposition of classes into sub-concepts,
- too low number of examples in sub-concepts (small disjuncts),
- high overlapping between classes.

One can notice that these authors studied the impact of single factors without considering other ones in the same experimental setup. Here, we want to consider all of them occurring together.

Besides the above mentioned factors we decide to consider two additional factors:

- different, more difficult shapes of the decision boundaries between classes
- and additional type of examples belonging to the minority class.

Explaining the second factor let us try to categorize types of minority class members. The examples located more deeply inside this class (even its sub-concepts) could be treated as *safe* ones. Such examples could be easier for correct recognition as they are surrounded by examples from the same class. Then, some other examples could be called *borderline* examples, if they are located inside the overlapping region. They are more unsafe or difficult to be learned as they occur closer to the decision boundary between classes with mixed distribution of majority class neighbors and possible noise could more influence changing the classification decision.

Moreover, it would be worthy to distinguish yet another type of so called *rare examples*. These examples are located in the majority class region and being distant enough to the decision boundary to be distinguished from borderline examples. In some of earlier experiments Stefanowski and his co-operators analysed local neighborhoods of minority class examples (see e.g. [40]). This analysis of local neighborhood was done with k-NN classifiers. An example was treated as safe, if its was correctly re-classified by its closest k neighbors. If the ratio neighbors from opposite classes was similar which could lead to mis-classification of the example it was treated as a borderline example. Then, if all its neighbours belong to opposite classes it was identified as an outlier. Moreover we noticed that some minority examples locally created pairs or sometimes triples also more distant from borderline examples. Analysing in this way several UCI data sets we noticed that they could be difficult with respect to recognizing the minority class as they contained much less safe examples than others. Moreover, the number of outliers or rare pairs/triples was sometimes relatively high. For instance, in cleveland data the minority class contained 35 examples with 22 outliers or rare examples and 13 borderline ones. Similar situation ocured for a few data while some other data sets, e.g. haberman or ecoli, contained more bordeline examples than outlier ones (e.g. haberman has 51 and 20 respectively with 10 safe examples only). As the number of outliers or rare examples is quite high comparing to the size of the minority class we claim that they could be precious and cannot be just skipped while building classifiers. In our point of view they are not treated as simple noise but rather less typical rare cases of the scattered minority class. We will further call them *rare examples*[2].

To sum up, the following factors are chosen to be present in new artificial data sets considered in our experiments:

- decomposition of the minority class into sub-parts (subclusters, etc.),
- size of the overlapping region between classes (majority class examples are generated into the minority class inside the ”borderline zone” of the given width and it is parametrized by the relative number of examples from the minority class that are located in this region),
- presence of rare examples (also parametrized by the relative number of examples from the minority class),
- linear vs. non-linear shape of decision boundaries,
- imbalance ratio (denoted as $i : j$, where i represents the minority class and j the majority one),
- total number of learning examples.

Similarly to other researchers we chose binary classification problems (the minority vs. the majority class) with examples randomly (either uniformly or not) distributed in the two-dimensional space (both attributes were real-valued). Following the literature on experiments with the factors influencing the performance of classifiers, we decided to prepare several artificial data sets in order to control these factors. We considered three different shapes of the minority class: *subclus*, *clover* and *paw*.

In *subclus*, examples from the minority class are located inside rectangles all surrounded uniformly by the majority class. The examples of the minority class are also uniformly distributed within its sub-region. This shape is a kind of two-dimensional generalization of data from related works on data decomposition and small disjuncts [29]. Fig. 1 shows such a shape with 3 subclusters; two zoom windows focus a reader attention on the exemplary borders.

Then, the next shape, called a *clover*, represents a more difficult, non-linear setting, where the minority class resembles a flower with elliptic petals. We decided to analyse two types of such clover shapes. In the first type the examples of majority class were also distributed inside the elliptic shapes fitted within the minority class

[2] In this sense it could be close to rare cases as discussed in the second section of G.Weiss study [62]. Although it is still class imbalance problem not a case of very rare data as sometimes considered in the context of one-class-learning

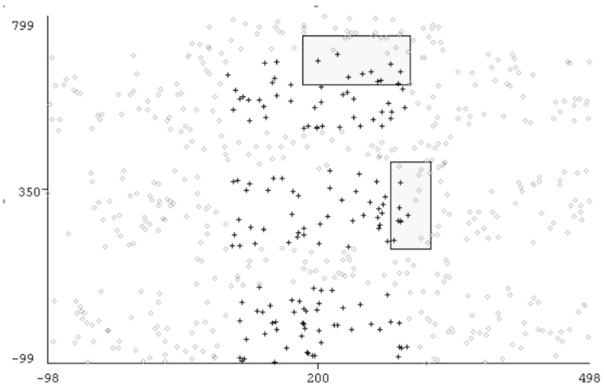


Fig. 1 Subclass data set – a minority class is decomposed into 3 parts (subclusters)

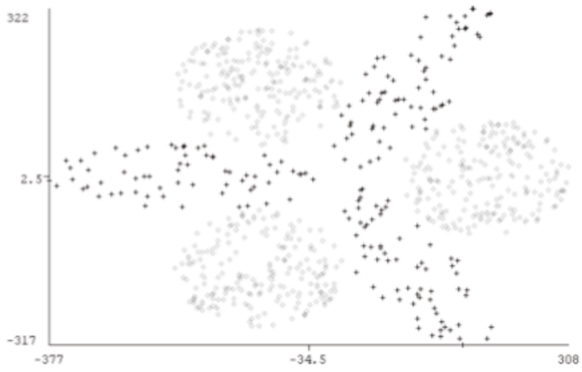


Fig. 2 Clover data set with 3 elements and sub-clusters of majority classes

”petals”. Figure 2 shows just such a *clover* with 3 petals. Then, we created another clover versions where examples from the majority class were uniformly distributed in all the free parts – see at Fig. 5 for *clover* with 5 petals (as this shape will be mainly used in further experiments – see Section 11.7 – examples are represented by points marked with different symbols).

Finally, in *paw* the minority class is decomposed into 3 elliptic sub-regions of varying cardinalities, where two subregions are located close to each other, and the remaining smaller sub-region is separated – two representatives of such shapes are showed in Fig. 3 and Fig. 4. In case of *02a* example the majority class are generated closer to the minority class regions and they are distributed more uniformly, while *02b* is more similar to *clovers* where the majority class is arranged in elliptical shapes. However, in both cases the majority class was also generated within some elliptical shapes. We also constructed another versions of *paw* where examples from the majority class are uniformly distributed in the allowed area - see an illustrative

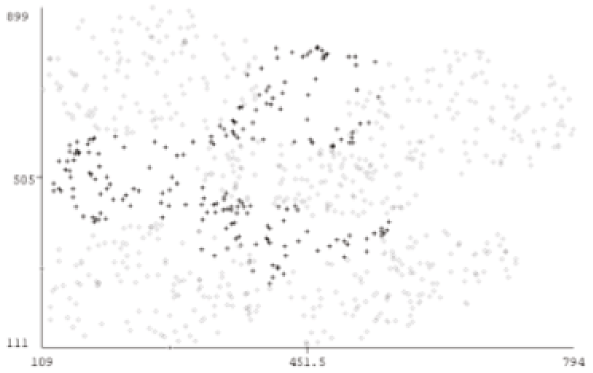


Fig. 3 Paw data set - version 02a

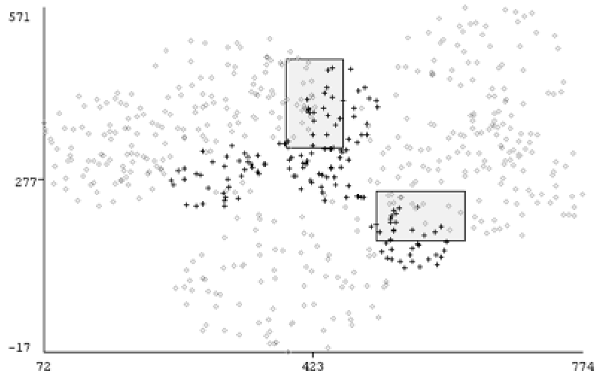


Fig. 4 Another paw data set - version 02b

example in Fig 6. We constructed such *paw* figure as it should better represents real-life data than the *clover*. Moreover, both *clover* and *paw* should be more difficult to learn than simple circles (or spheres) that were considered in some related works.

We generated a large collection of data sets with different numbers of examples (ranging from 200 to 1200) and imbalance ratios (from 1:3 to 1:9). Additionally, following Japkowicz's research on data complexity and splitting data shapes into interleaved sub-parts [29], we considered a series of the *subclus* and *clover* shapes with the number of sub-regions ranging from 1 to 5, and from 2 to 5 respectively.

Technically, this generator was implemented by Krzysztof Kaluzny in Java as a program compatible with WEKA platform; for more details see [32].

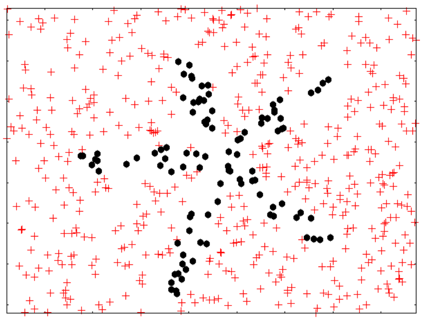


Fig. 5 Clover data set

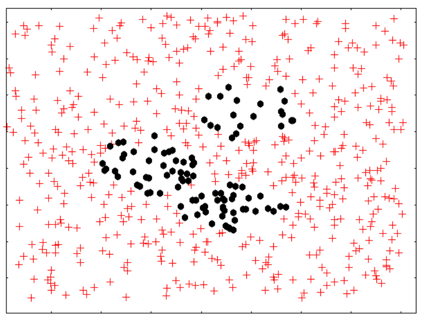


Fig. 6 Paw data set

11.5 Experimental Analysis of Influence of Critical Factors on Classifiers

In experiments three different classifiers were applied to the generated data sets (as presented in the previous section). *K*–nearest neighbod (K-NN), tree- and rule–based classifiers were chosen following the related experiments. K-NN was parametrized with $k = 3$ (we also tested 1 neighbour). While looking for these nearest neighbours, the distance is calculated with HVDM distance [64], i.e. aggregation of the Euclidean distance metric for numerical features and the Value Distance Metric [14] for the qualitative attributes. Decision trees were induced by Quinlan algorithm [44] - version J4.8 available in WEKA. Then, rules were generated by Ripper algorithm [13] (also JRip implementation from WEKA). Trees and rule were run without pruning to get more precise descriptions of the minority class. Evaluation measures were estimated by stratified 10 fold-cross validation. We focus mainly on the sensitivity (TPR) to study recognition of the minority class and AUC as a secondary criterion.

Due to the space limit, we are not able to present the complete results of all experiments but focus on the most interesting ones. For more details the reader is referred to the report describing much more experimental results [53].

In first experiments we studied the impact of the imbalance ratio combined with the size of data (i.e. total number of examples varied from 1200 to 200). Other critical factors were not considered, i.e. there was no overlapping or noisy examples. Let us only comment that our results were consistent with the observations reported in [30]. For the number of examples greater than 400 examples there was no significant influence of decreasing the imbalance ratio. Small decreases of sensitivity and partly AUC measures was observed only for very small cardinality of the data (smaller than 200) and a very high imbalanced ratio (1:11). It concerned all studied classifiers.

We also noticed that non-linear shapes of decision boundaries (as visible in *clover* or 02a, 02b data) were more difficult to recognize than linear rectangles *subclass* – see also Table 3.

Table 2 Influence of decomposing the minority class into sub-concepts on the sensitivity measure for K-NN classifier: Subclass data set. Two imbalance ratios and five different cardinalities of data sets are reported in columns.

Number of	1:5			1:9		
subclusters	600	400	200	600	400	200
2	0.82	0.8	0.78	0.78	0.76	0.45
3	0.78	0.72	0.70	0.66	0.74	0.25
4	0.75	0.70	0.68	0.64	0.50	0.15
5	0.73	0.68	0.42	0.58	0.45	0.11
6	0.64	0.62	0.36	0.42	0.32	0.10

Table 3 Influence of non-linear decision boundaries on AUC measure. Size of data – 1200 examples and 3 different imbalance ratios.

Type of	Trees			Rules			KNN		
data	1:1	1:3	1:5	1:1	1:3	1:5	1:1	1:3	1:5
02a	0.95	0.85	0.68	0.94	0.91	0.89	0.94	0.92	0.9
02b	0.95	0.92	0.89	0.95	0.92	0.85	0.95	0.93	0.9
subclass3	0.98	0.97	0.96	0.96	0.94	0.92	0.97	0.96	0.94
subclass5	0.98	0.96	0.94	0.96	0.92	0.90	0.96	0.96	0.94
clover3	0.94	0.93	0.92	0.94	0.92	0.90	0.96	0.94	0.92
clover5	0.93	0.92	0.89	0.91	0.88	0.84	0.94	0.92	0.90

In the next phase of experiments we studied more precisely the impact of increasing the decomposition of the minority class into sub–parts. Generally speaking, the obtained results were also consistent with earlier related research, in particular works of Japkowicz – increasing the number of sub-regions of the minority class combined with decreasing the size of a data set degraded the performance of a classifier [28, 29, 30]. For illustration see Table 2, where for two imbalance ratios (1:5 and 1:9) and stepwise decrease of examples (from 600 to 200) we divide the rectangle of the minority class into appropriate subclusters (of the same size). One can notice that for smaller number of examples increasing the number of subclusters degraded the values of sensitivity much larger than changing the imbalance ratio. If the number of subclusters is higher than 4 and the number of examples is no greater than 400 examples, they could be seen as small sub-regions (e.g. for 1:5 the total number of 66 minority examples are divided into rare areas having less than 15 examples comparing to 333 majority examples – which could refer to the idea of small disjuncts [30]). Decreasing the total number of examples to 200 makes the problem definitely more difficult. The tree and rule classifiers also decreased their performance for these *subclass* shapes.

Table 4 Influence of decomposing the minority class on the sensitivity of tree classifier: Clover data set. Data size – 600 and 400 examples

Number of	600				400			
elements	1:3	1:5	1:7	1:9	1:3	1:5	1:7	1:9
2	0.92	0.92	0.83	0.80	0.94	0.85	0.82	0.80
3	0.90	0.85	0.80	0.78	0.84	0.78	0.72	0.70
4	0.85	0.80	0.78	0.74	0.82	0.75	0.68	0.60
5	0.75	0.35	0.24	0.06	0.14	0.10	0	0
6	0.22	0.10	0	0	0.06	0	0	0

Table 5 Influence of overlapping on the sensitivity of the tree classifier learned from subclass data. Overlapping is expressed by % of borderline examples in the minority class. Total number of examples – 800.

Number of	1:5			1:9		
subclusters	0%	10%	20%	0%	10%	20%
3	0.96	0.91	0.85	0.94	0.9	0.75
4	0.96	0.89	0.78	0.94	0.87	0.74
5	0.96	0.87	0.76	0.90	0.81	0.66
6	0.94	0.84	0.74	0.88	0.68	0.38

The degradation of performance was larger if the decision boundary became non-linear even for larger data set. Table 3 illustrates results for all classifiers applied to data sets characterized by different imbalanced ratio and smaller or higher decomposition. As the number of examples is rather highest (1200 examples), for nearly balanced data we did not notice the decrease. Non-linear and more complicated shapes (e.g. 02a, or increasing a number of parts in *clover5*) made the problem more difficult, in particular for tree or rule- classifiers, when data became more imbalanced (ratio 1:5). K-NN classifier is rather more local approach than global classifiers (trees or rules) and it works better with more complicated, non-linear classes. Values of AUC decrease in a more visible way if the number of examples is smaller than 600 ones.

Knowing that non-linear shapes were more difficult, we studied more precisely the impact of decomposition the minority class in presence of smaller number of examples. As it is more visible for the sensitivity measure we show a representative results for tree classifier, see Table 4. One can notice that stepwise increasing the number of sub-regions (from 2 to 6) in *clover* shape degrades much more the sensitivity measure than stepwise increasing the class imbalance ratio (from 1:3 to 1:9). Rule and K-NN classifiers showed similar behaviour - although in case of K-NN the degradation was not so radical.

Table 6 Influence of overlapping and rare examples of the minority class on the sensitivity of tree classifier: Subclass data set.

Number of	800				600				400			
subclusters	0%	10%	20%	30%	0%	10%	20%	30%	0%	10%	20%	30%
3	0.96	0.84	0.70	0.56	0.94	0.85	0.70	0.55	0.9	0.82	0.7	0.42
4	0.94	0.84	0.68	0.4	0.92	0.82	0.58	0.3	0.89	0.7	0.4	0.34
5	0.9	0.82	0.56	0.36	0.9	0.78	0.52	0.32	0.87	0.68	0.24	0.18
6	0.88	0.64	0.40	0.34	0.85	0.6	0.36	0.3	0.5	0.22	0.14	0.08

Table 7 Influence of rare and borderline examples: 02a data set.

Classifier	Sensitivity				AUC			
	0%	10%	20%	30%	0%	10%	20%	30%
Tree	0.45	0.38	0.17	0.04	0.82	0.8	0.64	0.5
Rules	0.82	0.70	0.65	0.58	0.92	0.85	0.82	0.78
KNN	0.84	0.72	0.70	0.62	0.95	0.92	0.9	0.87

The next phase of experiments concerned the influence of overlapping in the boundary between classes and the presence of rare minority examples located inside the majority class area. Starting from the overlapping factor, we established the width of the overlapping inside the area of the minority class and parametrized it by percentage of examples from the minority class which were located in this overlapping boundary. The majority examples are uniformly generated inside this boundary with their number was equal to the number of the minority class located there.

Table 5 shows influence of such overlapping on the tree classifier. Although comparing number of sub-regions to amount of overlapping may be not justified we can "roughly" say that stepwise increasing of overlapping could decrease more the sensitivity than stepwise increasing decomposition. For instance, let us analyse the first column (%) - the sensitivity changes from 0.96 to 0.94. While for any of the number of subclusters the sensitivity decreases in range of nearly 0.2 (see, e.g. 4 subclusters, the sensitivity decreases from 0.96 to 0.78). The similar tendency can be observed for rule and K-NN classifiers, also for smaller data sets and non-linear shapes (however, decreases of the sensitivity are even higher).

The next factor was the presence of rare examples from the minority class. We studied their impact together with overlapping of classes in these data sets. More precisely, if the parameter is set to x% it means that half of these examples are generated inside the overlapping and the rest are generated as rare examples. For instance, value 20% means that it is previous case of 10% overlapping extended by 10% minority examples treated as rare one. The appropriate new experiment referring to previous Table 5 is now presented in the next Table 6.

Table 8 Influence of rare and borderline examples: clover4 data set.

Classifier	Sensitivity				AUC			
	0%	10%	20%	30%	0%	10%	20%	30%
Tree	0.5	0.25	0.10	0.08	0.72	0.62	0.55	0.52
Rules	0.68	0.44	0.38	0.35	0.8	0.72	0.68	0.62
KNN	0.90	0.88	0.72	0.62	0.95	0.92	0.82	0.78

Although it could be questionable to compare directly the step of changing rare level with the step of increasing the class decomposition, we can say that stepwise increasing the level of rare and borderline examples decreases much more the evaluation measures than dividing a class into smaller parts (e.g. for 600 examples and no rare moving from 3 to 6 subclusters decreased the sensitivity around 0.1, while adding up to 30% noise or borderline examples introduced a change of the sensitivity over 0.5). Moreover, as it could be expected, adding rare made the problem more difficult (compare results from Table 5 to Table 6).

The similar results were obtained for other shapes and classifiers, see summaries presented in Tables 7 and 8.

Finally we checked all these parameters in case of the other versions of non-linear shapers (as illustrated in Fig. 6 and Fig. 5) where the examples from the majority class are surrounding more closely the minority class. These versions were more difficult to learn and values of evaluation measures were smaller than in the above presented tables.

To sum up, results of the experiments shows that besides decomposition of the minority class, the next important critical factors are:

- overlapping between classes (expressed by the number of borderline examples)
- rare examples from the minority class (in particular if they occur together with similar number of borderline examples).

We can also hypothesize that these factors could cause higher degradation of classification performance than decomposition itself - this is a new result comparing to previous related works. Moreover, presence of all these factors together in the data set causes larger classification deterioration than too low imbalance ratio – in particular for non-linear decision boundaries.

11.6 Improving Classifiers by Focused Re-sampling Methods

In the previous section we experimentally showed critical factors for degrading the performance of the selected tree, rule and K-NN classifiers. Pre-procesing methods that change distribution of examples in classes are one of the main types of specialize methods for improving classifiers in case of class imbalance [11, 18]. These methods, sometimes also called *re-sampling techniques*, are classifier-independent and consist in transforming an original data distribution to change the balance between classes. Some them can also handle other properties of data distributions [3, 18].

Therefore, the next research problem of the following paper is to check the sensitivity of different re-sampling methods to overlapping, rare examples and class decomposition factors in the considered artificial data sets. We have chosen 4 methods some related to the proposal of the SPIDER method, introduced by Stefanowski and Wilk [54, 55]. More precisely they are *simple random over-sampling, cluster based over-sampling* and *nearest cleaning rule* NCR and SPIDER.

In the following review we briefly describe only these methods and SMOTE, which is also related to our proposal of SPIDER; for more extensive reviews see, e.g., [3, 11, 18, 62].

First of all, the simplest re-sampling techniques are random *over-sampling* which replicates examples from the minority class and random *under-sampling* which randomly eliminates examples from the majority classes until a required degree of balance between classes is reached (Many researchers attempted to obtain the same cardinality of the minority class as the majority one). However, several authors showed the random under-sampling or over-sampling were not sufficiently good at improving recognition of imbalanced classes. Random under-sampling may potentially remove some important examples and simple over-sampling may also lead to overfitting [9, 35]. Furthermore it is not easy to find an optimal ratio for balancing classes. Several authors have already shown that used ”even” distribution (i.e. obtaining the same cardinality in classes) is not optimal when dealing with such rare classes. For instance, the reader can consult the comprehensive study with many data sets and classifiers showing that depending on combination of data and classifiers the ratios of modified majority vs. minority class cardinalities like 3:1 and 2:1 quite often outperformed the most popular ratio 1:1 [33]

Therefore, researchers proposed more elaborated methods that attempt at taking into account data characteristics and factors influencing nature of class imbalance.

Following critical observations on the role of small disjuncts Japkowicz proposed an advanced oversampling method (*cluster oversampling*) that takes into account not only *between-class imbalance* but also *within-class imbalance*, where classes are additionally decomposed into smaller sub-clusters [30]. First, random oversampling is applied to individual clusters of the majority classes so that all the sub-clusters are of the same size. Then, minority class clusters are processed in the same way until class distribution becomes balanced. This approach was successfully verified in experiments with decomposed classes [30, 43].

11.6.1 Informed Undersampling

Kubat and Matwin in their paper on *one-side selection* claim that characteristics of mutual positions of learning examples is a source of difficulty for learning classifiers from imbalanced data [35]; see also their more application study [36]. They focus attention on *noisy* majority class examples located inside the minority class and *borderline* examples. According to their approach, such examples are removed from the majority classes, while the minority class is kept unchanged (these examples can be identified with so called Tomek links [58]). As result of such "focused" under-sampling ambiguous regions around the minority class are "cleaned". Moreover, some examples from "safer" regions of the minority class can be also discarded as they could be correctly classified by other learning examples.

Then, the *Nearest Cleaning Rule* (NCR) method was introduced in [37]. This method is based on the Edited Nearest Neighbor Rule (ENNR) and it removes these examples from the majority classes that are misclassified by its k nearest neighbors. Experimental results confirmed that both methods improved the sensitivity of the minority class comparing to simpler over- or under-sampling methods [35, 37].

11.6.2 Informed Oversampling Methods

A most well-known representative of informed over-sampling is SMOTE (Synthetic Minority Over-sampling Technique) introduced by Chawla et al. [9], which considers each example from the minority class and generates new synthetic examples along the lines between this example and some of its randomly selected k nearest neighbours (also belonging to the minority class). Experiments reported in [9] with C4.5 trees, Ripper rules and Naive Bayes classifiers showed that SMOTE improved recognition of the minority class. Moreover, its combination with under-sampling of the majority class was able to achieve better results than other under-sampling methods as ENNR alone – see, e.g., [3]. There are also other proposals of hybridization of the basic SMOTE with additional ”filtering” step, see e.g. the use of some rough sets inspired solutions [48] or more sophisticated ensemble noise filtering [49].

Although SMOTE and NCR showed to be promising in experimental evaluation, they also demonstrated several shortcomings that became motivations for introducing SPIDER – we will further discuss them in the next section. We should note that recently some researchers have also tried to propose various generalizations of SMOTE following similar critical observations – see discussion in [18]. Two most interesting generalizations of SMOTE are Borderline SMOTE that takes into account the different nature of examples from the minority class, and Safe-Level SMOTE, where also the distribution of the majority class is considered while generating synthetic examples from the minority class. Both methods are described in [18, 40]. Yet another proposal is based on controlling distributions in local neighborhoods of the seed example and its nearest neighbors from both minority and majority classes [40].

11.6.3 SPIDER Method

The critical analysis of undesirable properties of well-known focused re-sampling methods, especially NCR and SMOTE, became a starting point for developing by Stefanowski and Wilk the SPIDER method [54]. NCR and in particular one-side-selection are too strongly biased toward cleaning overlapping regions between classes and interior areas of the majority class. However, both methods may *remove* *too many examples* from the majority classes. Such greedy “cleaning” definitely leads to the increased sensitivity for the minority class but too extensive changes in the majority classes may deteriorate the ability of an induced classifier to recognize examples from these classes.

One of the main shortcoming of SMOTE is the *overgeneralization* problem. SMOTE blindly generalizes regions of the minority class without checking positions of the nearest examples from the majority classes. This strategy is particularly problematic in the case of skewed class distribution where the minority class is very sparse in comparison to the majority classes. In such a situation SMOTE may increase overlapping between classes by locating synthetic examples from the minority class among existing examples from the majority classes. Moreover, the number of synthetic examples generated by SMOTE has to be *globally parametrized*, thus reducing the flexibility of the approach. Results of experimental studies with simulated data sets [24] imply that an efficient method should be rather focused on local distributions of difficult examples than being controlled by a global parameter. Let us also notice that according to experimental results reported in the literature an appropriate value of this parameter strongly influences SMOTE’s performance and its proper tuning requires a computationally costly procedure (iterative testing of various possible values). Finally, random introduction of synthetic examples by SMOTE may be difficult to justify in some domains where it is important to preserve a link between original data and a constructed classifier in order to explain suggested decisions.

The SPIDER method relies on the local characteristics of examples (i.e., characteristics of their local neighborhood) and distinguishing between different types of examples. Two types of examples are distinguished – *safe* and *not-safe*. Safe examples should be correctly classified by a constructed classifier, while not-safe ones are likely to be misclassified and require special processing. In SPIDER the type of an example is discovered by applying the *nearest neighbor rule* (NNR) with the heterogeneous value distance metric (HVDM) [64] – i.e., the distance is calculated with the Euclidean distance metric for numerical features and with the value distance metric [14] for qualitative features. According to NNR an example is *safe* if it is correctly classified by its k nearest neighbors, otherwise it is *not-safe*.

More precise categorization of an example is based on the analysis of its neighbors from the other classes (i.e., different than the class of a considered example). If an example is not-safe and its nearest examples belong to other classes, then this example is identified as *certain-not-safe* and we interpret it as a rare case (or an outlier) located deeply inside the other classes. Such example should be treated in a different way than a not-safe example with some neighbours from the same class – this one is rather located in / or closer to an overlapping region between classes. Unlike related methods that distinguish the type of examples in the minority class only, SPIDER identifies the nature of examples in all classes. SPIDER assumes two decision classes – the minority class c_{min} and the majority class c_{maj} – if an original data set contains several majority classes they are collapsed together.

The method consists of two main phases – *identification* and *pre-processing*. In the first phase the type of examples is identified according to the ”local” characteristics of their neighbors and 'flagged' accordingly. Firstly, examples from the majority class c_{maj} are processed. In particular, depending on the *relabel* option the method either removes or relabels certain-not-safe / noisy examples[3] from c_{maj} (i.e., changes their classification to c_{min}). Then, in the second phase, it identifies the characteristic of examples from c_{min} considering changes introduced in the first phase. Not-safe examples from this class require special processing – i.e they are amplified (by replicating them with different degree) according to the *ampl* option.

All options of SPIDER involve modification of the minority and majority classes, however, the degree and scope of changes varies between options. *Weak* amplification is the simplest and less greedy modification of the minority class. It focuses on not-safe examples from c_{min} and slightly over-samples them by adding as many of their copies as there are safe examples from c_{maj} in their neighborhoods. The second option – *relabel* – also changes these certain noisy examples from c_{maj} which could be interpreted as noisy outliers located more deeply inside the minority class. For the last option – *strong* – the degree of amplification of not-safe examples c_{min} could be higher depending on analysis of an extended neighborhood. Much more thorough description of the method is provided in [54, 55], including precise pseudocode of the algorithm.

11.7 Experiments with Focused Re-sampling Methods

In the next experiments we will study the impact of overlapping, rare examples and partly class decomposition on the performance of selected pre-processing methods for handling class imbalance, including our proposal of SPIDER. Here, we summarize some of the main results of the more comprehensive study on this topic recently carried out by Napierala, Stefanowski and Wilk [42].

According to the results of Stefanowski's earlier experiments (presented in Section 11.5) a group of data sets with 800 examples, the imbalance ratio of 1:7, and 5 sub-regions for the *subclus* and *clover* shapes is selected for experiments. We chose their more difficult versions where the majority class is uniformly distributed around the minority class shapes – see the Figures 5 and 6. Let us remind that all these data sets presented a significant challenge for a stand-alone classifier. Similar behaviour was observed for data sets with 600 examples, but due to space limit we did not describe these data sets in the paper. Due to specific aims of study [42], two symbolic rule and tree classifiers were applied. Trees were induced by C4.5 algorithm, while rules were induced by MODLEM algorithm (introduced by Stefanowski in [50]; see also for its description in [51, 52].

Firstly, the impact of disturbing the borders of sub-regions in the minority class was evaluated. It was simulated by increasing the ratio of borderline examples from the minority class subregions. This ratio (further called the *disturbance ratio*) was

[3] In case of the majority class we can consider possible noise example if a not-safe example is located deeply inside the region of the minority class (all its k-nearest neighbors belong to the opposite class) and it could be wrongly re-classified by its neighbors

changed from 0 to 70%. The width of the borderline overlapping areas was comparable to the width of the sub-regions (sub-parts in data shapes).

The constructed classifiers were combined with the following focused pre-processing methods:

- standard random oversampling (abbreviated as RO),
- Japkowicz's cluster oversampling (CO),
- nearest cleaning rule NCR,
- and SPIDER (SPID).

Cluster oversampling was limited to the minority class, and SPIDER method was used with the strong amplification as such combination performed best in our earlier studies [54, 55]. For baseline results (Base), both classifiers were ran without any pre-processing.

Table 9 G-mean for artificial data sets with varying degree of the disturbance ratio in the overlapping region.

Data set	Rules					Trees				
	Base	RO	CO	NCR	SPID	Base	RO	CO	NCR	SPID
subclus-0	0.9373	0.9376	0.9481	0.9252	0.9294	0.9738	0.9715	0.9715	0.9613	0.9716
subclus-30	0.7327	0.7241	0.7242	0.7016	0.7152	0.6524	0.7933	0.7847	0.7845	0.8144
subclus-50	0.5598	0.5648	0.6020	0.6664	0.6204	0.3518	0.7198	0.7113	0.7534	0.7747
subclus-70	0.4076	0.4424	0.4691	0.5957	0.5784	0.0000	0.7083	0.7374	0.6720	0.7838
clover-0	0.7392	0.7416	0.7607	0.7780	0.7908	0.6381	0.8697	0.8872	0.6367	0.6750
clover-30	0.6361	0.6366	0.6512	0.7221	0.6765	0.2566	0.7875	0.7652	0.6758	0.7686
clover-50	0.5066	0.5540	0.5491	0.6956	0.6013	0.1102	0.7453	0.7570	0.6184	0.7772
clover-70	0.4178	0.4658	0.4898	0.6583	0.5668	0.0211	0.7140	0.7027	0.6244	0.7665
paw-0	0.9041	0.9126	0.9182	0.9184	0.8918	0.6744	0.9318	0.9326	0.6599	0.7330
paw-30	0.7634	0.7762	0.7701	0.7852	0.7780	0.3286	0.8374	0.8334	0.8527	0.8337
paw-50	0.6587	0.6863	0.6865	0.7517	0.7120	0.3162	0.8013	0.7858	0.8200	0.8075
paw-70	0.5084	0.5818	0.5691	0.7182	0.6506	0.0152	0.7618	0.7472	0.7824	0.8204

We do not report all results from [42] but summarize the most representative ones. However, let us remark that with respect to recognition of the minority class alone, expressed by the sensitivity measure, results clearly showed that all methods of pre-processing improved the sensitivity of both classifiers in comparison to Base classifiers (in particular for more difficult decision boundaries and larger disturbance). Generally speaking, simpler over-sampling RO and CO performed comparably on all data sets, and on non-disturbed data sets they often over-performed focused methods NCR and SPIDER. On more difficult sets (disturbance = 50–70%) both methods NCR and SPIDER were significantly better than oversampling methods. Then, we took into account balance between sensitivity and specificity, so recognizing examples also from the other majority class. Results of the geometric mean (G-mean) are presented in Table 9.

These experiments also showed that the degradation in performance of a classifier is strongly affected by the number of borderline examples. If the overlapping area is large enough (in comparison to the area of the minority sub-clusters), and at least

30% of examples from the minority class are located in this area, then focused re-sampling methods (NCR, SPIDER) strongly outperform random and cluster over-sampling with respect to sensitivity and G-mean. Moreover, the performance gain increases with the number of borderline examples. On the contrary, if the number of borderline examples is small, then oversampling methods sufficiently improve the recognition of the minority class and they are comparable to focused method with respect to G-mean.

Table 10 Sensitivity for artificial data sets with different types of testing examples

Data set	Rules					Trees				
	Base	RO	CO	NCR	SPID	Base	RO	CO	NCR	SPID
subcl-safe	0.58	0.58	0.62	0.78	0.64	0.32	0.84	0.86	0.98	1.00
subcl-B	0.84	0.84	0.84	0.86	0.84	0.00	0.82	0.84	0.36	0.92
subcl-C	0.12	0.1 0	0.16	0.24	0.26	0. 00	0.54	0.00	0.00	0.52
subcl-BC	0.48	0.47	0.50	0.55	0.55	0.00	0.68	0.42	0.18	0.72
clover-safe	0.30	0.38	0.44	0.70	0.60	0.02	0.96	0.92	0.04	0.98
clover-B	0.84	0.82	0.82	0.84	0.86	0.04	0.94	0.92	0.04	0.94
clover-C	0.14	0.08	0.14	0.24	0.36	0.00	0.30	0.02	0.00	0.40
clover-BC	0.49	0.45	0.48	0.54	0.61	0.02	0.62	0.47	0.02	0.67
paw-safe	0.84	0.92	0.84	0.84	0.80	0.42	0.90	0.96	0.74	1.00
paw-B	0.88	0.88	0.86	0.88	0.90	0.14	0.90	0.90	0.40	0.92
paw-C	0.16	0.14	0.12	0.26	0.16	0.04	0.20	0.00	0.00	0.34
paw-BC	0.52	0.51	0.49	0.57	0.53	0.09	0.55	0.45	0.00	0.63

In [42] we carried out additional experiments where we studied the impact of rare examples from the minority class, located outside the borderline area, on the performance of a classifier. To achieve this, we introduced new rare examples (single and pairs) and denoted them with C. Similarly to the first series of experiments we used data sets of three shapes (*subclus*, *clover* and *paw*), 800 examples and the imbalance ratio of 1:7. We also employed rule- and tree-based classifiers combined with the same pre-processing methods. However, we changed the 10-fold cross validation to the train-test verification in order to ensure that learning and testing sets had similar distributions of the C examples. In each training set 30% of the minority class examples were safe examples located inside sub-regions, 50% were located in the borderline/overlapping area (we denote them with B), and the remaining 20% constituted the C rare examples.

For each training set we prepared 4 testing sets containing the following types of examples from the minority class: only safe examples, only B examples, only C examples, and B and C examples combined together (BC). Results are presented in Table 10. They clearly show that for the "difficult" rarity (C or BC) SPIDER and in most cases NCR were superior to RO, CO and Base. SPIDER was also comparable to RO and CO in case of safe and (sometimes) B examples.

To sum up these experiments reveal the superiority of SPIDER and in most cases NCR in handling rare examples located inside the majority class (also accompanied with borderline ones). Such result has been in a way expected, as both methods were introduced to handle such situations. The experiments also demonstrated that even random oversampling is comparable to SPIDER and better than NCR in classifying safe examples from the minority class.

Besides the above mentioned experiments with artificial data, the focused re-sampling methods, including SPIDER, were also compared on some real data sets coming from UCI Machine Learning Repository [2]. As this kind of experiments is not within the main aim of this paper, and its page size is limited, we do not show these precise results but attempt at summarizing the general conclusions from two earlier papers ([54] - early results with rule based classifiers, and more extended comparison [55] including additional methods and classifiers). In these experiments SPIDER applied together with C4.5 and MODLEM algorithms was compared to competitive methods (NCR and SMOTE) and basic classifiers used without any pre-processing. The results of experiments showed that although NCR often led to the highest increase of sensitivity, at the same time it significantly deteriorated specificity and overall accuracy. SPIDER was the second best with respect to improving the sensitivity of the minority class (improvement was more visible for rule than trees), and slightly better than SMOTE or comparable to it. Moreover, it did not deteriorate the recognition of the majority classes as much as NCR. In case of SPIDER and possible pre-processing options, *weak* resulted in the best specificity and overall accuracy (often at the cost of sensitivity), *strong* resulted in a good balance between specificity and sensitivity – evaluated by G-mean – and *relabel* improved sensitivity in the similar range as *strong*, however at the cost of specificity.

Recently we showed another way of balancing recognition of the minority and majority classes (expressed by optimizing G-mean) which include using SPIDER inside the generalized framework of the adaptive ensembles called IIvotes [6].

Considering results of the experiments with UCI imbalanced data, we could also refer to the analysis of the ”nature” of these data sets taking into account local neighbourhood characteristic for the focused re-sampling methods. In [40] we used $k-NN$ analysis of each minority class example and considered is as certain unsafe (outlier) or borderline (unsafe-possible) as defined in SPIDER. Results (for $k = 3$ or partly 5) showed that all studied data sets are rather difficult with respect to classifier ability for recognizing the minority class. First of all, we noticed that for some data sets the number of outlier examples is quite high comparing to the size of the minority class. Other data sets contained also many borderline examples without too many safe regions of the minority class. Referring to the earlier comparison of oversampling methods we noticed that for such data sets SPIDER and NCR led to improvements of the sensitivity. On the other hand, for a two data sets (as e.g. new thyroid) with more safer examples, base classifiers without preprocessing or simpler oversampling worked sufficiently good. Let us remind that such a data as new thyroid is more imbalanced than other data sets. In our opinion this very simple analysis confirm our earlier observations on the role of critical factors from experiments with controlled artificial data sets.

11.8 Final Remarks

The problem of learning classifiers from imbalanced data has been considered. It is one of the most challenging topics in machine learning and data mining [65]. Moreover, it is still ”open” from a theoretical point of view and it is very important in many application domains. On the other hand, one can notice by studying related literature that it has received growing research interest in the last decade and several specialized methods have already been proposed. Although some of them have been validated in experiments, it is still a need to ask more general research questions about the class imbalance, data characteristics and competence of some popular methods. This paper is an attempt to partly answer to these questions.

Firstly, the nature of this problem and sources of difficulties for achieving good recognition of the minority class are discussed. As it has been already noticed by other researchers, the small number of examples in the minority class is not the main source of difficulty [24, 28, 29, 47]. The degradation of classification performance is rather related to other critical factors as decomposition of the classes into smaller sub-parts including too few examples (so called small disjuncts [30]), overlapping between classes (existence of too many borderline examples in the minority class), presence of rare or noisy examples located farther from the decision boundary (deeper inside the distribution of the opposite class).

Following this literature review, in the first part of this paper we decided to carry out an experimental study on the impact of the critical factors on re-sampling methods dealing with imbalanced data. Unlike the related works we decided to consider them occurring together in the data. Moreover, we pay more attention to presence of borderline and rare examples. We also considered more complicated shapes of classes than in earlier works. This is why we introduced new types of artificial data sets for experimental evaluation. Generated artificial data include simpler rectangle shapes of the minority class (*subclass* following inspirations from earlier works) and more complicated non-linear boundaries as *clover*, *paw* and similar *02* shapes.

In the first phase of experiments we studied impact of critical factors in these artificial data sets on performance of the most popular rule-, tree- and K-NN classifiers. First of all, some results confirmed earlier results obtained for simpler artificial data on the importance of the minority class decomposition into smaller sub-concepts [30]. We also showed that for more nonlinear decision boundaries increasing decomposition of the class into small sub-parts decreased sensitivity or AUC measures. It was also observed that K-NN could classify the non-linear shapes better than trees - it could be explained by its local performance comparing to a more global way of constructing trees (see also more extensive discussion of specific properties of K-NN in [46]). On the other hand, both tree and rule classifiers worked better for rectangle shapes of the minority class. Moreover, rule classifier Ripper was slightly better than C4.5 tree. Its behaviour could be caused by a specific way of constructing a decision list in the final classifier, i.e. the algorithm induced rules only for the minority class (with a controlled pruning level) and they are ordered in a kind of an exception list [13]. If the new / testing example does not match any of these rules it is classified by a default rule to the majority class. As we ran Ripper without strong pruning, it could be better suited to class imbalance in non-linear, complicated, decision boundaries (clover, paw, 02) than more global tree classifiers.

Then, experimental results clearly showed that the combination of class decomposition with overlapping makes learning very difficult (in particular for sensitivity measure and tree classifiers). Focussing attention on the rare examples from the minority class is an original contribution of this study – as according to the best knowledge they have not been studied yet in experimental studies. It was clearly visible that the presence of rare example significantly degraded performance of all classifiers. We could also say that stepwise increasing numbers of borderline and rare examples in the minority class decreased all evaluation measures more that increasing decomposition of this class into new sub-parts. This is also a new observation comparing to previous related research.

To sum up this part of experiments, we hope that our results expand the body of knowledge on the critical role of borderline and rare examples with respect to earlier results based on simpler artificial data sets and other factors [24, 28, 29, 47].

The next part of this paper concerns problems of handling these difficulties by the following re-sampling methods: random over-sampling, cluster over-sampling, informed under-sampling by NCR and SPIDER. Our experiments showed that the degradation in classification performance was strongly affected by the number of borderline examples. If the overlapping area was large enough (in comparison to the area of the minority sub-clusters), and at least 30% of examples from the minority class were located in this area (i.e., they are borderline examples), then focused re-sampling methods (as SPIDER and partly NCR) strongly outperformed random and cluster oversampling with respect to sensitivity and G-mean measures. Moreover, it seams that the performance gain increased with the number of borderline examples. The other experiments revealed the superiority of SPIDER and in some cases NCR in handling rare examples located inside the majority class (also accompanied with borderline ones).

We hope that the above mentioned comparative studies with artificial simulated data also extended by experiments on UCI data sets could give more insight into conditions of the usefulness of particular re-sampling methods to improve classifiers learned from imbalanced data.

Acknowledgements. The research has been partly supported by the Polish Ministry of Science and Higher Education, grant no. N N519 441939.

References

1. Anyfantis, D., Karagiannopoulos, M., Kotsiantis, S., Pintelas, P.: Robustness of learning techniques in handling class noise in imbalanced datasets. In: Proc. of the IFIP International Federation for Information Processing Comf. AIAI 2007, pp. 21–28 (2007)
2. Asuncion, A., Newman, D.J.: UCI Machine Learning Repository. School of Information and Computer Science. University of California, Irvine, CA (2007), http://www.ics.uci.edu/~mlearn/MLRepository.html

3. Batista, G., Prati, R., Monard, M.: A study of the behavior of several methods for balancing machine learning training data. ACM SIGKDD Explorations Newsletter 6(1), 20–29 (2004)
4. Batista, G.E.A.P.A., Prati, R.C., Monard, M.C.: Balancing Strategies and Class Overlapping. In: Famili, A.F., Kok, J.N., Peña, J.M., Siebes, A., Feelders, A. (eds.) IDA 2005. LNCS, vol. 3646, pp. 24–35. Springer, Heidelberg (2005)
5. Bay, S., Kumaraswamy, K., Anderle, M.G., Kumar, R., Steier, D.M.: Large scale detection of irregularities in accounting data. In: Proc. of the ICDM Conf., pp. 75–86 (2006)
6. Błaszczyński, J., Deckert, M., Stefanowski, J., Wilk, S.: Integrating Selective Preprocessing of Imbalanced Data with Ivotes Ensemble. In: Szczuka, M., Kryszkiewicz, M., Ramanna, S., Jensen, R., Hu, Q. (eds.) RSCTC 2010. LNCS, vol. 6086, pp. 148–157. Springer, Heidelberg (2010)
7. Brodley, C.E., Friedl, M.A.: Identifying Mislabeled Training Data. Journal of Artificial Intelligence Research 11, 131–167 (1999)
8. Casagrande, N.: The class imbalance problem: A systematic study. Research Report IFT 6390. Montreal University
9. Chawla, N., Bowyer, K., Hall, L., Kegelmeyer, W.: SMOTE: Synthetic Minority Oversampling Technique. J. of Artifical Intelligence Research 16, 341–378 (2002)
10. Chawla, N.V., Lazarevic, A., Hall, L.O., Bowyer, K.W.: SMOTEBoost: Improving Prediction of the Minority Class in Boosting. In: Lavrač, N., Gamberger, D., Todorovski, L., Blockeel, H. (eds.) PKDD 2003. LNCS (LNAI), vol. 2838, pp. 107–119. Springer, Heidelberg (2003)
11. Chawla, N.: Data mining for imbalanced datasets: An overview. In: Maimon, O., Rokach, L. (eds.) The Data Mining and Knowledge Discovery Handbook, pp. 853–867. Springer (2005)
12. Chawla, N., Japkowicz, N., Kolcz, A.: Editorial: Special Issue on Learning from Imbalanced Data Sets. ACM SIGKDD Explorations Newsletter 6(1), 1–6 (2004)
13. Cohen, W.: Fast effective rule induction. In: Proc. of the 12th Int. ICML Conf., pp. 115–123 (1995)
14. Cost, S., Salzberg, S.: A Weighted Nearest Neighbor Algorithm for Learning with Symbolic Features. Machine Learning Journal 10(1), 1213–1228 (1993)
15. Davis, J., Goadrich, M.: The Relationship between Precision- Recall and ROC Curves. In: Proc. Int. Conf. on Machine Learning, ICML 2006, pp. 233–240 (2006)
16. Fawcett, T.: ROC Graphs: Notes and Practical Considerations for Data Mining Researchers. Technical Report HPL-2003-4. HP Labs (2003)
17. Fawcett, T., Provost, F.: Adaptive Fraud Detection. Data Mining and Knowledge Discovery 1(3), 29–316 (1997)
18. He, H., Garcia, E.: Learning from imbalanced data. IEEE Transactions on Data and Knowledge Engineering 21(9), 1263–1284 (2009)
19. He, J.: Rare Category Analysis. Ph.D Thesis. Machine Learning Department. Carnegie Mellon University Pittsburgh (May 2010), CMU-ML-10-106 Report
20. Holte, C., Acker, L.E., Porter, B.W.: Concept Learning and the Problem of Small Disjuncts. In: Proc. of the 11th JCAI Conference, pp. 813–818 (1989)
21. Galar, M., Fernandez, A., Barrenechea, E., Bustince, H., Herrera, F.: A Review on Ensembles for the Class Imbalance Problem: Bagging-, Boosting-, and Hybrid-Based Approaches. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 99, 1–22 (2011)
22. Gamberger, D., Boskovic, R., Lavrac, N., Groselj, C.: Experiments With Noise Filtering in a Medical Domain. In: Proceedings of the Sixteenth International Conference on Machine Learning, pp. 143–151 (1999)

23. Garcia, S., Fernandez, A., Herrera, F.: Enhancing the effectiveness and interpretability of decision tree and rule induction classifiers with evolutionary training set selection over imbalanced problems. Applied Soft Computing 9, 1304–1314 (2009)
24. García, V., Sánchez, J., Mollineda, R.A.: An Empirical Study of the Behavior of Classifiers on Imbalanced and Overlapped Data Sets. In: Rueda, L., Mery, D., Kittler, J. (eds.) CIARP 2007. LNCS, vol. 4756, pp. 397–406. Springer, Heidelberg (2007)
25. Garcia, V., Mollineda, R.A., Sanchez, J.S.: On the k-NN performance in a challenging scenario of imbalance and overlapping. Pattern Analysis and Applications 11, 269–280 (2008)
26. Grzymala-Busse, J.W., Goodwin, L.K., Zheng, X.: An approach to imbalanced data sets based on changing rule strength. In: AAAI Workshop at the 17th Conference on AI Learning from Imbalanced Data Sets, Austin, TX, pp. 69–74 (2000)
27. Grzymala-Busse, J.W., Stefanowski, J., Wilk, S.: A comparison of two approaches to data mining from imbalanced data. Journal of Intelligent Manufacturing 16(6), 565–574 (2005)
28. Japkowicz, N., Stephen, S.: Class imbalance problem: a systematic study. Intelligent Data Analysis Journal 6(5), 429–450 (2002)
29. Japkowicz, N.: Class imbalance: Are we focusing on the right issue? In: Proc. II Workshop on Learning from Imbalanced Data Sets, ICML Conference, pp. 17–23 (2003)
30. Jo, T., Japkowicz, N.: Class Imbalances versus small disjuncts. ACM SIGKDD Explorations Newsletter 6(1), 40–49 (2004)
31. Joshi, M.V., Agarwal, R.C., Kumar, V.: Mining needles in a haystack: classifying rare classes via two-phase rule induction. In: Proc. of SIGMOD KDD 2001 Conference on Management of Data (2001)
32. Kaluzny, K.: Analysis of class decomposition in imbalanced data. Master Thesis (supervised by J.Stefanowski). Faculty of Computer Science and Managment, Poznan University of Technology (2009)
33. Khoshgoftaar, T., Seiffert, C., Van Hulse, J., Napolitano, A., Folleco, A.: Learning with Limited Minority Class Data. In: Proc. of the 6th Int. Conference on Machine Learning and Applications, pp. 348–353 (2007)
34. Kononenko, I., Kukar, M.: Machine Learning and Data Mining. Horwood Pub. (2007)
35. Kubat, M., Matwin, S.: Addresing the curse of imbalanced training sets: one-side selection. In: Proc. of the 14th Int. Conf. on Machine Learning ICML 1997, pp. 179–186 (1997)
36. Kubat, M., Holte, R., Matwin, S.: Machine Learning for the Detection of Oil Spills in Radar Images. Machine Learning Journal 30, 195–215 (1998)
37. Laurikkala, J.: Improving identification of difficult small classes by balancing class distribution. Tech. Report A-2001-2, University of Tampere (2001); Another version was published in: Laurikkala, J.: Improving Identification of Difficult Small Classes by Balancing Class Distribution. In: Quaglini, S., Barahona, P., Andreassen, S. (eds.) AIME 2001. LNCS (LNAI), vol. 2101, pp. 63–66. Springer, Heidelberg (2001)
38. Lewis, D., Catlett, J.: Heterogenous uncertainty sampling for supervised learning. In: Proc. of 11th Int. Conf. on Machine Learning, pp. 148–156 (1994)
39. Ling, C., Li, C.: Data Mining for Direct Marketing Problems and Solutions. In: Proceedings of the Fourth International Conference on Knowledge Discovery and Data Mining (KDD 1998), pp. 73–79. AAAI Press, New York (1998)
40. Maciejewski, T., Stefanowski, J.: Local Neighbourhood Extension of SMOTE for Mining Imbalanced Data. In: Proceeding IEEE Symposium on Computational Intelligence in Data Mining, within Joint IEEE Series of Symposiums of Computational Intelligence, April 11-14, pp. 104–111. IEEE Press, Paris (2011)

41. Mitchell, T.: Machine learning. McGraw Hill (1997)
42. Napierała, K., Stefanowski, J., Wilk, S.: Learning from Imbalanced Data in Presence of Noisy and Borderline Examples. In: Szczuka, M., Kryszkiewicz, M., Ramanna, S., Jensen, R., Hu, Q. (eds.) RSCTC 2010. LNCS(LNAI), vol. 6086, pp. 158–167. Springer, Heidelberg (2010)
43. Nickerson, A., Japkowicz, N., Milios, E.: Using unsupervised learning to guide resampling in imbalanced data sets. In: Proc. of the 8th Int. Workshop on Artificial Intelligence and Statistics, pp. 261–265 (2001)
44. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann (1992)
45. Pelleg, D., Moore, A.W.: Active learning for anomaly and rare-category detection. In: Proc. of NIPS (2004)
46. Prati, R.C., Batista, G.E.A.P.A., Monard, M.C.: Learning with Class Skews and Small Disjuncts. In: Bazzan, A.L.C., Labidi, S. (eds.) SBIA 2004. LNCS (LNAI), vol. 3171, pp. 296–306. Springer, Heidelberg (2004)
47. Prati, R., Batista, G., Monard, M.: Class imbalance versus class overlapping: an analysis of a learning system behavior. In: Proc. 3rd Mexican Int. Conf. on Artificial Intelligence, pp. 312–321 (2004)
48. Ramentol, E., Caballero, Y., Bello, R., Herrera, F.: SMOTE-RSB*: A Hybrid Preprocessing Approach based on Oversampling and Undersampling for High Imbalanced Data-Sets using SMOTE and Rough Sets Theory. Knowledge and Information Systems Journal (2011) (accepted)
49. Saez, J.A., Luengo, J., Stefanowski, J., Herrera, F.: Addressing the Noisy and Borderline Examples Problem in Classication with Imbalanced Datasets via a Class Noise Filtering Method-based Re-sampling Technique. Manuscript submitted to Pattern Recognition (2011)
50. Stefanowski, J.: The rough set based rule induction technique for classification problems. In: Proc. of the 6th European Conf. on Intelligent Techniques and Soft Computing EUFIT 1998, pp. 109–113 (1998)
51. Stefanowski, J.: Algorithms of rule induction for knowledge discovery. Habilitation Thesis published as Series Rozprawy no. 361. Poznan University of Technology Press (2001) (in Polish)
52. Stefanowski, J.: On Combined Classifiers, Rule Induction and Rough Sets. In: Peters, J.F., Skowron, A., Düntsch, I., Grzymała-Busse, J.W., Orłowska, E., Polkowski, L. (eds.) Transactions on Rough Sets VI. LNCS, vol. 4374, pp. 329–350. Springer, Heidelberg (2007)
53. Stefanowski, J.: An experimental analysis of impact class decomposition and overlapping on the performance of classifiers learned from imbalanced data. Research Report of Institute of Computing Science, Poznan University of Technology, RB- 010/06 (2010)
54. Stefanowski, J., Wilk, S.: Improving Rule Based Classifiers Induced by MODLEM by Selective Pre-processing of Imbalanced Data. In: Proc. of the RSKD Workshop at ECML/PKDD, Warsaw, pp. 54–65 (2007)
55. Stefanowski, J., Wilk, S.: Selective Pre-processing of Imbalanced Data for Improving Classification Performance. In: Song, I.-Y., Eder, J., Nguyen, T.M. (eds.) DaWaK 2008. LNCS, vol. 5182, pp. 283–292. Springer, Heidelberg (2008)
56. Stefanowski, J., Wilk, S.: Extending Rule-Based Classifiers to Improve Recognition of Imbalanced Classes. In: Ras, Z.W., Dardzinska, A. (eds.) Advances in Data Management. SCI, vol. 223, pp. 131–154. Springer, Heidelberg (2009)
57. Sun, A., Lim, E.P., Liu, Y.: On strategies for imbalanced text classication using SVM: A comparative study. Decision Support Systems 48(1), 191–201 (2009)

58. Tomek, I.: Two Modications of CNN. IEEE Transactions on Systems, Man and Communications 6, 769–772 (1976)
59. Van Hulse, J., Khoshgoftarr, T., Napolitano, A.: Experimental perspectives on learning from imbalanced data. In: Proceedings of ICML 2007, pp. 935–942 (2007)
60. Van Hulse, J., Khoshgoftarr, T.: Knowledge discovery from imbalanced and noisy data. Data and Knowledge Engineering 68, 1513–1542 (2009)
61. Wang, B., Japkowicz, N.: Boosting support vector machines for imbalanced data sets. Knowledge and Information Systems 25(1), 1–20 (2010)
62. Weiss, G.M.: Mining with rarity: a unifying framework. ACM SIGKDD Explorations Newsletter 6(1), 7–19 (2004)
63. Weiss, G.M., Provost, F.: Learning when training data are costly: the efect of class distribution on tree induction. Journal of Artificial Intelligence Research 19, 315–354 (2003)
64. Wilson, D.R., Martinez, T.: Reduction techniques for instance-based learning algorithms. Machine Learning Journal 38, 257–286 (2000)
65. Wu, J., Xiong, H., Wu, P., Chen, J.: Local decomposition for rare class analysis. In: Proc. of KDD 2007 Conf., pp. 814–823 (2007)
66. Yang, Q., Wu, X.: 10 challenging problems in data mining research. International Journal of Information Technology and Decision Making 5(4), 597–604 (2006)

Chapter 12
A Granular Computing Paradigm for Concept Learning

Yiyu Yao and Xiaofei Deng

Abstract. The problem of concept formation and learning is examined from the viewpoint of granular computing. Correspondences are drawn between granules and concepts, between granulations and classifications, and between relations over granules and relations over concepts. Two learning strategies are investigated. A global attribute-oriented strategy searches for a good partition of a universe of objects and a local attribute-value-oriented strategy searches for a good covering. The proposed granular computing paradigm for concept learning offers twofold benefits. Results from concept formulation and learning enrich granular computing and a granular computing viewpoint sheds new light on concept formulation and learning.

12.1 Introduction

Concepts are basic units of human thought and have been considered in different disciplines, including philosophy, cognitive science, inductive learning, cluster analysis and machine learning. Many views of concepts and categories have been proposed and studied, such as the classical view, the prototype view, the exemplar view, the frame view, and the theory view [8, 15, 21]. Each view captures a particular perspective and emphasizes a specific aspect, with different intended applications. A comprehensive understanding of concepts is based on an integration of those different views. While some views are suitable for human concept formation and learning, other views are appropriate for machine-oriented approaches. The classical view of concepts is perhaps one of the most used ones in machine-oriented concept learning [32].

In the classical view, a concept is understood as a pair of intension and extension. The intension of a concept consists of all properties or attributes (more generally,

Yiyu Yao and Xiaofei Deng
Department of Computer Science, University of Regina, Regina,
Saskatchewan, Canada S4S 0A2
e-mail: {yyao,deng200x}@cs.uregina.ca

some formulas of a language) that are valid for all those objects to which the concept applies. The extension of a concept is the set of objects or entities which are instances of the concept. A concept is thus described jointly by its intension and extension. The simple classical view of concepts can be easily related to the notion of granules in an emerging field of study known as granular computing. Specifically, the extension of a concept may be considered as a granule and the intension of the concept as the description of the granule. Consequently, concept learning may be reconsidered in the paradigm of granular computing.

There are two crucial tasks in concept learning. One is to find a good description of a concept based on its extension; the other is to derive relations between concepts based on their corresponding extensions. We typically employ a rule to describe a relationship between two concepts, in which the intensions of concepts are used. To make a concept learning algorithm effective and its results meaningful, we must consider several issues. One issue is the selection of a set of meaningful basic concepts, from which target concepts can be expressed and interpreted. Another issue is to design strategies for learning. Different strategies may lead to different descriptions of the target concepts.

Based on results from the rough set theory [17], a granular computing paradigm of concept learning is introduced and examined in this chapter. Two learning strategies will be studied in an information table, in which a finite set of objects is described by using a finite set of attributes. An attribute-oriented strategy constructs a partition of the universe by using a subset of attributes. The basic granules are equivalence classes of the partition. Concept learning is formulated as a divide and conquer method in search of a good subset of attributes. The results are a decision tree, which can be translated into a family of disjoint rules. An attribute-value-oriented strategy constructs a covering of the universe. The basic granules are subsets of objects defined by a family of attribute-value pairs. Concept learning is formulated as a chip and conquer method in search of a good family of sets of attribute-value pairs. The results are a family of decision trees, from which a set of overlapping rules can be derived.

The rest of this chapter is organized as follows. Section 12.2 presents an overview of a triarchic theory of granular computing. Section 12.3 discusses connections between granular computing and concept learning. Section 12.4 proposes a model of concept learning based on granular computing, involving an investigation of two classes of strategies for concept learning.

12.2 A Triarchic Theory of Granular Computing

The notions of categorization, abstraction, formulation and approximation at multiple levels of granularity play a crucial role in human perception, cognition, understanding and problem solving [12, 24, 28, 31]. One would also expect that they play an equal important role in intelligent systems. For a system to be called intelligent, it must have similar built-in mechanisms that support human intelligence. Granular computing, as an emerging field of study, aims at a systematic investigation of granule based theories and methodologies for supporting human problem solving on one hand and machine problem solving on the other [27, 33].

The triarchic theory of granular computing [25, 26, 27, 29, 30] provides a conceptual model by adopting useful structures called granular structures and weaving together three powerful ideas: structured thinking, structured problem solving, and structured information processing. It emphasizes on the exploitation of useful structures that properly reflect multiple levels of granularity or a granularity pyramid. A single hierarchical granular structure called a hierarchy provides a multilevel understanding and representation of a problem or a system. But it typically captures one particular aspect and therefore offers one view. By constructing a family of hierarchies, it is possible to obtain multiple different views. Granular structures are a family of complementary hierarchies working together for a complete and comprehensive multiview understanding and representation. The use of granular structures, characterized by multilevel and multiview, establishes a solid basis on which sit structured thinking, structured problem solving and structured information processing.

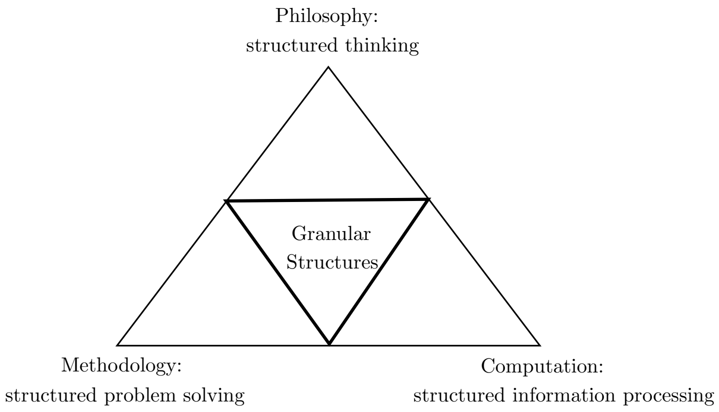


Fig. 1 The Granular Computing Triangle

The main ideas of the triarchic theory of granular computing are summarized by the granular computing triangle of Fig. 1. The center of the triangle is granular structures, the three angles represent three perspectives of granular computing centered around granular structures.

12.2.1 Multilevel, Multiview Granular Structures

A key to the success of granular computing is the use of properly constructed granular structures. In the triarchic theory, a hierarchy is used to achieve one particular multilevel view and a family of hierarchies is used to achieve multiview. Granular structures are a family of hierarchies. Fig. 2 illustrates a multilevel hierarchical granular structure. Its main ingredients are explained as follows.

12.2.1.1 Granules and Granularity

Granules are a primitive notion of granular computing. They are the basic vocabularies of granular computing. Granules are the basic elements in forming granular structures. Using the terminology of systems theory, a granule represents a part of a whole. Granular computing studies the whole through an integration of its parts in terms of their interrelations and their connections to the whole. In Fig. 2, granules are small circle or dots on a plane and each represents a part of the whole given by the entire plane. The physical interpretations and construction of granules are left to particular applications.

The granularity of a granule may be interpreted as an intrinsic or inherent property of the granule. Intuitively, granularity may be interpreted as the degree of abstraction, generalization, complexity or details. A granule with higher degree of granularity is more abstract. It is reasonable to assumed that granularity can be at least partially ordered, so it is possible to form a granularity pyramid. The concept of granularity is essential for constructing and interpreting granules.

12.2.1.2 Granulations and Levels

Each granule provides a local description of a specific part of a problem or a system. By collecting a family of granules of similar nature or similar granularity, we obtain a complete description of a problem or a system. Such a family of granules is called a granulation of the problem at a particular level. Granules in the family are called focal elements of discussion at the level. In Fig. 2, each level is represented by a plane. While granules at the same level are of similar nature, granules at different levels may be very different. Consequently, we may use different vocabularies and languages for descriptions at different levels.

The processing methods at different levels may also be different. Institutively, each level of granulation may be considered to be a particular viewpoint of the problem. Thus, a hierarchy provides a family of multiple levels of viewpoints. The concept of levels is a very versatile and universal notion that appears in a wide range of disciplines [1, 5, 7, 11, 31]. An interpretation of a granulation hierarchy in terms of levels makes a granular structure to be flexible and useful [31]. The properties of a particular level, namely, a granulation, is determined collectively by a family of granules through their connections and interactions.

12.2.1.3 Partial Orderings of Granules and Levels

Granules can be ordered based on their relationship or granularity. This leads to a refinement-coarsening relationship between granules. Under this relationship, one can identify dual-roles of a granule. A granule may be considered to be a part that is to be coarsened with other granules into the whole of a coarser or more abstract granule. It may be considered to be a whole that is to be refined into a family of finer or more specific granules. In Fig. 2, the ordering of granules are given by dotted lines

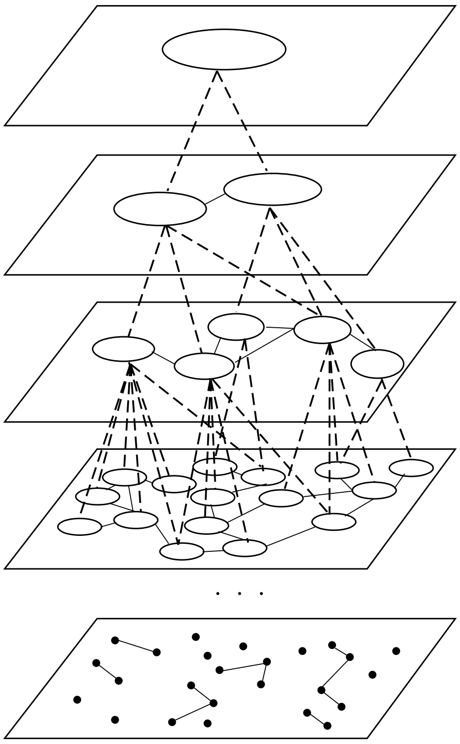


Fig. 2 Multilevel Hierarchical Granular Structure

between granules in two adjacent levels. It is reasonable to assume that the ordering relation is transitive. In Fig. 2, we omit the connections between no-adjacent levels, as they can be obtained by transitivity.

The ordering of granules leads naturally to the ordering of levels, the result is a multilevel granulation hierarchy as shown in Fig. 2. Although multilevel structures may be rather restrictive to reflect accurately about the reality, it does lead to simplification of computational issues and practical and reasonable approximate solutions to many real world problems.

12.2.1.4 Three Properties of Granules

A granule has at least three properties: internal properties, external properties, and contextual properties. The individual elements of a granule and their interactions determine internal properties of the granule; the external properties characterize the granule as an inseparable whole; a granule is only meaningful in a certain context. The individual elements of a granule are parts of the granule and are themselves granules. Actually, when forming a granule, we have to ignore the subtle differences between its elements, as well as the connections between each other. In other words, a group of elements are put into the same granule by considering their similar properties and connections and ignoring the subtle differences. We may focus on different issues and form granules with different granularity based on different situations in problem solving.

12.2.1.5 Relations and Operations on Granules

In a hierarchy, we consider three types of relations on granules. The in-level relations are relationships between granules within the same level. They show the interrelationships and interactions between granules at a particular level. In Fig. 2, they are depicted by solid lines connecting granules on the same plane. In fact, they can be used to study the internal properties of granules in the next upper level. Downward refinement relations show how a coarser granule can be refined into a family of finer granules in the next lower level. They are related to downward operations on granules, such as refinement operation, decomposition operation, or zoom-in operation on granules. Upward coarsening relations show how a family of finer granules is coarsened into a coarser granule in the next upper level. They are related to upward operations on granules, such as coarsening operation, composition operation or zoom-out operation on granules. In Fig. 2, the last two classes of relations are represented by dotted lines between granules in two adjacent levels.

In a hierarchy, we assume that operations on granules are only defined for granules in two immediately adjacent levels. By repeated applications of operations, it is possible to connect granules in all levels. This assumption, although restrictive, makes computation with a granulation hierarchy tractable.

12.2.2 Philosophy: Structured Thinking

Granular computing promotes structured approaches based on multilevel hierarchical granular structures. From existing studies, it may be observed that the philosophy of granular computing is not entirely new but draws extensively from reductionist thinking, levelism [3, 6, 11] and systems thinking [23].

Reductionist thinking focuses on breaking a complex problems into relatively simpler parts and inferring properties of the whole by a summary of properties of its parts. It offers an effective technique of analytic thinking. In a top-down fashion, a complex system or problem can be decomposed into many parts and these parts can be further divided if needed. When the top-down decomposition is applied to granular computing, a multilevel hierarchical granular structure may be derived. In a bottom-up manner, one may explain entities (i.e., granules) in an upper level based on entities in a lower level.

Levelism makes use of levels of abstraction and takes a hierarchical view that a complex problem may be divided and stratified into levels. The notion of levels may be interpreted in many different ways, including, for example, objective levels of reality, epistemological levels of understanding and explanation, methodological levels of study [3, 6, 11]. To a large extent, levelism is more relevant to granular computing, where the notion of levels of granularity plays a crucial role. Granular structures are stratified levels and levelist thinking leads to level-wise granular processing.

Systems thinking stresses on systemic properties of a whole system that are emerged from the composition, organization and interaction of its parts and cannot be reduced to the properties of its parts. It offers a technique of synthetic thinking for obtaining a holistic view of a system. Moreover, systems thinking also considers different system levels and has the ability to shift attention between system levels. In the context of granular computing, ideas from systems thinking can be used to study emergent properties at different levels of granularity.

Structured thinking of granular computing emphasizes the importance of useful multilevel structures that are common in reductionist thinking, levelist thinking and systems thinking. It takes advantage of those existing philosophical views, in order to develop its own philosophical standpoint.

12.2.3 Methodology: Structured Problem Solving

Methodology of granular computing is structured problem solving guided by structured thinking, involving the construction and utilization of multilevel, multiview granular structures. Granular computing relies on a set of practical heuristics and systematic approaches that make effective use of multiple levels of granularity. Like the classical 3R (reading, writing, and arithmetic), granular computing may be viewed as the fourth R, representing a set of problem solve skills that can be used by everyone.

The working principles of structured problem solving have been studied in many disciplines. In some sense, granular computing attempts to extract these principles and make them discipline independent and, hence, more accessible. As examples, we examine a few of such principles.

The first two principles concern mainly multilevel and multiview understanding and a plea for the use of granular structures. They can be stated as follows:

- the principle of multilevel view;
- the principle of multiview understanding.

While the multilevel principle emphasizes on multiple levels, i.e., viewpoints within a particular view, the multiview principle states the needs for a holistic understanding from many different angles. By the complementing nature of a family of hierarchies, the limitation of a particular hierarchy may be avoided, and a best hierarchy may be chosen.

The advantages of methodology of granular computing are derived from exploring granular structures. The next three principles deal with the exploration of granular structures. They are stated as:

- the principle of focused efforts;
- the principle of view switching;
- the principle of granularity conversion.

The principle of focused efforts is a kind of divide and conquer strategy and can be applied to both a family of hierarchies and a single hierarchy. With respect to a family of hierarchies, it requires that one concentrates on a particular hierarchy relatively independent of other hierarchies; with respect to a single hierarchy, it requires that one pays attention to a particular level without too much interference from other levels.

The principle of view switching demands a comparative study of the same problem under different views. It is necessary to switch to a different view when needed. The principle of granularity conversion guides top-down and bottom-up conversion with respect to a particular hierarchy. Through refinements and coarsening, one is able to work on different levels of granularity.

12.2.4 Computation: Structured Information Processing

Granular computing is a paradigm of structured information processing. It can be described by an information processing pyramid [2]. Granulations at multiple levels of differing granularity and their conversion are a base of granular information processing.

Representation and process are two basic tasks of computing with granular structures. A representation usually associates with a formal system that explicitly describes entities. Representations in granular computing must reflect multilevel and multiview structures. A process can be interpreted as actions that carry out information processing tasks. Downward refinement operations, upward coarsening operations and the zoom-in, zoom-out operations are examples of process.

With a multilevel granular structure, we have at least three modes of structured information processing. Starting with the top level, one can progressively compute in a top-down manner towards lower levels through refinement operations. An approximate solution obtained at a higher level is refined into a more accurate solution at a lower level. The process can be terminated whenever a required approximate solution is obtained. In contrast, one may start with the bottom levels and move in a bottom-up manner towards upper levels through coarsening operations. The processes can be terminate when the required level of abstraction is obtained. A third mode is a middle-out approach that starts with a particular level and moves up or down the hierarchy. In general, granular information processing may be an interactive process involving a mixture of the three modes. Moreover, a granular structure is not given prior to the processing, but is constructed during the processing.

12.3 Granular Computing and Concept Learning

By drawing correspondences between granules and concepts, categorization and granulations, granule construction and concept formation, we demonstrate that granule computing is relevant to concept learning.

12.3.1 Granules and Concepts

A concept is a mental representation or a mental symbol of human thought [13, 14]. There are many views of concept and concept learning, such as the classical view, prototype view, exemplar view and explanation-based view. Each view explains concepts and concept learning in an unique way, with its advantages and disadvantages. In this chapter, we adopt the classical view of concepts by establishing a correspondence between concepts and granules [32].

The classical view treats a concept as a basic unit of thought that consists of two parts, namely, the intension and extension. The intension is a subset of attributes or properties, which are valid for instances (objects) of the concept; the extension is the set of instances of the concept. That is, the intension provides an abstract description of common features shared by a set of instances, while extension employs the set of instances to interpret the concept. All objects or concrete examples in the extension have the same set of common properties described by the intension.

The meaning triangle proposed by Ogden and Richards [16] illustrates the classical view of a concept with an added node to represent natural language coding of a concept. As depicted in Fig. 3, the meaning triangle makes a distinction between *Word*, *Concept* and *Referent* in concept formation. The *Concept*, also called the intension, thought, or idea, represents an abstract entity usually associated with human thinking. The *Word*, also called symbol or name, is the concrete linguistic entity embodied in speech or written text. The *Referent*, also called the extension or object, represents a physical object in the external world. An arrow from *Word* to *Concept* shows that concepts are coded by a language. Another arrow from *Concept* to *Referent* describes that the extensional objects are mapped into intensional concepts through perception. The dashed line from *Word* to *Referent* depicts an indirect mapping through coding and perception [20, 22].

In the context of concept formation and learning, one may view a granule in terms of the meaning triangle of a concept. As showed in Fig. 4, a granule can be jointly characterized by three elements: *Name of a granule*, *Description of a granule* and *Instances in a granule*. The *Description of a granule* is a representation of a granule, which is a logic formula in this study, the *Instances in a granule* are the family of objects forming the granule, and the *Name of a granule* is a label assigned to the granule so that the granule can be conveniently referred to.

We may employ a logical language to study description of granules and a set-theoretic language to study instances of granules. According to the connection between concepts and granules, we have a formal definition of a granule in the context of concept learning.

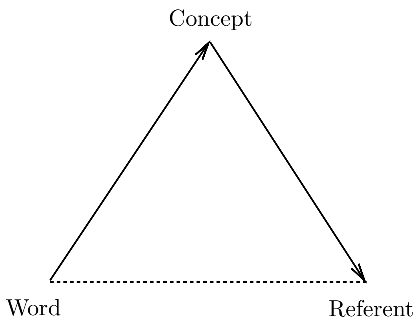


Fig. 3 The meaning triangle of a concept

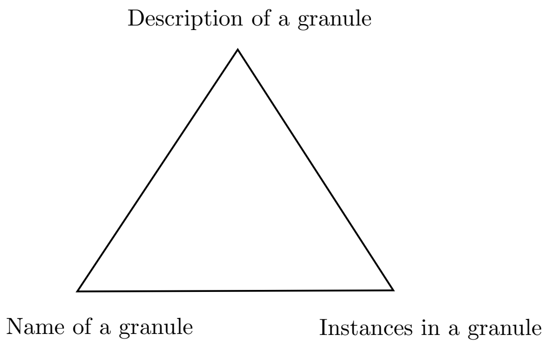


Fig. 4 The meaning triangle of a granule

Definition 1. A granule is defined as a triplet:

$$(g, i(g), e(g)),$$

where g is the name assigned to the granule, $i(g)$ is a representation of a granule, and $e(g)$ is a set of objects that are instances of the granule.

In many cases, $i(g)$ and g are the same. Suppose L is a logic language, U is a set of objects, and W is a set of words or labels. Then, $i(g)$ is a formula of language L, $e(g)$ is a subset of U, and g is an element from W.

12.3.2 Granulation and Classification

A granulation of a universe is a family of granules, with each granule being a subset of the universe. Two types of granulations, called partitions and coverings, can be formed. For simplicity, we only consider a finite nonempty universe.

Definition 2. A family of nonempty subsets of a finite universe U, $\pi = \{b_1, b_2, \ldots, b_k\}$, is called a partition of U. If it satisfies the following properties:

$$(P_1) \qquad b_i \cap b_j = \emptyset,\ for\ i \neq j, 1 \leq i, j \leq k, \tag{1}$$

$$(P_2) \qquad \bigcup \pi = \bigcup_{i=1}^{k} b_i = U, \tag{2}$$

each subset $b \in \pi$ is called a block of π.

Definition 3. Suppose Π is the family of all partitions on a finite universe U, a partial order (i.e., a reflexive, antisymmetric and transitive relation) can be defined on Π as follows, for $\pi_1, \pi_2 \in \Pi$,

$$\pi_1 \preceq \pi_2 \Longleftrightarrow \forall b \in \pi_1 \exists b' \in \pi_2 (b \subseteq b').$$

The partial ordering provides a refinement-coarsening relation over partitions. When $\pi_1 \preceq \pi_2$, π_1 is called a refinement of π_2 and π_2 a coarsening of π_1. The relation can be used to build multilevel granulations of a universe, which is a granular structure.

A partition of universe is often called a classification of the universe, where each block represents the extension of a concept. With respect to a partition, granules are pair-wise disjoint. As a generalization, one may consider a covering of a universe.

Definition 4. A family of nonempty subsets of a finite universe U, $\theta = \{c_1, c_2, \ldots, c_m\}$, is called a covering of U if it satisfies the condition,

$$(C_1) \qquad \bigcup \theta = \bigcup_{i=1}^{m} c_i = U. \tag{3}$$

It is called a non-redundant covering if it satisfies the condition:

$$(C_2) \qquad \bigcup (\theta - \{c\}) \neq U, \quad \forall c \in \theta. \tag{4}$$

That is, each of the subsets in θ is necessary.

Similar to partial ordering on partitions, one can introduce a relation on the family of all coverings.

Definition 5. Suppose Θ is the family of all coverings on a finite universe U. One can define a refinement-coarsening relation on Θ as,

$$\theta_1 \preceq \theta_2 \Longleftrightarrow \forall c \in \theta_1 \exists c' \in \theta_2 (c \subseteq c'). \tag{5}$$

The ordering relation $\preceq$ is reflexive and transitive, but not necessarily antisymmetric. When the relation is restricted to the family of all non-redundant coverings, it is also antisymmetric.

12.3.3 Concept Learning as Searching

An interesting result in cognitive science claims that a concept can be represented by a family of rules that distinguish objects and determine the categorization to which they might belong [8]. This immediately leads to rule-based concept learning. In concept formation and learning, once a concept is formed, it can be used to define new concepts. That is, a crucial task of concept formation and learning is to define new concepts by using existing known concepts.

12.3.3.1 Learning a Single Concept

Consider first the task of learning a single concept. Suppose K is a family of subsets of a finite universe U, representing extensions of a family of known concepts. Let $e(g) \in K$ be the extension of a known concept, and $C = e(c) \subseteq U$ the extension of a target concept named c that is to be learned. When $e(g) \subseteq C$, we can formulate a rule,

$$i(g) \rightarrow i(c),$$

That is, if an object is described by $i(g)$ then it is an instance of concept c. In this way, we partially define an unknown concept c by a known concept g. In general, one can form a family of rules $i(g_i) \rightarrow i(c), i = 1, 2, \ldots, p$, to represent c by several known concepts based on the condition $e(g_1) \cup e(g_2) \cup \cdots \cup e(g_p) \subseteq C$.

When defining an unknown concept, one may expect that number of rules are as small as possible and each rule is as general as possible.

Definition 6. Suppose C is the extension of an unknown concept and $e(g_1), e(g_2) \in K$ are extensions of two known concepts, c is the name of the unknown concept. If $e(g_1) \subseteq C, e(g_2) \subseteq C$ and $e(g_1) \subseteq e(g_2)$, we say that $e(g_2)$ is more general than $e(g_1)$ and $e(g_1)$ is more specific than $e(g_2)$ in specifying C.

Based on the set-inclusion relation on extensions of concepts, one can design algorithms to search for the best description of unknown concepts by some known concepts.

12.3.3.2 Learning a Classification

For a classification problem, we have a target classification and a family of known concepts, the task is to represent the target classification through known concepts. A solution to a classification problem can be modeled as a search in the space of partitions or coverings whose elements are known concepts.

Definition 7. Let $\pi_D = \{D_1, D_2, \ldots, D_t\}$ denotes a target classification. A partition π is called a partition solution to the classification problem if

$$\pi \preceq \pi_D.$$

Similarly, a covering θ is called a covering solution if

$$\theta \preceq \pi_D.$$

If $\pi \preceq \pi_D$ is a solution, then each $b \in \pi$ is included in a class $D_j \in \pi_D$, namely, $b \subseteq D_j$. Thus, we can form a classification rule,

$$i(b) \rightarrow i(D_j),$$

A family of classification rules can be formulated based on all blocks in π.

In practical situations, an arbitrary solution may not be satisfactory. One needs to impose certain conditions. For example, one may require that blocks in π are as large as possible, in other words, their corresponding concepts are as general as possible. This requirement can be precisely stated based on the refinement-coarsening relation $\preceq$ on Π.

Definition 8. Suppose $\pi \preceq \pi_D$ and $\pi' \preceq \pi_D$ are two solutions to the classification π_D. If $\pi \preceq \pi'$, we say that π' is a more general solution than π.

Definition 9. A solution $\pi \preceq \pi_D$ is called a maximal general solution if there do not exists another solution $\pi' \preceq \pi_D$ such that $\pi' \neq \pi$ and $\pi \preceq \pi'$.

A maximal general solution may not be unique and there may exist many maximal general solutions. Based on the notations introduced so far, the problem of learning classification rules may be modeled as a search for a partition such that $\pi \preceq \pi_D$. In particular, one may search for a maximal general solution. The refinement-coarsening relation $\preceq$ provides a search direction. One way is to start from a more specific solution, i.e., a finer partition, to move towards a more general solution, i.e., a coarser partition. Alternatively, one may start from a coarser partition that is not a solution and then refine it until a solution is obtained.

The same framework can be easily applied to search for a covering solution. In this case, one needs to consider additional properties on a covering. For example, a covering must be non-redundant so that no redundant rules would be produced. A covering must contain subsets that have least overlap so that a set of less overlapping classification rules would be generated.

12.4 A Model for Learning a Classification

Based on the discussions of the previous sections, we present a model for learning a classification. The model explicitly considers the following issues raised in Section 12.3:

- Define a logic language for representing intensions of concepts;
- Construct a family of known concepts;
- Construct a space of partitions for the purpose of searching for a partition solution to a classification;
- Construct a space of coverings for the purpose of searching for a covering solution to a classification.

These issues are investigated by drawing results from rough set theory. A partition-based learning strategy, also called an attribute-oriented strategy, is obtained from algorithms such as ID3 [18], C4.5 [19] and reduct construction in rough set theory [17]. A covering-based strategy, also called an attribute-value-oriented strategy, is obtained from the class of sequential covering algorithms such as the PRISM algorithm [4] and LERS algorithms [9].

12.4.1 A Decision Logic Language in an Information Table

For precisely defining intension and extension of a concept, rough set theory uses an information table, in which a set of objects is described by a set of attributes.

Definition 10. An information table is a system $S = (U, At, \{V_a \mid a \in At\}, \{I_a \mid a \in At\})$, where U is a finite nonempty universe of objects, At is a finite nonempty set of attributes, V_a is the domain of attribute $a \in At$, and $I_a : U \to V_a$ is an information function that maps an object $x \in U$ to a value $v \in V_a$.

We consider a logic language in an information table defined by using attribute-value pairs and logic conjunctive operator $\wedge$.

Definition 11. A decision logic language (DL) is recursively defined as:

1. Atomic formulas: for any $a \in At, v \in V_a$, $(a = v)$ is an atomic formula.
2. Composite formulas: if ϕ and ψ are formulas, $\phi \wedge \psi$ is a formula.

The logic language defined in this study is only a sublanguage of a decision logic language used in rough set theory [17]. In particular, we are only interested in concepts that are conjunctively defined by a set of atomic formulas.

The semantics of formulas in DL is defined through a meaning assignment that associates each formula with a subset of objects in U, based on the satisfiability of a formula by an object [17, 34].

Definition 12. An object $x \in U$ satisfies an atomic formula $a = v$, written $x \models (a = v)$, if $I_a(x) = v$; it satisfies a formula $\phi \wedge \psi$ if it satisfies both ϕ and ψ, namely, $x \models \phi \wedge \psi$ if and only if $x \models \phi$ and $x \models \psi$.

Definition 13. Suppose DL is the set of all formulas of the language DL. The meaning of a formula $\phi \in DL$ is a subset of objects defined by:

$$m(\phi) = \{x \in U \mid x \models \phi\}. \tag{6}$$

It is possible that two distinct formulas may produce the same meaning set in an information table. This suggests that a concept may have more than one description. The meaning assignment satisfies the following two properties:

$$\begin{aligned} &(m_1) \qquad m(a = v) = \{x \in U \mid I_a(x) = v\}, \\ &(m_2) \qquad m(\phi \wedge \psi) = m(\phi) \cap m(\psi). \end{aligned}$$

We can compute the meaning of a composite formula from its atomic formulas. By the construction of formulas in DL, one can more conveniently represent a formula by a set of attribute-value pairs as follows. Let $A(\phi)$ denote the set of all attribute-value pairs corresponding to atomic formulas in ϕ. It follows that $A(a=v)=\{(a,v)\}$ and $A(\phi \wedge \psi)=A(\phi) \cup A(\psi)$. In the following discussion, we will use ϕ and $A(\phi)$ interchangeably.

With the introduced logic language, we have a precise representation of a concept or a granule. The intension of a concept is given by a logic formula of DL and the extension is the meaning of the formula.

12.4.2 Conjunctively Definable Concepts

The language DL enables us to precisely define a concept by a logic formula. However, due to the expressive power and the limitation of the set of attributes, we may not find a formula for an arbitrary subset of objects. This implies that the language DL only allows us to define a sub-family of the power set 2^U of U.

Definition 14. Suppose that a subset of objects $C \subseteq U$ represents the extension of a concept. We say that C is a conjunctively definable set or concept under DL if and only if there exists a formula $\phi \in DL$ such that

$$C = m(\phi). \tag{7}$$

Otherwise, it is conjunctively undefinable. The set of all conjunctively definable concepts is given by:

$$K = \{m(\phi) \mid \phi \in DL\} \subseteq 2^U. \tag{8}$$

The family of conjunctively definable concepts consists of the building blocks, namely, known concepts, from which new unknown concepts may be learned. That is, we use known concepts to express other concepts.

12.4.3 Attribute-Oriented Search Strategies in a Space of Partitions Defined by Subsets of Attributes

A crucial step in learning a classification is to construct a space of partitions based on the family of conjunctively definable sets K. Theoretically speaking, one can obtain such a space by collecting all partitions whose blocks are definable concepts from K. However, constructing and searching the space of all such partitions is practically difficult. Instead, one may use a subspace by considering partitions with additional properties [36]. Rough set theory uses a space of partitions defined by subsets of attributes.

In an information table, each subset of attributes defines an equivalence relation on U.

Definition 15. Suppose $P \subseteq At$ is a subset of attributes, an equivalence relation defined by P is given by: for $x, y \in U$,

$$xE_P y \Longleftrightarrow \forall a \in P(I_a(x) = I_a(y)). \tag{9}$$

That is, x and y are equivalent if they have the same values on all attributes in P. The induced partition is given by $U/E_P = \{[x]_{E_P} \mid x \in U\}$, where $[x]_{E_P} = \{y \in U \mid xE_Py\}$ is the equivalence class containing x.

It can be verified that $[x]_{E_P}$ is defined by a logic formula $\bigwedge_{a\in P} a = I_a(x)$, that is, $m(\bigwedge_{a\in P} a = I_a(x)) = [x]_{E_P}$. Thus, $U/E_P \subseteq K$ is a partition whose blocks are conjunctively definable sets. More importantly, the set-inclusion relation on subsets of attributes leads to a refinement-coarsening relation on their induced partitions. For two subsets of attributes, $P_1, P_2 \subseteq At$ with $P_1 \subseteq P_2$, the following implication holds:

$$P_1 \subseteq P_2 \Longrightarrow U/E_{P_2} \preceq U/E_{P_1}. \tag{10}$$

Consequently, searching in the space $\Pi_{At} = \{U/E_P \mid P \subseteq At\}$ may be viewed as searching in the space 2^{At}.

In rough set theory, searching for a partition solution to a classification is formulated as finding an attribute reduct.

Definition 16. Suppose $S = (U, At = C \cup D, \{V_a\}, \{I_a\})$ is a consistent classification or a decision table, where C is the set of condition attributes, D is the set of classification or decision attribute and $\pi_C \preceq \pi_D$. A subset $R \subseteq C$, is called a relative reduct of C with respect to D, if R satisfies two conditions:

1. $U/E_R \preceq U/E_D$;
2. $\forall a \in R, \neg(U/E_{(R-\{a\})} \preceq U/E_D)$.

Condition (1) suggests that π_R is a solution to the classification π_D. That is, attributes in R are jointly sufficient. Suppose $P \subseteq C$ is a subset of attributes with $\pi_P \preceq \pi_D$. An attribute in P is said to be redundant if $\pi_{P-\{a\}} \preceq \pi_D$. Condition (2) of Definition 16 states that R does not contain any redundant attributes. That is, attributes in R are individually necessary. The two conditions together imply that π_R is a maximal general solution to π_D in Π_C.

There are three searching strategies for finding a reduct, namely, deletion strategy, addition-deletion strategy and addition strategy [35]. A deletion strategy starts from the entire set of conditional attributes and sequentially deletes redundant attributes. An addition-deletion strategy starts with the empty set and sequentially adds attributes until a subset of attributes satisfying condition (1) of Definition 16. is obtained; it then delete redundant attributes. An addition strategy only adds attributes that will form a reduct.

Fig. 5 gives an algorithm that implements a deletion strategy for constructing a reduct. The algorithm starts from the most specific partition solution π_C. At each iteration of the while loop, an attribute may be deleted to generate a more general solution. When every attribute is checked, the algorithm produces a reduct that provides a maximal general solution. Many authors have introduced and studied fitness functions, including dependency measures, mutual information, conditional entropy and others.

Algorithm 1: Construction of a Relative Reduct by Deletion

```
Input: A classification table $S = (U, At = C \cup D, \{V_a\}, \{I_a\})$;
       A fitness functin $\delta$ ;
Output: A relative reduct $R$
1  begin
2      $R = C$;
3      $unchecked = C$;
4      while $unchecked \neq \emptyset$ do
5          Compute the fitness of all the attributes in $unchecked$ using the fitness function
           $\delta$;
6          Select an attribute $a \in unchecked$ with minimum fitness;
7          $unchecked = unchecked - \{a\}$;
8          if $\pi_{R-\{a\}} \preceq \pi_D$ then $R = R - \{a\}$;
9      end
10 end
```

Fig. 5 Attribute-oriented Relative Reduct Construction

12.4.4 Attribute-Value-Oriented Search Strategies in a Space of Coverings Defined by Families of Sets of Attribute-Value Pairs

The partition based model can be modified to formulate a covering based model. Recall that each definable concept in K is defined by a conjunction of a family of atomic formula and it can be eventually expressed by the corresponding set of attribute-value pairs. Thus, a covering $\theta \subseteq K$ of the universe U can be viewed as a family of sets of attribute-value pairs. This leads to attribute-value-oriented search strategy in a space of coverings.

The set of all definable set K is a covering of U. The problem of finding a covering solution can be formulated in terms of a reduct of K relative to π_D.

Definition 17. A covering $\theta_R \subseteq K$ is called a relative reduct of K with respect to π_D if θ_R satisfies the following conditions:

1. $\theta_R \preceq \pi_D$;
2. For any $c \in \theta_R, \forall c' \in K (c \subset c' \Longrightarrow \neg(((\theta_R - \{c\}) \cup \{c'\}) \preceq \pi_D))$;
3. For any $c \in \theta_R, \neg(\bigcup(\theta_R - \{c\}) = U \wedge ((\theta_R - \{c\}) \preceq \pi_D))$.

Condition (1) states that the covering must be a solution to π_D. The covering $(\theta_R - \{c\}) \cup \{c'\}$ is produced by replacing c in θ_R with one of its proper supersets in K. Condition (2) states that such a replacement will not produce a new solution. This implies that each set in θ_R must be maximal. Condition (3) states that each set in θ_R is necessary; the deletion of any of them will produce a family of sets that is either not a covering or not a covering solution to π_D.

Algorithm 2: Construction of a Relative Attribute-Value Reduct

```
Input: A decision table $S = (U, At = C \cup D, \{V_a\}, \{I_a\})$;
       a fitness function $\gamma$ ;
Output: A relative attribute-value reduct $\theta_R$
begin
    $\theta_K = \{c \in K \mid \exists D_j \in \pi_D (c \subseteq D_j)\}$;
    $\theta_K^* = \{c \in \theta_K \mid \neg(\exists c' \in \theta_K (c \subset c'))\}$;
    $\theta_R = \theta_K^*$;
    $unchecked = \theta_R$;
    while $unchecked \neq \emptyset$ do
        Compute the fitness of all the sets in $unchecked$ using the fitness function $\gamma$ ;
        Select a set $c \in unchecked$ with minimum fitness ;
        $uncheked = unchecked - \{c\}$;
        if $\cup(\theta_R - \{c\}) = U \wedge (\theta_R - \{c\}) \preceq \pi_D$ then $\theta_R = \theta_R - \{c\}$;
    end
end
```

Fig. 6 Attribute-value-oriented Relative Reduct Construction

By the definition of a relative attribute-value reduct, we can select a family of sets from K based on conditions (1)-(3) of Definition 17. In order to ensure condition (1), each set in a covering must be a subset of a class in π_D. Thus, we construct a covering solution to π_D as follows:

$$\theta_K = \{c \in K \mid \exists D_j \in \pi_D (c \subseteq D_j)\}$$

According the requirement that each set in a covering must be a maximal set by condition (2), we simplify θ_K by removing the sets in θ_K that are subsets of other sets in θ_K:

$$\theta_K^* = \{c \in \theta_K \mid \neg(\exists c' \in \theta_K (c \subset c'))\}.$$

That is, θ_K^* is a covering such that no set in θ_K^* is a subset of another set in θ_K^*. Finally, we can delete sequentially redundant set in θ_K^*.

Fig. 6 is an algorithm for constructing a covering. It should be pointed out that the proposed algorithm is a straightforward implementation based on the definition of relative reduct. One may consider more efficient implementations based on existing sequential covering algorithm such as PRISM [4] and LERS [9, 10].

12.5 Conclusion

This chapter examines a granular computing paradigm for concept learning. To a large degree, it reformulates and reinterprets some existing results in the proposed paradigm with two purposes. One is to provide a concrete model of granular computing and the other to provide a different view for concept learning.

The triarchic theory of granular computing is briefly summarized. The connections between granules in granular computing and concepts in concept learning are established. We argue that a concept may be expressed by a granule based on the classical view of concepts. For classification, the notion of a granulation, i.e., a family of granules, is introduced. Two types of granulation, namely, partitions and coverings, are investigated. The two types of granulation lead to two classes of strategies for solving a classification problem. A model for learning a classification is proposed. Based on the rough set theory and other concept learning algorithms, two strategies are introduced, namely, an attribute-oriented strategy for searching a space of partitions and an attribute-value oriented strategy for search space of coverings. More importantly, the notion of a relative attribute-value reduct is formally expressed, which is complementary to the widely used notion of a relative attribute reduct.

References

1. Ahl, V., Allen, T.F.H.: Hierarchy Theory: A Vision, Vocabulary, and Epistemology. Columbia University Press, New York (1996)
2. Bargiela, A., Pedrycz, W.: Granular Computing: An Introduction. Kluwer Academic Publishers, Boston (2002)
3. Brown, H.C.: Structural levels in the scientist's world. The Journal of Philosophy, Psychology and Scientific Methods 13, 337–345 (1916)
4. Cendrowska, J.: PRISM: An algorithm for inducing modular rules. International Journal of Man-Machine Studies 27, 349–370 (1987)
5. Craik, F.I.M., Lockhart, R.S.: Levels of processing: A framework for memory research. Journal of Verbal Learning and Verbal Behavior 11, 671–684 (1972)
6. Floridi, L.: The method of levels of abstraction. Minds and Machines 18, 303–329 (2008)
7. Foster, C.L., Foster, C.: Algorithms, Abstraction and Implementation: Levels of Detail in Cognitive Science. Academic Press, London (1992)
8. Goldstone, R.L., Kersten, A.: Concepts and categorization. In: Alice, F.H., Robert, W.P., Irving, B.W. (eds.) Handbook of Psychology, Experimental Psychology, pp. 597–621. Wiley, New Jersey (2003)
9. Grzymala-Busse, J.: LERS - A system for learning from examples based on rough sets. In: Slowinski, R. (ed.) Intelligent Decision Support: Handbook of Applications and Advances of the Rough Sets Theory, pp. 3–18. Kluwer Academic Publishers, Boston (1992)
10. Grzymala-Busse, J., Rzasa, W.: Approximation space and LEM2-like algorithms for computing local coverings. Fundamenta Informaticae 85, 205–217 (2008)
11. Heil, J.: Levels of reality. Ratio 16, 205–221 (2003)
12. Hobbs, J.R.: Granularity. In: Proceedings of the 9th International Joint Conference on Artificial Intelligence, pp. 432–435 (1985)
13. Hunt, E.B.: Concept Learning: An Information Processing Problem. Wiley, New York (1962)
14. Margolis, E., Laurence, S.: The ontology of concepts - abstract objects or mental representations? Noûs 41, 561–593 (2007)
15. Medin, D.L., Smith, E.E.: Concepts and concept formation. Annual Review of Psychology 35, 113–138 (1984)

16. Ogden, C.K., Richards, I.A.: The Meaning of Meaning: A Study of the Influence of Language upon Thought and of the Science of Symbolism, 8th edn. Harcourt Brace, New York (1946)
17. Pawlak, Z.: Rough Sets: Theoretical Aspects of Reasoning about Data. Kluwer Academic Publishers, Boston (1991)
18. Quinlan, J.R.: Induction of decision trees. Machine Learning 1, 81–106 (1986)
19. Quinlan, J.: C4.5: Programs for Machine Learning. Morgan Kaufmann, San Francisco (1993)
20. Regoczei, S., Hirst, G.: The meaning triangle as a tool for the acquisition of abstract, conceptual knowledge. International Journal of Man-Machine Studies 33, 505–520 (1990)
21. Smith, E.E., Medin, D.L.: Categories and Concepts. Harvard University Press, Cambridge (1981)
22. Tzafestas, S.G.: Knowledge Based Systems: Advanced Concepts, Techniques & Applications. World Scientific, New Jersey (1997)
23. Wuketits, F.M.: Synthetic and analytical thinking. Fresenius' Journal of Analytical Chemistry 326, 320–323 (1987)
24. Yao, Y.Y.: Information granulation and rough set approximation. International Journal of Intelligent Systems 16, 87–104 (2001)
25. Yao, Y.Y.: Perspectives of granular computing. In: Proceedings of the 2005 IEEE International Conference on Granular Computing, vol. 1, pp. 85–90 (2005)
26. Yao, Y.Y.: Three perspectives of granular computing. Journal of Nanchang Institute of Technology 25, 16–21 (2006)
27. Yao, Y.Y.: Granular computing for data mining. In: Proceedings of SPIE Conference on Data Mining, Intrusion Detection, Information Assurance, and Data Networks Security (2006), doi:10.1117/12.669023
28. Yao, Y.Y.: The Art of Granular Computing. In: Kryszkiewicz, M., Peters, J.F., Rybiński, H., Skowron, A. (eds.) RSEISP 2007. LNCS (LNAI), vol. 4585, pp. 101–112. Springer, Heidelberg (2007)
29. Yao, Y.Y.: A unified framework of granular computing. In: Pedrycz, W., Skowron, A., Kreinovich, V. (eds.) Handbook of Granular Computing, pp. 401–410. Wiley, New York (2008)
30. Yao, Y.Y.: Granular computing: past, present, and future. In: Proceedings of the 2008 IEEE International Conference on Granular Computing, pp. 80–85 (2008)
31. Yao, Y.Y.: Integrative levels of granularity. In: Bargiela, A., Pedrycz, W. (eds.) Human-Centric Information Processing Through Granular Modelling, pp. 31–47. Springer, Berlin (2009)
32. Yao, Y.Y.: Interpreting concept learning in cognitive informatics and granular computing. IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics 39, 855–866 (2009)
33. Yao, Y.Y.: Human-inspired granular computing. In: Yao, J.T. (ed.) Novel Developments in Granular Computing: Applications for Advanced Human Reasoning and Soft Computation, pp. 1–15. Information Science Reference, Herskey (2010)
34. Yao, Y.Y., Zhou, B.: A logic language of granular computing. In: Proceedings of the 6th IEEE International Conference on Cognitive Informatics, pp. 178–185 (2007)
35. Yao, Y.Y., Zhao, Y., Wang, J.: On reduct construction algorithms. In: Proceedings of the 1st International Conference on Rough Sets and Knowledge Technology. LNCS(LNAI), vol. 6042, pp. 297–304 (2006)
36. Zhao, Y., Yao, Y.Y., Yao, J.T.: Level-wise construction of decision trees for classification. International Journal of Software Engineering and Knowledge Engineering 16, 103–126 (2006)

Part B

Applications

Chapter 13
Identifying Calendar-Based Periodic Patterns

Jhimli Adhikari and P.R. Rao

Abstract. A large class of problems deals with temporal data. Identifying temporal patterns in these datasets is a natural as well as an important task. In the recent time, researchers have reported an algorithm for finding calendar-based periodic pattern in a time-stamped data and introduced the concept of certainty factor in association with an overlapped interval. In this paper, we have extended the concept of certainty factor by incorporating support information for effective analysis of overlapped intervals. We have proposed a number of improvements in the algorithm for identifying calendar-based periodic patterns. In this direction we have proposed a hash based data structure for storing and managing patterns. Based on our modified algorithm, we identify full as well as partial periodic calendar-based patterns. We provide a detailed data analysis incorporating various parameters of the algorithm and make a comparative analysis with the existing algorithm, and show the effectiveness of our algorithm. Experimental results are provided on both real and synthetic databases.

Keywords: Calendar-based pattern, Certainty factor, Overlapped interval, Periodic pattern, Temporal pattern, Time-stamped database.

13.1 Introduction

A large amount of data being collected every day has a temporal connotation. For example, databases those originate from transactions in a supermarket, logs in a network, transactions in a bank, and events related to manufacturing industry are

Jhimli Adhikari
Department of Computer Science, Narayan Zantye College,
Bicholim, Goa - 403 529, India
e-mail: `jhimli_adhikari@yahoo.co.in`

P.R. Rao
Department of Computer Science and Technology, Goa University, Goa - 403 206, India
e-mail: `pralhadrrao@gmail.com`

all inherently related to time. Data mining techniques could also be applied to these databases to discover various temporal patterns to understand the behavior of customers, markets, or monitored processes in different points of time. Temporal data mining is concerned with the analyses of data to find out patterns and regularities from a set of temporal data. In this context sequential association rule [5], periodical association rule [17], calendar association rule [18], calendar-based periodic pattern [20], and up-to-date pattern [12] are some interesting temporal patterns reported in the recent time.

For effective management of business activities, we often wish to discover knowledge from time-stamped data. There are several important aspects of mining time-stamped data including trend analysis, similarity search, forecasting and mining of sequential and periodic patterns. In a database from a retail store, the sales of ice cream in summer and the sales of blanket in winter should be higher than those of the other seasons. Such seasonal behaviour of specific items can only be discovered when a proper window size is chosen for the data mining process [22]. A supermarket manager may discover that turkey and pumpkin pie are frequently sold together in November in every year. Discovering such patterns may reveal interesting information that can be used for understanding the behaviour of customers, markets or monitored processes in different time periods. However, these types of seasonal patterns cannot be discovered by traditional non-temporal data mining approaches that treat all the data as one large segment with no attention paid to utilizing the time information of the transactions. If one looks into the entire dataset rather than the transactions that occur in November, it is likely that one will not be able to discover the pattern of turkey and pumpkin pie since the overall support for them will be evidently low. In general, a time-stamped database might exhibit some periodic behaviours. Length of a period might vary from one context to another context. For example, in case of sales of ice cream, the basic time interval could be of three months, since in many regions March, April and May together is considered as summer. Also, in case of sales of blanket, the basic time interval could be considered from November to February in every year. In addition, in many business applications, one might be interested in quarterly patterns over the years, where length of the period is equal to three months. A large amount of data is collected every day in the form of event time sequences. These sequences are valuable sources to analyze not only the frequencies of certain events, but also the patterns with which these events occur. For example, from data consisting of web clicks one may discover that a large number of web browsers who visit *www.washingtonpost.com* in morning hours also visit *www.cnn.com*. Using such information one can group users as daily morning users, daily evening users, weekly users, etc. This information might be useful for communicating to the users. Temporal patterns in a stock market, such as whether certain months, days of the week, time periods or holidays provide better returns than other time periods have received particularly a large amount of attention. Due to the presence of various types of applications in many fields, periodic pattern mining is an interesting area of study.

Mahanta et al. [20] used set superimposition [8] to find the membership value of each fuzzy interval. The concept of set *superimposition* is defined as follows. If set A is superimposed over set B or set B is superimposed over set A then set superimposition operation can be expressed as A (S) $B = (A - B)$ (+) $(A \cap B)^{(2)}$ (+) $(B - A)$, where (S) denotes the set superimposition operation. Here, the elements of $(A \cap B)^{(2)}$ are the elements of $(A \cap B)$ represented twice and (+) represents union of disjoint sets. Authors have also designed an algorithm for mining calendar-based periodic patterns. While applying this concept authors have assumed equi-fuzzy intervals, and accordingly the concept of certainty factor has been proposed for each sub-interval. Certainty factor of an interval over different time periods expresses the likelihood of reporting the pattern in that particular interval. If two intervals overlap then the certainty factor is more for the overlapped region than the non-overlapped region. When two intervals are superimposed, authors have assumed 1/2 equi-fuzzy membership value for each interval. After superimposition, the fuzzy membership value for the overlapped region becomes 1. The fuzzy membership value for non-overlapped region remains 1/2. But these two intervals may have different supports for the pattern. The *support* [3] of a pattern represents a fraction of transactions containing the pattern. A pattern is *frequent* if its support is greater than equal to a user-defined threshold, *minsupp*. The certainty factor and support of a pattern in an interval are two different concepts. For an effective analysis of overlapped regions, these two concepts need to be introduced along with an overlapped region. Thus, in this paper we propose an extended analysis of superimposed intervals. The main weak point of the aforementioned paper is that the concept of set superimposition is not necessary in the proposed algorithm. Therefore, we have proposed a modified algorithm for identifying full as well as partial calendar-based periodic patterns. We have also improved our algorithm by introducing a hash based data structure for storing relevant information associated with intervals. In addition, we have suggested some other improvements in the proposed algorithm. Before concluding this section, we give an example of a time-stamped database that will be used for providing illustrative examples on various concepts.

Example 1. Consider the following database D of transactions. Each record contains items purchased as well as the date of the transaction. •

Table 1 A sample time-stamped database

time-stamp	items	time-stamp	items	time-stamp	items
29/03/1990	*a*, *b*, *c*	07/04/1992	*a*, *c*, *e*, *g*, *h*	17/04/1993	*a*, *c*, *f*
06/04/1990	*a*, *c*, *e*	12/04/1992	*c*, *e*	06/04/1994	*a*, *b*, *c*,*d*
21/04/1990	*a*, *d*	14/04/1992	*c*, *e*, *f*	10/04/1994	*g*, *h*
25/04/1990	*a*, *c*, *d*	19/04/1992	*f*, *g*	13/04/1994	*a*, g
06/03/1991	*a*, *c*	04/03/1993	*a*, *c*	18/04/1994	*g*, *h*, *i*
12/03/1991	*a*, *c*, *e*	09/03/1993	*a*, *c*, *g*	20/04/1994	*a*, *c*, *e*, *f*
19/04/1991	*f*, *g*	01/04/1993	*c*, *h*, *i*		
03/03/1992	*a*, *c*, *d*	07/04/1993	*c*, *d*		

We have omitted the time of a transaction, since our data analysis is not associated with the time component of a transaction. We will refer to this database from time to time for the purpose of illustrating various concepts.

Rest of the paper is organized as follows. We discuss related work in Section 2. In Section 3, we have discussed calendar-based periodic patterns and proposed an extended certainty factor of an interval. We have designed an algorithm for identifying calendar-based periodic patterns in Section 4. Experimental results are provided in Section 5. We conclude the paper in Section 6.

13.2 Related Work

A calendar time expression is composed of calendar units in a specific calendar and represents different time features, such as an absolute time interval and a periodic time over a specific time period. A calendar-based periodic pattern is associated with time hierarchy for calendar years. In this paper we have dealt with calendar dates over the years.

Verma et al. [23] have proposed an algorithm H-Mine, where a header table H is created separately for each interval. Each frequent item entry has three fields viz., item-id, support count and a hyper-link. In order to deal with the patterns in time-stamped databases we have proposed a hash-based data structure where at the first index level we store distinct years that appear in the transactions. Then we keep an array of pointers corresponding to every year in the index table. The *k-th* pointer of this array points to tables containing interesting itemsets of size *k*.

Lee et al. [15] have proposed two data mining systems for discovering fuzzy temporal association rules and fuzzy periodic association rules. The mined patterns are expressed in fuzzy temporal and periodic association rules that satisfy the temporal requirements specified by the user. In the proposed algorithm the mined patterns are dependent on user inputs such as maximum gap between two intervals and minimum length of an interval.

Li et al. [18] proposed two classes of temporal association rules, temporal association rules with respect to precise match and temporal association rules with respect to fuzzy match, to represent regular association rules along with their temporal patterns. Our work differs from it, since we identify frequent itemsets along with the associated intervals. Then we use match ratio to determine whether a pattern is full periodic or partial periodic. Subsequently, Zimbrao et al. [25] reported a similar work. Authors incorporate multiple granularities of time intervals from which both cyclic and user-defined calendar patterns can be achieved. Ale and Rossi [6] proposed an algorithm to discover temporal association rules. In this algorithm, support of an item is calculated only during its lifespan. In the proposed work we compute and store supports of itemsets when they satisfy the requirements of the user.

Lee et al. [16] have proposed a technique for mining partial multiple periodic patterns without redundant rules. Without mining every period, authors checked the necessary period and used this information to do further mining. Instead of considering the whole database, the information needed for mining partial periodic patterns is transformed into a bit vector that can be stored in a main memory. This approach needs to scan the database at the most two times. Our approach extracts both partial and full periodic patterns together by scanning the database repeatedly to find the higher-level patterns as done using apriori algorithm [4].

In the context of support definition, Kempe et al. [13] have proposed a new support definition that counts the number of pattern instances, handles multiple instances of a pattern within one interval sequence and allows time constraints on a pattern instance.

Lee et al. [14] have proposed a new temporal data mining technique that can extract temporal interval relation rules from temporal interval data by using Allen's theory [7]. Authors designed a preprocessing algorithm for generalization of temporal interval data. Also, authors have proposed an algorithm for discovering a temporal interval relation. Although there are thirteen different types of relations between two intervals, in our work we have focused on only overlapped intervals to find locally frequent itemsets of larger size and detect periodicity of patterns.

Ozden et al. [21] proposed a method of finding patterns having periodic nature where the period has to be specified by the user. Han et al. [11] proposed several algorithms for mining partial periodic patterns by exploring some interesting properties such as the apriori property and the max-subpattern hit set property by shared mining of multiple periods.

13.3 Calendar-Based Periodic Patterns

In Section 1, we have presented some important applications of calendar-based periodic patterns. A calendar-based periodic pattern is dependent on the schema of a calendar. There are various ways one could define the schema of a calendar. We assume that the schema of calendar-based pattern is based on day, month and year. This schema is also useful to determine weekly-based pattern, since first seven days of any month correspond to the first week, days 8 to14 of any month correspond to the second week, and so on. Thus, one can have several types of calendar-based periodic patterns viz., daily, weekly, monthly and yearly. Based on a schema, some examples of calendar patterns are given as follows: every day of January, 1999; every 16-th day of January in each year; second week of every month. Again, each of these periodic patterns could be of two types viz., partially periodic pattern and full periodic pattern. A problem related to periodicity could be of finding patterns occurring at regular time intervals. Thus it emphasizes on two aspects viz., pattern and interval.

A calendar pattern refers to a market cycle that repeats periodically on a consistent basis. Seasonality could be a major force in a marketplace. While calendar patterns are based on a framework of multiple time granularities viz., day, month and year, but the periodic patterns are defined in term of a single granularity. Here patterns are dependent on the lifespan of an item in a database. Lifespan of an item (x) is a pair (x, [t_1, t_2]), where t_1 and t_2 denote the time that the item x appears in the database for the first time and last time, respectively. The problem of periodic pattern mining can be categorized into two types. One is full periodic pattern mining, where every point in time granularity [9] contributes to a cyclic behavior of the pattern. The other and more general one is called partial periodic pattern mining, which specifies the behavior of the pattern at some but not all points of time granularity [9] in the database. Partial periodicity is a looser form of periodicity than full periodicity, and it also occurs more commonly in a real world database. A pattern is associated with a real number m ($0 < m < 1$), called match ratio [18] that reveals that a pattern holds with respect to fuzzy match satisfying at least $100m\%$ of the time intervals. Match ratio is an important measure which determines whether a calendar-based pattern could be full periodic or partial periodic. When the match ratio is equal to 1 then it is a full periodic pattern. In case of partial periodic pattern the match ratio lies between 0 and 1. While finding yearly periodic patterns, Mahanta et al. [20] have proposed match ratio in somewhat a different way. Authors have proposed match ratio as the number of intervals is divided by the number of years in the lifespan of the pattern for the purpose of mining yearly pattern. It might be difficult to work with this definition, since a mining algorithm returns itemsets and their intervals. A mining algorithm might not be concerned with reporting the first and last appearances of an itemset. Therefore, we will follow the definition proposed by Li et al [18].

We have discussed the concept of certainty factor in Section 1. Also we have noticed that the analysis of overlapped region using certainty factor might not be sufficient. Therefore, we propose an extension to it.

13.3.1 Extending Certainty Factor

The concept of certainty factor is based on the concept of set superimposition. If we are interested in yearly patterns, during the analysis of superimposed intervals the year component is ignored. We explain here the concept of set superimposition using the following example.

Example 2. Consider the database of Example 1. Itemset $\{a, c\}$ is present in 3 out of 4 transactions in the intervals [29/03/1990 - 25/04/1990]. Also, $\{a, c\}$ is present in 2 out of 3 transactions in the intervals and [06/03/1991 - 19/04/1991]. Therefore, $\{a, c\}$ is frequent in these intervals at minimum support level 0.66. These two intervals are being superimposed where each of these intervals has fuzzy set membership value 1/2. The overlapped area of these two intervals is [29/03 - 19/04]. Based on the concept of set superimposition, an itemset reported in a non-overlapped region has the fuzzy set membership value 1/2. But, an itemset reported in the overlapped interval [29/03 - 19/04] has fuzzy set membership value 1/2 + 1/2 = 1. •

For the purpose of mining periodic patterns, Mahanta et al. [20] have proposed certainty factor. It is based on a set of overlapped intervals corresponding to a pattern occurring on a periodic basis. For example, one might be interested in identifying yearly periodic patterns in a database. Authors have considered all the intervals having equi-fuzzy membership value. For example, if n intervals are superimposed then every interval has $1/n$ equi-fuzzy membership value and in an overlapped area the membership value will be added. The certainty of the pattern in the overlapped interval is more than the certainty in the other intervals.

Let $[t_1, t'_1]$ and $[t_2, t'_2]$ be two overlapped intervals where a pattern X gets reported with certainty value 1/2. When the two intervals are superimposed the certainty factors of X associated with the various subintervals are given as follows:

$$[t_1, t'_1]^{1/2} (S) [t_2, t'_2]^{1/2} = [t_1, t_2)^{1/2} [t_2, t'_1]^{1} (t'_1, t'_2]^{1/2} \qquad \ldots(1)$$

The notion of certainty factor seems to be an important contribution made by the authors. It represents the certainty of reporting a pattern in an interval by considering a sequence of periods. For example, we might be interested in knowing the certainty of pattern $\{a, c\}$ in the month of April with respect to the database in Example 1. It is an important statistical evidence of a pattern in an interval over a sequence of years (periods). For example, one could say that the evidence of the pattern $\{a, c\}$ is certain in the month of April when the years viz., 1990, 1991, 1992 and 1993 are considered. But the concept of certainty factor does not convey the information regarding the frequency of a pattern in an overlapped region. In addition, it gives equal importance to all the intervals by considering them as equi-fuzzy intervals. From the perspective of the evidence of a pattern, such assumption might be realistic. But from the perspective of the depth of evidence, such concept might not be sufficient. Thus, we propose an extension to the concept of certainty factor. In the proposed extension, we incorporate the information regarding support of a pattern in an interval. There are many ways one could keep the information regarding support. In Example 1, there are four overlapping intervals corresponding to the pattern $\{a, c\}$. There exists a region where all the intervals are overlapped, while some regions may not be overlapped at all. Apart from the certainty factor of a region, one could also keep the support information of the pattern in that interval. In general, a region could be overlapped by all intervals. Let there be n supports of a pattern corresponding to n intervals. Then the question comes to our mind, how to keep the support information of the pattern for n intervals. The answer to this question might not be agreeable to all. One might be interested in keeping the average support of the pattern along with the certainty factor for that interval. Some of us might be interested in keeping information regarding the minimum and maximum of n supports. In an extreme case, one might be interested in keeping all the n supports of the pattern corresponding to n intervals. Let us consider that we are interested in yearly pattern. Let the lifespan of a pattern be forty years. Then one has to keep a maximum of forty supports corresponding to an overlapped region. It might not be realistic to maintain all the forty supports. Let *s-info*(X, $[t_1, t_2]$) be the support information of the pattern X for the interval $[t_1, t_2]$.

Let a pattern X be frequent in time intervals $[t_i, t'_i]$, $i = 1, 2, \ldots, n$. Each of these intervals is taken from a different period of time such that $\bigcap_{i=1}^{n} [t_i, t_i^{'}] \neq \varphi$. In Example 1, patterns $\{a\}$, $\{c\}$ and $\{a, c\}$ get reported in the month of April in every year. By generalizing (1), the certainty factor of X in overlapped regions could be obtained as follows:

$$[t_1, t'_1]^{1/n} (S) [t_2, t'_2]^{1/n} (S) \ldots (S) [t_n, t'_n]^{1/n} = [t^{(1)}, t^{(2)})^{1/n} [t^{(2)}, t^{(3)})^{2/n} [t^{(3)}, t^{(4)})^{3/n} \ldots$$
$$[t^{(r)}, t^{(r+1)})^{r/n} \ldots \times [t^{(n)}, t^{'(1)}]^{1} (t^{'(1)}, t^{'(2)}]^{n-1/n} \ldots (t^{'(n-2)}, t^{'(n-1)}]^{2/n} (t^{'(n-1)}, t^{'(n)}]^{1/n} \qquad \ldots(2)$$

where $\{t^{(i)}\}_{i=1}^{n}$ is the sequence obtained from $\{t_i\}_{i=1}^{n}$ by sorting in ascending order and $\{t'^{(i)}\}_{i=1}^{n}$ is obtained from $\{t'_i\}_{i=1}^{n}$ by sorting in ascending order. We propose an extended certainty factor of X in the above overlapped intervals as follows:

When X is reported in $[t^{(n)}, t'^{(1)}]$ then the certainty value is 1 with support information *s-info*(X, $[t^{(n)}, t'^{(1)}]$). But, the certainty value of X for the outside of $[t^{(1)}, t'^{(n)}]$ is 0 with support information 0. When X is reported in $[t^{(r-1)}, t^{(r)})$, then the certainty value is $(r - 1) / n$ with support information *s-info*(X, $[t^{(r-1)}, t^{(r)})$), for $r = 2, 3, \ldots, n$. Otherwise, the certainty value of X for $(t'^{(r-1)}, t'^{(r)}]$ is $(n - r + 1) / n$ with support information *s-info*(X, $(t'^{(r-1)}, t'^{(r)}]$), for $r = 3, 4, \ldots, n$.

Suppose we are interested in identifying yearly periodic patters. So each time interval is taken from a year. From the perspective of n years, the pattern X gets reported in every year in the interval $[t^{(n)}, t'^{(1)}]$. So, the certainty of X is 1 (highest) in this interval. But, X is not frequent pattern outside of $[t^{(1)}, t'^{(n)}]$. Therefore, from the perspective of all the years the certainty of X is 0 (lowest) outside of the interval. The certainty factor also provides the information regarding how many intervals are overlapped on a sub-interval. For example, if the certainty factor of a sub-interval is 2/5, for given five intervals, then two intervals are overlapped on the sub-interval. On the other hand, *s-info* provides the information regarding degree of frequency of X in an interval. To illustrate the above concept we consider the following example.

Example 3. The purpose of this example is to explain the proposed concept of extended certainty factor stated above. Let the years 1980, 1981, 1982 and 1983 be of our interest. We would like to check whether the pattern X is yearly periodic. Assume that the mining algorithm has reported X as frequent in the time intervals $[t_1, t'_1]$, $[t_2, t'_2]$, $[t_3, t'_3]$ and $[t_4, t'_4]$ for the years 1980, 1981, 1982 and 1983, respectively. Also, let the supports of X in $[t_1, t'_1]$, $[t_2, t'_2]$, $[t_3, t'_3]$ and $[t_4, t'_4]$ be 0.2, 0.15, 0.16 and 0.12, respectively. Based on the proposed extended concept, we wish to analyze the time interval $[t_1, t'_4]$ by overlapping these intervals corresponding to the four years. The overlapped intervals are depicted in Figure 1.

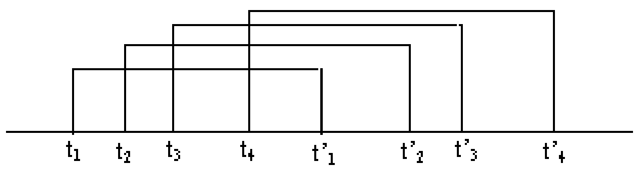


Fig. 1 Overlapped intervals for finding yearly pattern X

While computing support information we use here the range measure for a set of values. One could use another support information depending on the requirement. An analysis of the overlapped intervals corresponding to X is presented in Table 2. Certainty of a sub-interval is based on the number of intervals overlapped on it. For example, $[t_1, t_2)$ has certainty 1/4, since there is only one interval out of four intervals.

Table 2 An analysis of the overlapped intervals for finding yearly pattern X

interval	certainty factor	*s-info*	interval	certainty factor	*s-info*
$[t_1, t_2)$	1/4	0.2 - 0.2	$(t'_1, t'_2]$	3/4	0.12 - 0.15
$[t_2, t_3)$	1/2	0.15 - 0.2	$(t'_2, t'_3]$	1/2	0.12 - 0.16
$[t_3, t_4)$	3/4	0.15 - 0.2	$(t'_3, t'_4]$	1/4	0.12 - 0.12
$[t_4, t'_1]$	1	0.12 - 0.2			

Here *s-info* corresponding to interval $[t_3, t_4)$ represents the fact that the maximum and minimum supports of overlapped intervals are 0.2 and 0.15 respectively. •

Certainty factor and support information are not the same. They represent two different aspects of a pattern in an interval. Certainty factor is normally associated with multiple time intervals. It expresses the likelihood of reporting a pattern in a sub-interval of the multiple overlapped intervals. But the concept of support is associated with a single time-interval. It is defined as the fraction of the transactions containing the pattern in a time-interval. Thus, for an effective analysis of a superimposed interval both the certainty factor and support information are needed in association with an interval.

13.3.2 Extending Certainty Factor with Respect to Other Intervals

In Figure 1 we have shown four intervals overlapped corresponding to four different years. But in reality the scenario could be different. For four intervals, there may exist different combinations of overlapped intervals. But, whatever may be the case, the certainty factor of a sub-interval depends on the number of intervals overlapped in that sub-interval and *s-info* depends on the supports of the pattern in the intervals that are being overlapped on a sub-interval. Let us consider a sub-interval $[t, t']$, where m out of n intervals are overlapped on $[t, t']$. Based on certainty factor [20], we propose an extended certainty factor as follows:

When X is reported in $[t, t']$, then the certainty value is m/n with support information *s-info*$(X, [t, t'])$, where *s-info*$(X, [t, t'])$ is based on supports of X in the m intervals overlapped on $[t, t']$. We illustrate this issue with the help of Example 4. Before that, we present a few definitions related to overlapped intervals. Let *maxgap* be the user-defined maximum gap (time units) between current time-stamp of a pattern and the time-stamp of the pattern when it was last seen. If the gap between current time-stamp of a pattern and the time-stamp of the pattern when it was last seen is greater than *maxgap* then a new interval is formed for the pattern with the current time-stamp as the start of the interval. Also, the previous interval of the pattern was ended when it was last seen. Let *mininterval* be the minimum period length of a time interval. Each interval should be of sufficient length, otherwise a pattern appearing once in a transaction also becomes frequent in an interval. If two intervals are overlapped and the length of the overlapped region exceeds *mininterval* then the overlapped region could be interesting.

Example 4. We refer to the database of Example 1. Let the value of *maxgap* be 40 days. Then pattern $\{a, c\}$ gets reported in the following intervals: [29/03/1990 - 25/04/1990], [06/03/1991 - 12/03/1991], [03/03/1992 - 07/04/1992], [04/03/1993 - 17/04/1993], and [06/04/1994 - 20/04/1994]. Let the value of *mininterval* be 10 days. The interval [06/03/1991 - 12/03/1991] does not satisfy the criterion of *mininterval.* Also let the value of *minsupp* be 0.5. Then $\{a, c\}$ is not locally frequent in the interval [06/04/1994 - 20/04/1994]. We shall analyse the pattern $\{a, c\}$ in the following intervals: [29/03/1990 - 25/04/1990], [03/03/1992 - 07/04/1992], and [04/03/1993 - 17/04/1993]. After superimposition, we require to analyse the interval [03/03 - 25/04]. We present superimposed intervals in Figure 2.

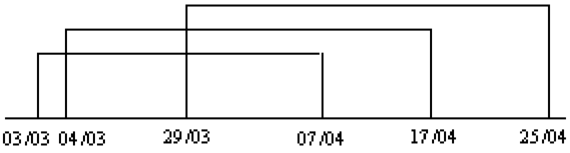


Fig. 2 Overlapped intervals for finding yearly pattern $\{a, c\}$

We present an analysis of the time interval [03/03 - 25/04] based on the concept of extended certainty factor. Extended certainty factor of a pattern in an interval provides information of both the certainty factor and *s-info* for a pattern. In Table 3 we present an analysis of intervals for finding yearly pattern $\{a, c\}$. •

Table 3 An analysis of the time interval [03/03 - 25/04] for finding yearly pattern $\{a, c\}$

interval	certainty	*s-info*	interval	certainty	*s-info*
[03/03 - 04/03)	1/5	1.0 - 1.0	(07/04 - 17/04]	2/5	0.6 - 0.75
[04/03 - 29/03)	2/5	0.6 - 0.75	(17/04 - 25/04]	1/5	0.75 - 0.75
[29/03 - 07/04]	3/5	0.6 - 1.0			

In the above we have presented an analysis of the time interval [03/03 - 25/04]. The subintervals [03/03 - 04/03) and (17/04 - 25/04] are also shown, but they do not satisfy *mininterval* criterion. In the experimental results we have not presented such subintervals.

13.4 Mining Calendar-Based Periodic Patterns

Itemsets in transactions could be considered as a basic type of pattern in a database. Many interesting patterns such as association rules [3], negative association rules [24], Boolean expressions induced by itemset [1] and conditional patterns [2] are based on itemset patterns. Some itemsets are frequent in certain time intervals but may not be frequent throughout the lifespan of the itemsets. In other words, some itemsets may appear in the transactions for a certain time period and then disappear for a long period and then reappear. In view of making a data analysis involving various itemsets, it might be required to extract the itemsets together with the associated time-slots.

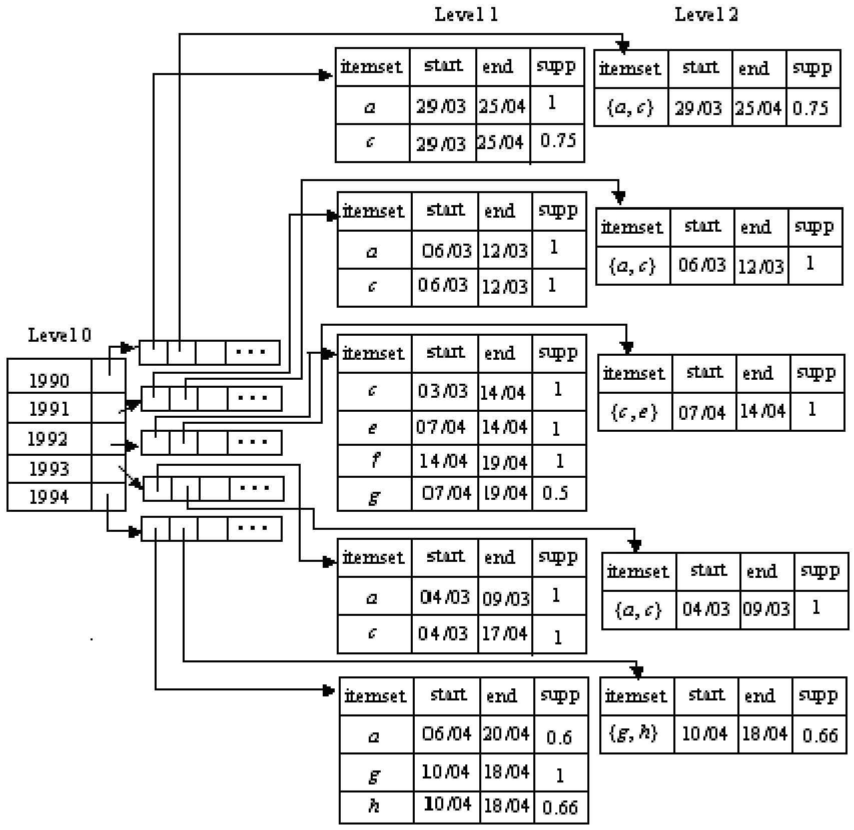
13.4.1 Improving Mining Calendar-Based Periodic Patterns

The goal of this paper is to study the existing algorithm, and to propose an effective algorithm by improving the limitations of the existing algorithm for mining calendar-based periodic patterns. As noted earlier that the concept of certainty factor of an interval does not provide good analysis of overlapped intervals. Therefore, the concept of extended certainty factor has been proposed. In view of designing an effective algorithm, we also need to understand the existing algorithm. Mahanta et al. [19] have proposed an algorithm for finding all the locally frequent itemsets of size one. While studying the algorithm we have found that some variables contradict their definitions. Authors defined two variables *ptcount* and *ctcount* as follows. The variable *ptcount* is used to count the number of transactions in an interval in which the current item belongs. On the other hand, the variable *ctcount* is used to count the number of transactions in that interval. Therefore, the assignment $ptcount[k] = ctcount[k]$ in the algorithm, seems to be not appropriate. Also, the variable *icount* is defined as the number of items present in the whole dataset. Therefore, the initialization, $icount = 1$, placed just before starting a new interval seems to be inappropriate. Moreover, the validity of the experimental results is low, since it is based on only one dataset. In view of improving the algorithm further we propose a number of modifications mentioned as follows: (i) The proposed algorithm makes corrections on the existing algorithm using the points noted above. (ii) It makes effective data analysis by incorporating extended certainty factor. (iii) We propose a hash-based data structure to improve the space efficiency of our algorithm. (iv) Also, we have improved the average time complexity of the algorithm. (v) We make a comparative analysis with the existing algorithm. (vi) In addition, we have improved the validity of the experimental results by conducting experiments on more datasets.

13.4.2 Data Structure

We discuss here the data structure used in the proposed algorithms for mining itemsets along with the time intervals in which they are frequent. We describe the data structure using Example 5 given below.

Example 5. Consider the database *D* of Example 1. Transactions consisting of items *a*, *b*, *c*, *d*, *e*, *f*, *g*, *h*, and *i* occurred in the years from 1990 to 1994. We propose Algorithm 1 to mine locally frequent itemsets of size one along with their intervals. The algorithm produces output as shown at level 1 of Figure 3. We assume here *maxgap*, *mininterval* and *minsupp* as 40 days, 5 days and 0.5, respectively. We are interested in identifying yearly periodic patterns. At the level 0 we have shown all the years that appeared in the transaction. The pointer corresponding to the year 1990 keeps all the locally frequent itemsets of size one, their supports and intervals. All the five years are stored in an index table at level 0. After level 0, we keep an array of pointers for every year. The first pointer corresponding to year 1990 points to a table containing interesting itemsets of size one, their intervals and local supports. The second pointer corresponding to year 1990, points to a table containing interesting itemsets of size two, their intervals and local supports, and so on. Here itemsets of size three corresponding to a year do not get reported. Different itemsets, their intervals, and supports are shown in Figure 3.



itemset	start	end	supp
a	29/03	25/04	1
c	29/03	25/04	0.75

itemset	start	end	supp
{*a*, *c*}	29/03	25/04	0.75

itemset	start	end	supp
a	06/03	12/03	1
c	06/03	12/03	1

itemset	start	end	supp
{*a*, *c*}	06/03	12/03	1

itemset	start	end	supp
c	03/03	14/04	1
e	07/04	14/04	1
f	14/04	19/04	1
g	07/04	19/04	0.5

itemset	start	end	supp
{*c*, *e*}	07/04	14/04	1

itemset	start	end	supp
a	04/03	09/03	1
c	04/03	17/04	1

itemset	start	end	supp
{*a*, *c*}	04/03	09/03	1

itemset	start	end	supp
a	06/04	20/04	0.6
g	10/04	18/04	1
h	10/04	18/04	0.66

itemset	start	end	supp
{*g*, *h*}	10/04	18/04	0.66

Fig. 3 Data structure used in the proposed algorithms

13.4.3 A Modified Algorithm

As mentioned in Section 4.1, we have proposed a number of improvements to the algorithm proposed by Mahanta et al. [19] for finding locally frequent itemsets of size one. We calculate the support of each item in an interval and store it whenever the item is frequent in that interval. Intervals that satisfy the user-defined constraints *mininterval* and *minsupp* are retained. The modification made seems to be significant from the overall viewpoint of apriori algorithm. We have used a hash-based data structure to improve efficiency of storing and accessing locally frequent itemsets of size one. We explain all the variables and their functions in the following paragraph.

Let *item* be an array of items in *D*. Also let the total number of items be *n*. We use index *level_0* to keep track of different years. It is a two-dimensional array containing 2 columns. First column of *level_0* contains the different years in increasing order. A two-dimensional array *itemset_addr* is used to store the addresses of tables containing itemsets. *itemset_addr*[*row*][*j*] contains the address of the table containing locally frequent itemsets of size *j* for the year *row*. The second column of *level_0* stores addresses of arrays pointing to these tables. Tables at *level_p* store the frequent itemsets of size *p*, $p = 1, 2, 3, \ldots$. Variables *row* and *row_p* are used to index arrays *itemset_addr* and *level_p* respectively, $p = 0, 1, 2, \ldots$. We consider a transaction as a record containing transaction date (*date*) and items purchased. Function *year*( ) is used to extract year from a given date. *firstseen*[*k*] and *lastseen*[*k*] specify the date when the *k*-th item is seen for the first time and last time in an interval, respectively. Each item in the database is associated with the arrays *itemIntervalFreq* and *nTransInterval*. Cells *itemIntervalFreq*[*k*] and *nTransInterval*[*k*] are used to keep the number of transactions containing item *k* and total number of transactions in a time interval, respectively. Variable *nItemsTrans* is used to keep track of the number of items in the current transaction. The goal of the Algorithm 1 is to find all the locally frequent itemsets of size one, their intervals and supports. The algorithm is presented as follows.

Algorithm 1. Mine locally frequent items and their intervals
procedure *MiningFrequentItems* (*D*, *maxgap*, *mininterval, minsupp*)
Inputs: *D, maxgap, mininterval, minsupp*
D: database to be mined
minsupp: as defined in Section 1
maxgap, mininterval: as defined in Section 3.2
Outputs:
Locally frequent items, their intervals and supports as mentioned in Figure 3

```
01: let nItemsTrans = 0; row = 1; row_0 =1; row_1 = 1;
02: for $k = 1$ to $n$ do
03:     lastseen[k] = 0; itemIntervalFreq[k] = 0; nTransInterval[k] = 0;
04: end for
05: read a transaction $t \in D$;
```

```
06: level_0[row_0][1] = year(t.date);
07: level_0[row_0][2] = itemset_addr[row][1];
08: while not end of transaction in D do
09:   transLength = |t|;
10:   if (level_0[row_0][1] $\neq$ year(t.date)) then
11:     for k = 1 to n do
12:       if (|lastseen[k] $-$ firstseen[k]| $\geq$ mininterval) and
          (itemIntervalFreq[k] / nTransInterval[k] $\geq$ minsupp) then
13:       store the k-th item, its firstseen, lastseen and local support at level_1[row_1];
14:       increase row_1 by 1;
15:       end if {12}
16:   end for {11}
17:   row_1 = 1;
18:   increase row_0 by 1; increase row by 1;
19:   level_0[row_0][1] = year(t.date); level_0[row_0][2] = itemset_addr[row][1];
20:   for k = 1 to n do
21:     lastseen[k] = 0; itemIntervalFreq[k] = 0; nTransInterval[k] = 0;
22:     end for
23:   end if {10}
24:   for k = 1 to n do
25:     if (item[k] $\in$ t) then
26:       increase nItemsTrans by 1;
27:       if (lastseen[k] = 0) then
28:         initialize both lastseen[k] and firstseen[k] by t.date;
          initialize both itemIntervalFreq[k] and nTransInterval[k] by 1;
29:       else if ( | t.date $-$ lastseen[k] | $\leq$ maxgap ) then
30:           lastseen[k] = t.date;
31:           increase itemIntervalFreq[k] by 1; increase nTransInterval[k] by 1;
32:           end if
33:       else if ( | lastseen[k] $-$ firstseen[k] | $\geq$ mininterval) and
      (itemIntervalFreq[k] / nTransInterval[k] $\geq$ minsupp) then
34:   store the k-th item, its firstseen, lastseen and local support at level_1[row_1];
35:          increase row_1 by 1;
36:          initialize both lastseen[k] and firstseen[k] by t.date;
37:          initialize both itemIntervalFreq[k] and nTransInterval[k] by 0;
38:          end if {33}
39:       end if {27}
40:     else increase nTransInterval[k] by 1;
41:     end if {25}
42:     if (nItemsTrans = transLength) then exit from for-loop; end if
43:   end for {24}
44:   read a transaction t $\in$ D;
45: end while {08}
46: for k = 1 to n do
```

```
47:   if ($|lastseen[k] - firstseen[k]| \geq mininterval$) and
      ($itemIntervalFreq[k]$ / $nTransInterval[k] \geq minsupp$) then
48:     store the k-th item, its firstseen, lastseen and local support at level_1[row_1];
49:     increase row_1 by 1;
50:   end if {47}
51: end for {46}
52: sort arrays level_1 on non-increasing order on primary key item and
    secondary key start date;
end procedure
```

At line 5 we read the first transaction of database. Afterwards the first row of the index *level_0* is initialized with the first year obtained from the transaction. The pointer field of the first row of *level_0* is initialized by the address of the first row of the table *itemset_addr*. Lines 8-45 are repeated until all the transactions are read. At line 10 we check whether the current transaction belongs to a different year. If it happens so then we close the last interval of different items using lines 11-16. We retain those intervals that satisfy criteria of *mininterval* and *minsupp*. Lines 17-22 assign the necessary initializations for a different year. Lines 25-41 are repeated for each item in the current transaction. Line 27 checks whether the item is first time seen in the transaction and the necessary assignment is done in line 28. Lines 29-32 determine whether the current transaction-date is coming under the current interval by comparing the difference between *t.date* and *lastseen* with *maxgap*. Lines 33-38 construct an interval and compute the local support. Line 42 avoids the unnecessary repetition by comparing the transaction length. Line numbers 46-51 close all the last intervals for last year. Line 52 sorts arrays *level_1* on non-increasing order on primary key item and secondary key start date.

The time complexity of the algorithm has been reduced significantly by computing the length of current transaction (at line number 9) and putting a check at line number 25. Consider a database containing 10,000 items. Let the current transaction be of length 20 and these items are within the first 100 items. Then the *for-loop* at line number 24 need not have to continue for the remaining 9,900 items, but the worst-case complexity of the algorithm remains the same as before.

We shall now present below an algorithm that makes use of locally frequent itemsets obtained by Algorithm 1 and apriori property [4]. We use array *level_1* to generate the candidate sets at the second level. Then array *level_2* is used to generate candidate sets at the third level, and so on. We apply pruning using conditions at line number 6 to eliminate some itemsets at the next level. This pruning step ensures that the size of the itemsets at the current level is one more than the size of an itemset at the previous level. Also we apply pruning using user-defined thresholds such as *maxgap*, *mininterval* and *minsupp*.

Algorithm 2. Mine locally frequent itemsets at higher level and the associated intervals

```
procedure MiningHigherLevelItemsets (D, S)
Inputs: D, S
D: database to be mined
S: partially constructed data structure containing locally frequent itemsets of size one
Outputs: locally frequent itemsets at higher levels, the associated intervals and supports as mentioned in Figure 3
01: let $L_1$ = set of elements at level_1 of S; let k = 2;
02: while $L_{k-1} \neq \phi$ do
03:    $C_k = \phi$;
04:    for each itemset $l_1 \in L_{k-1}$ do
05:       for each itemset $l_2 \in L_{k-1}$ do
06:          if $((l_1[1] = l_2[1]) \wedge \ldots \wedge (l_1[k-2] = l_2[k-2]) \wedge (l_1[k-1] < l_2[k-1]))$ then
07:             $c = l_1 \bowtie l_2$; $C_k = C_k \bigcup c$;
08:          end if {06}
09:       end for {05}
10:    end for {04}
11:    for each element $c \in C_k$ do
12:       construct intervals for c as mentioned in Algorithm 1;
13:       if the intervals corresponding to c satisfy maxgap, mininterval and minsupp then
14:          add c and the intervals to level_k of S;
15:       end if {13}
16:    end for {11}
17:    increase k by 1;
18:    let $L_k$ = set of elements at level_k of S;
19: end while {02}
end procedure
```

Using Algorithms 1 and 2, one could construct the data structure *S* presented in Figure 3 completely. One can use *S* to determine whether an itemset pattern is fully / partially periodic.

13.5 Experimental Studies

We have carried out several experiments for mining calendar-based periodic patterns on different databases. All the experiments have been implemented on a 2.4 GHz, core i3 processor with 4 GB of memory, running Windows 7 HB, using Visual C++ (version 6.0) software. We present experimental results using *retail* [10], *BMS-WebView-1* [10], and *T10I4D100K* [10] databases. Since the records in these databases contain only items purchased in transactions, we have attached time-stamps randomly as calendar date for the transactions. The characteristics of the databases are given in Table 4.

Table 4 Database characteristics

D	*NT*	*ALT*	*AFI*	*NI*	Size in Megabytes
retail	88,162	11.31	60.54	16,470	3.97
BMS-WebView-1	1,49,639	2.00	44.57	6,714	1.97
T10I4D100K	1,00,000	11.10	1276.12	870	3.83

Each of the databases *retail*, *BMS-WebView-1* and *T10I4D100K* has been divided into 30 sub-databases, called yearly databases, for the purpose of conducting experiments. The characteristics of these databases are given in Table 5. Let *D*, *NT*, *ALT, AFI,* and *NI* be the given database, the number of transactions, average length of a transaction, average frequency of an item, and the number of items, respectively. In Table 5 we have shown how the transactions have been time-stamped. The yearly databases obtained from *retail*, *BMS-WebView-1* and *T10I4D100K* are named as R_i, B_i and T_i respectively, $i = 1, \ldots, 30$. For simplicity, we have kept the number of transactions in each of the yearly databases fixed, except for the last database. We assume that the first and the last transactions occur on 01/01/1961 and 31/12/1990 respectively, and also assume that each year contains 365 days. In our experimental studies we report yearly periodic patterns and the associated periodicities. We also compute certainty factor and match ratio of a pattern with respect to overlapped intervals.

Table 5 Characteristics of yearly databases

D	*NT*	starting date, ending date	average number of transactions per day
R_1	2920	01/01/1961, 31/12/1961	8
…	…	…	…
R_{29}	2920	01/01/1989, 31/12/1989	8
R_{30}	3482	01/01/1990, 31/12/1990	9.54
B_1	5110	01/01/1961, 31/12/1961	14
…	…	…	…
B_{29}	5110	01/01/1989, 31/12/1989	14
B_{30}	1449	01/01/1990, 31/12/1990	3.97
T_1	3285	01/01/1961, 31/12/1961	9
…	…	…	…
T_{29}	3285	01/01/1989, 31/12/1989	9
T_{30}	4735	01/01/1990, 31/12/1990	12.97

In addition to partial periodic patterns, we mine full periodic patterns in the above databases. Itemset patterns of size one and two of *retail* is shown in Tables 6 and 7 respectively. In *retail* the itemsets {39} and {48} occur in all the thirty years and they are periodic throughout the year. Therefore, these itemsets are full periodic in the interval [1/1-31/12]. Itemset {41} is partially periodic, since the match ratio is less than 1. Initially it becomes frequent for thirteen years and then it does not get reported, and again it becomes frequent for the last six years. The subintervals that do not satisfy the *mininterval* criterion are not shown. We have noticed some peculiarity in the mined patterns. For example, many patterns such as {0} and {1} are frequent throughout a year. Although, it is peculiar but it remains also an artificial phenomenon, since the time-stamps are enforced by us. There are many itemsets such as {16217} are frequent in many years with non-overlapping intervals. In Table 6, we present itemsets of size one that are also part of interesting itemsets of size two as shown in Table 7. While computing the certainty factor of an itemset we have used lifespan of the itemset. For example, itemset {0} gets reported from two years and it becomes frequent in both the years. Therefore its certainty factor is 2/2 = 1.

Table 6 Selected yearly periodic itemsets of size one (for *retail*)

retail (*minsupp* = 0.25, *mininterval* = 8, *maxgap* = 10)				
itemset	intervals	certainty	*s-info*	match ratio
{0}	[1/1-31/12]	2/2	0.35-0.66	1.0
{1}	[3/1-31/12]	2/3	0.57-0.66	0.67
{39}	[1/1-31/12]	30/30	0.52-0.63	1.0
{41}	[1/1-22/12]	13/30	0.26-0.32	0.43
{41}	[2/12-30/12]	6/30	0.27-0.32	0.20
{48}	[1/1-31/12]	30/30	0.43-0.53	1.0
{16217}	[1/1- 30/5]	1/1	0.87-0.87	1.0
{16217}	[7/9- 31/12]	1/1	0.97-0.97	1.0

Table 7 Yearly periodic itemsets of size two (for *retail*)

retail (*minsupp* = 0.25, *mininterval* = 8, *maxgap* = 10)				
itemset	intervals	certainty	*s-info*	match ratio
{0, 1}	[15/10-31/12]	1/1	0.46	1.0
{39, 41}	[1/1-30/12]	1/1	0.25	1.0
{39, 48}	[1/1-30/12]	30/30	0.28-0.38	1.0
{39, 16217}	[1/1- 30/5]	1/1	0.34	1.0
{48, 16217}	[7/9- 31/12]	1/1	0.27	1.0

Interesting itemset patterns of size one and two in *BMS-WebView-1* are shown in Tables 8 and 9 respectively. Here full periodic patterns are not reported since all the itemsets in *BMS-WebView-1* have match ratio less than 1. Therefore, these patterns are partial periodic. Itemset {12355} becomes frequent in three years but it has lifespan for seven years. In this database the items are sparse. Therefore, one requires choosing a smaller *minsupp*. From Table 9 one could observe that itemset

{33449, 33469} shows periodicity by appearing two times in six years and the remaining interesting itemsets are reported for a year only.

Table 8 Yearly periodic itemsets of size one (for *BMS-WebView-1*)

BMS-WebView-1(*minsupp*=0.06, *mininterval* = 7, *maxgap* = 10)				
itemset	intervals	certaint y	*s-info*	match ratio
{10311}	[29/1-6/10]	2/6	0.063 - 0.86	0.33
{12355}	[21/12-28/12]	3/7	0.060 - 0.061	0.43
{12559}	[22/4-11/5]	1/2	0.064 - 0.066	0.5
{33449}	[3/1-26/12]	5/7	0.063 - 0.08	0.71
{33469}	[3/1-31/3]	5/7	0.067 - 0.08	0.71

Table 9 Yearly periodic itemsets of size two (for *BMS-WebView-1*)

BMS-WebView-1(*minsupp*=0.06, *mininterval* = 7, *maxgap* = 10)				
itemset	intervals	certaint y	*s-info*	match ratio
{10311, 12559}	[30/4-9/4]	1/1	0.06-0.06	1.0
{10311, 33449}	[3/3-11/4]	1/1	0.065	1.0
{33449, 33469}	[15/2-25/3]	2/6	0.061-0.064	0.33

In Table 10 we present yearly periodic itemsets of size one for *T10I4D100K* database. In this database patterns with full periodicity are not available, since the intervals corresponding to an item are not overlapped. We have presented examples of such items in the following table. From interval column, one could observe that the itemsets are frequent for the short intervals, but do not appear at the same time for all the years. For example, itemset {966} appears in three intervals in 1961, but it does not show any periodicity since the intervals are not overlapped. It is interesting to note that the itemset {966} appears at the beginning, both in first and second months, of the year, then at the middle of the year i.e., for the third and fourth months, and finally at the end of the year (eleventh and twelfth month). This is also true for itemset {998}. Interesting itemset patterns of size two are not reported from this database.

An itemset that satisfies *minsupp*, *mininterval* criteria are reported. Also, a locally frequent itemset in two intervals for a particular year is also reported from the intervals, provided the intervals satisfy *maxgap* criterion. The number of interesting intervals could increase by lowering the thresholds. In the following paragraphs we have presented a study on this aspects.

Table 10 Selected yearly periodic itemsets of size one (for *T10I4D100K*)

T10I4D100K (*minsupp*=0.13, *mininterval* = 7, *maxgap* = 10)					
itemset	interval	*s-info*	itemset	interval	*s-info*
{966}	[28/1/1961 - 19/2/1961]	0.16	{998}	[13/11/1964 - 22/11/1964]	0.17
{966}	[16/3/1961 - 23/3/1961]	0.17	{998}	[15/12/1964 - 27/12/1964]	0.16
{966}	[14/12/1961 - 25/12/1961]	0.16	{998}	[2/9/1965 - 13/9/1965]	0.14
{966}	[22/3/1964 - 13/4/1964]	0.15	{998}	[27/11/1966 - 8/12/1966]	0.14
{966}	[1/11/1975 - 8/11/1975]	0.16	{998}	[27/11/1973 - 11/12/1973]	0.13
{966}	[12/4/1981 - 19/4/1981]	0.17	{998}	[14/12/1983 - 23/12/1983]	0.17
{966}	[2/12/1988 - 12/12/1988	0.15	{998}	[15/12/1984 - 23/12/1984]	0.16

13.5.1 Selection of Mininterval and Maxgap

The selection of *mininterval* and *maxgap* might be crucial since the process of data mining would depend on factors like seasonality, type of application and the data source. Some items are used for a particular season; while others are purchased throughout the year. When the items are purchased throughout the year, the choices of *mininterval* and *maxgap* do not have much significance in mining yearly patterns. This observation seems to be valid for the items in *retail* and *BMS-WebView-1*. But the items in *T10I4D100K* are frequent in smaller intervals and therefore, *mininterval* and *maxgap* might have an impact on data mining. On the other hand, the requirement of an organization might determine an important parameter for mining calendar-based patterns. The distribution of items in databases also matters in selecting the right values of *mininterval* and *maxgap*. For a sparse database *maxgap* could be longer, and it could be even longer than *mininterval* provided *minsupp* remains small.

13.5.1.1 Mininterval

In the following experiments we would like to analyse the effect of *mininterval* for given *maxgap* and *minsupp*. We observe in Figures 4, 5 and 6, the number of intervals decreases as *mininterval* increases. An itemset might be frequent in many intervals. The number of itemsets frequent in an interval decreases as the length of *mininterval* increases. Although the above observation is true in general, but the type of the graphs might differ from one data source to another. In *retail* many itemsets are locally frequent for longer period of time. In Figure 4 we observe that there exists nearly 110 intervals for *mininterval* of 29 days. Whereas in *BMS-WebView-1* and *T10I4D100K*, the itemsets are frequent for shorter duration. As a result, the number of intervals reduces significantly when *mininterval* remains small. Thus the choice of *mininterval* is an important issue.

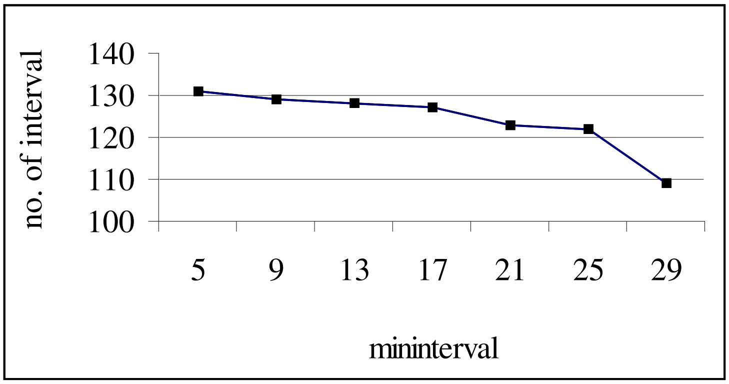


Fig. 4 *Retail* (*minsupp* = 0.25, *maxga*p = 7)

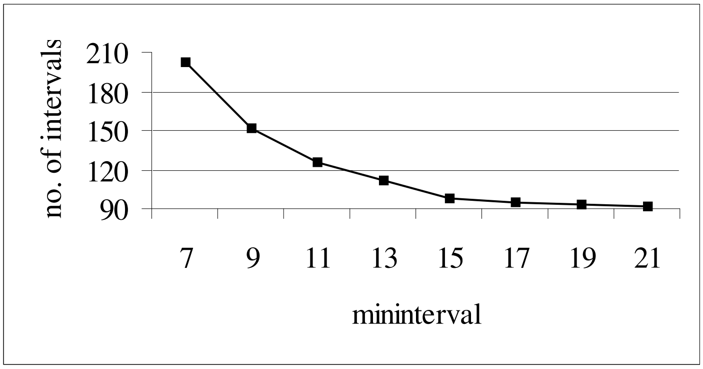


Fig. 5 *BMS-WebView-1* (*minsupp* = 0.06, *maxga*p = 7)

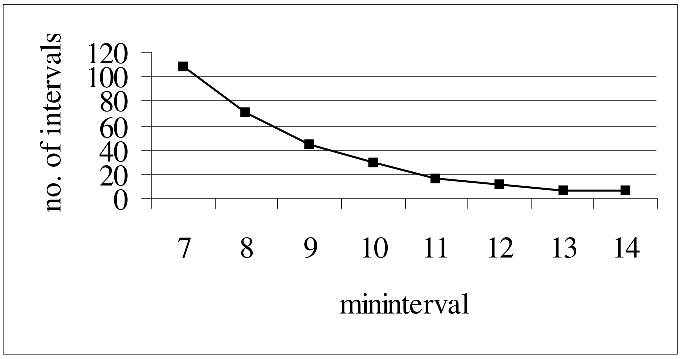


Fig. 6 *T10I4D100K* (*minsupp* = 0.13, *maxga*p = 7)

13.5.1.2 Maxgap

In view of analyzing *maxgap* parameter, we present graphs of the number of intervals versus *maxgap* at given *minsupp* and *mininterval* in Figures 7, 8, and 9. The graphs show that the number of intervals decreases as *maxgap* increases. In *retail* the number of intervals decreases rapidly when *maxgap* varies from 5 to 10. Afterwards the change is not so significant. In *BMS-WebView-1* the decrement takes place almost at a uniform rate. Unlike *retail* and *BMS-WebView-1*, the number of intervals decreases faster at the smaller values of *maxgap* in *T10I4D100K* dataset.

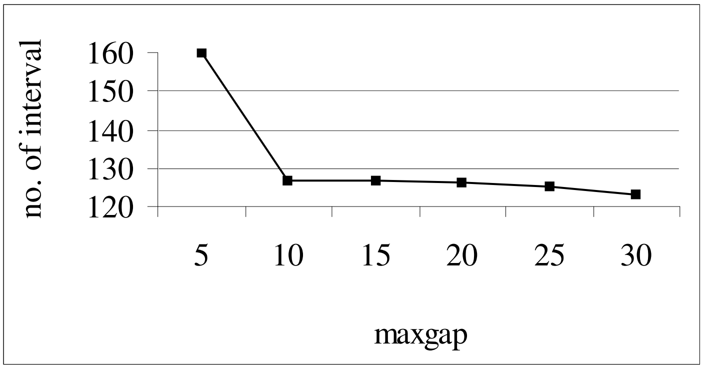


Fig. 7 *Retail* (*minsupp* = 0.25, *mininterval* = 10)

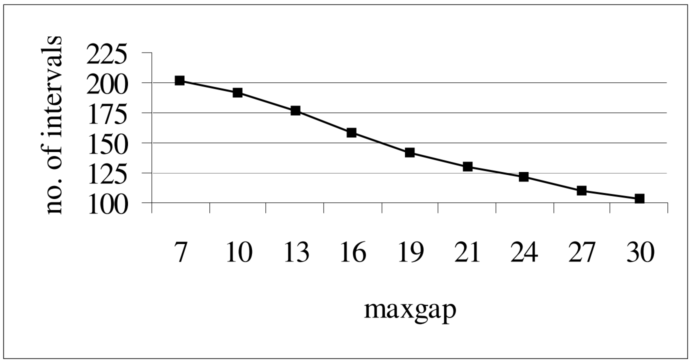


Fig. 8 *BMS-WebView-1*(*minsupp* = 0.06, *mininterval* = 7)

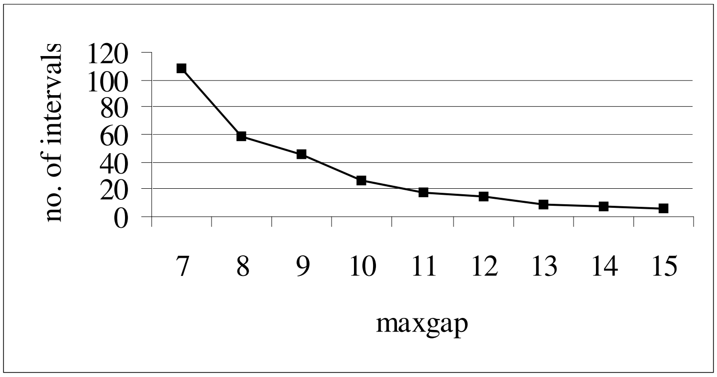


Fig. 9 *T10I4D100K* (*minsupp* = 0.13, *mininterval* = 7)

13.5.2 Selection of Minsupp

The number of intervals and minimum support are inversely related to given *maxgap* and *mininterval*. We observe this phenomenon in Figures 10, 11 and 12. When the value of *maxgap* is smaller the number of intervals reported is quite large. Initially the number of intervals reported significantly with small decrement of *minsupp*. Later the decrement of number of intervals is not so significant.

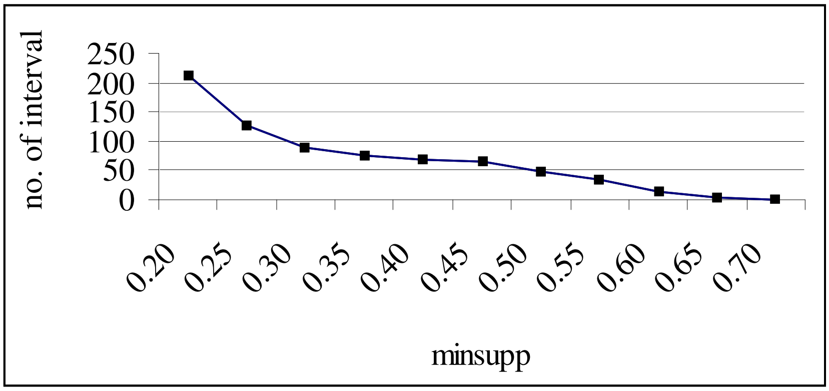


Fig. 10 *Retail* (*mininterval* = 10, *maxgap* = 12)

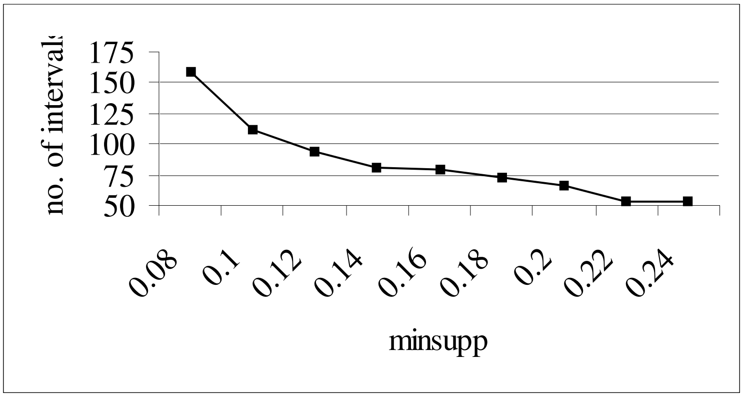


Fig. 11 *BMS-WebView-1*(*mininterval* = 7, *maxga*p = 10)

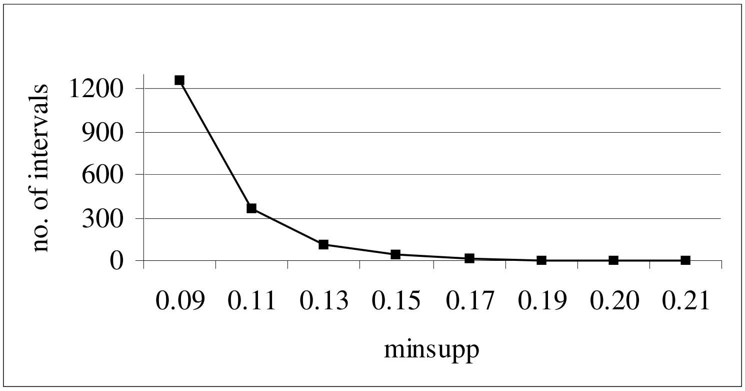


Fig. 12 *T10I4D100K* (*mininterval* = 7, *maxga*p = 9)

13.5.3 Performance Analysis

In this section, we present the performance of our algorithm and compare it with existing algorithm for mining calendar-based periodic patterns. To measure the performance, two experiments have been conducted. In the first experiment, we have measured the scalability of the two algorithms with respect to different database sizes. In the second experiment, we have measured the scalability of the two algorithms with respect to different support thresholds. In Figures 13, 14 and 15, we have shown the relationship between the database size and execution time for mining periodic patterns. We observed that the number of patterns increases as the number of transactions increases. Thus, the execution time increases with the increase of database size. Initially both the algorithms take almost equal amount of time. In Figures 13 and 14 we observe that execution time for mining 88162 transactions of *retail* and 1,49,639 transactions of *BMS-WebView-1* take nearly equal amount of time. The reason is that the average length of transactions in *retail* is more than that of *BMS-WebView-1*. Therefore, the execution time is not only dependent on the size of the database, but also depends on the factors such as *ALT* and *NI*. The experimental results in Figures 13, 14 and 15 show that our algorithm performs better than the existing algorithm.

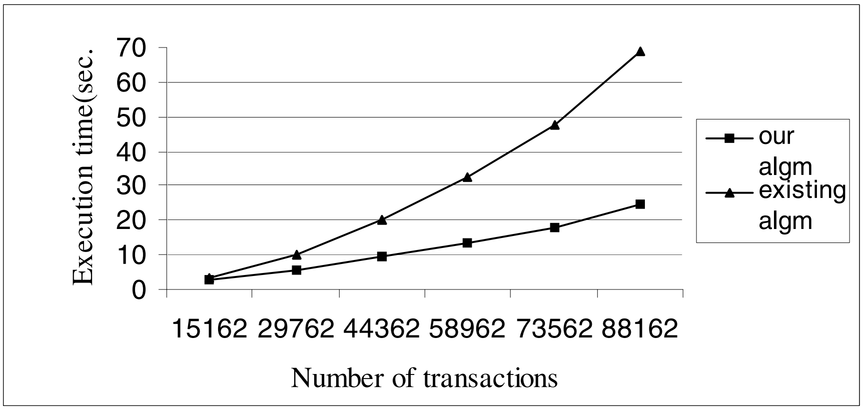


Fig. 13 Execution time vs. size of database at *minsupp* = 0.25, *mininterval* = 8, *maxga*p = 10 (*retail*)

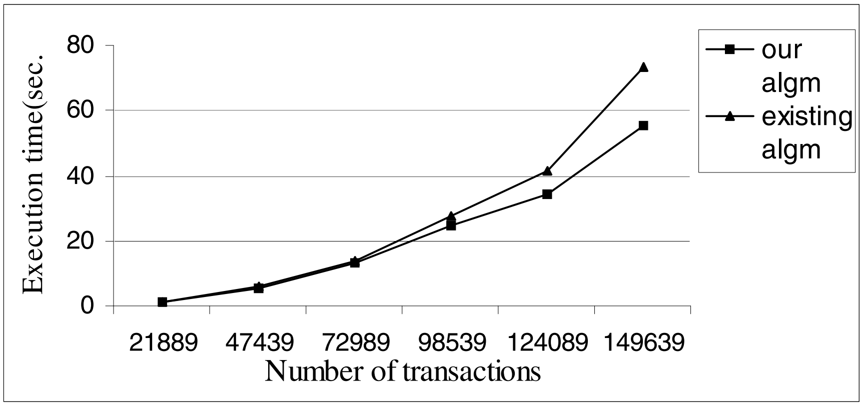


Fig. 14 Execution time vs. size of database at *minsupp* = 0.1, *mininterval* = 7, *maxga*p = 10 (*BMS-WebView-1*)

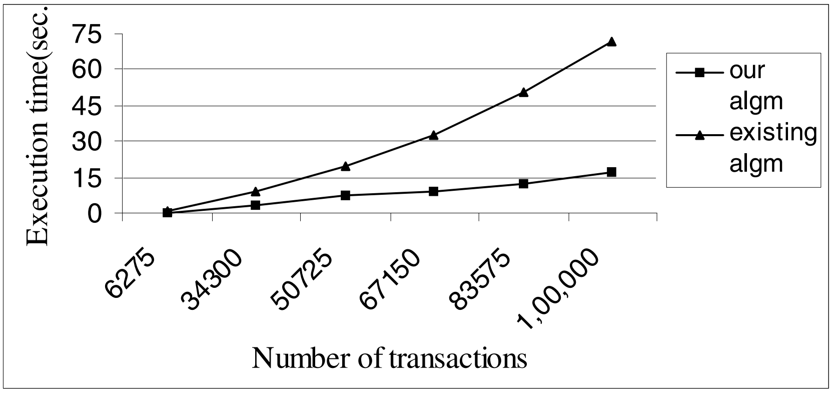


Fig. 15 Execution time vs. size of database at *minsupp* = 0.13, *mininterval* = 7, *maxga*p = 10 (*T10I4D100K*)

In Figures 16, 17, and 18, we have presented another comparison by considering *minsupp* threshold. When the minimum support increases the number of frequent itemsets decreases and so the execution time also decreases. The experimental results have shown that the execution time of both the algorithms decrease slowly when the support threshold increases, and our algorithm takes less time than the existing algorithm.

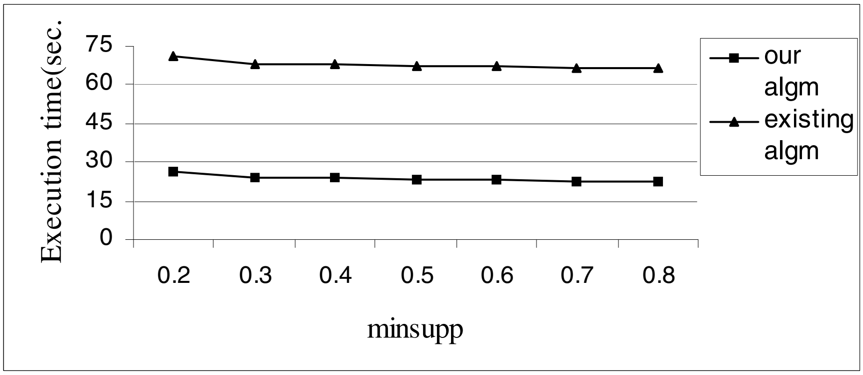


Fig. 16 Execution time vs. *minsupp* (*mininterval* = 8, *maxga*p = 10) for *retail*

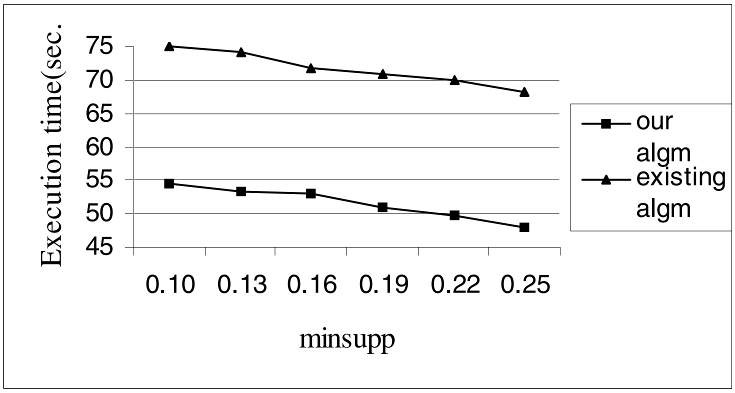


Fig. 17 Execution time vs. *minsuopp* (*mininterval* = 7, *maxga*p = 10) for *BMS-WebView-1*

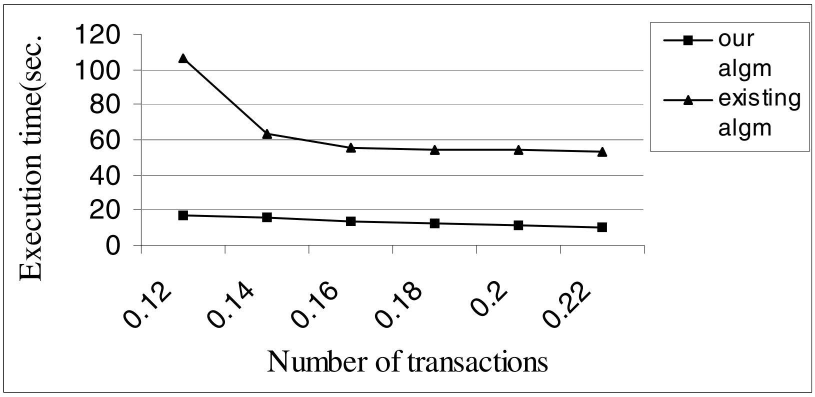


Fig. 18 Execution time vs. *minsupp* (*mininterval* = 7, *maxga*p = 10) for *T10I4D100K*

13.6 Conclusions

In this paper we have proposed modifications to the existing algorithm for mining locally frequent itemsets along with the set of intervals and associated supports. We have also extended the concept of certainty factor for a detailed analysis of overlapped intervals. We have proposed a number of improvements on the existing algorithm for finding calendar-based periodic patterns. For managing locally frequent itemsets effectively we have introduced a hash based data structure. We have presented an extensive data analysis by involving constraints such as *mininterval*, *minsupp* and *maxga*p. In addition we have compared our algorithm with the existing algorithm. Experimental results show that our algorithm executes faster than the existing algorithm. Experimental results also report that whether a periodic pattern is full or partial. The proposed algorithm can also be used to extract yearly, monthly, weekly and daily calendar-based patterns.

Acknowledgement. The first author would like to thank the University Grants Commission (UGC), Government of India, for having sponsored her on faculty improvement programme with leave to take up full time research.

References

1. Adhikari, A., Rao, P.R.: A framework for mining arbitrary Boolean expressions induced by frequent itemsets. In: Proceedings of the International Conference on Artificial Intelligence, pp. 5–23 (2007)
2. Adhikari, A., Rao, P.R.: Mining conditional patterns in a database. Pattern Recognition Letters 29(10), 1515–1523 (2008)
3. Agrawal, R., Imielinski, T., Swami, A.: Mining association rules between sets of items in large databases. In: Proceedings of the ACM SIGMOD Conference Management of Data, pp. 207–216 (1993)
4. Agrawal, R., Srikant, R.: Fast algorithms for mining association rules. In: Proceedings of 20th Very Large databases (VLDB) Conference, pp. 487–499 (1994)
5. Agrawal, R., Srikant, R.: Mining Sequential Patterns. In: Proceedings of International Conference on Data Engineering (ICDE), pp. 3–14 (1995)
6. Ale, J.M., Rossi, G.H.: An approach to discovering temporal association rules. In: Proceedings of ACM Symposium on Applied Computing, pp. 294–300 (2000)
7. Allen, J.F.: Maintaining knowledge about temporal intervals. Communications of the ACM 26(11), 832–843 (1983)
8. Baruah, H.K.: Set superimposition and its application to the theory of fuzzy sets. J. Assam Science Soc. 10(1 and 2), 25–31 (1999)
9. Bettini, C., Jajodia, S., Wang, S.X.: Time Granularities in Databases. In: Data Mining and Temporal Reasoning. Springer (2000)
10. Frequent itemset mining dataset repository, `http://fimi.cs.helsinki.fi/data`
11. Han, J., Dong, G., Yin, Y.: Efficient mining on partial periodic patterns in time series database. In: Proceedings of Fifteenth International Conference on Data Engineering, pp. 106–115 (1999)
12. Hong, T.P., Wu, Y.Y., Wang, S.L.: An effective mining approach for up-to-date patterns. Expert Systems with Applications (36), 9747–9752 (2009)
13. Kempe, S., Hipp, J., Lanquillon, C., Kruse, R.: Mining frequent temporal patterns in interval sequences. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 16(5), 645–661 (2008)
14. Lee, J.W., Lee, Y.J., Kim, H.K., Hwang, B.H., Ryu, K.H.: Discovering temporal relation rules mining from interval data. In: Proceedings of the 1st Euro Asian Conference on Advance in Information and Communication Technology, Iran, (2002)
15. Lee, Y.J., Lee, J.W., Chai, D., Hwang, B., Ryu, K.H.: Mining temporal interval relational rules from temporal data. Journal of Systems and Software 82(1), 155–167 (2009)
16. Lee, G., Yang, W., Lee, J.M.: A parallel algorithm for mining multiple partial periodic patterns. Information Science 176(24), 3591–3609 (2006)
17. Li, D., Deogun, J.S.: Discovering Partial Periodic Sequential Association Rules with Time Lag in Multiple Sequences for Prediction. In: Hacid, M.-S., Murray, N.V., Raś, Z.W., Tsumoto, S. (eds.) ISMIS 2005. LNCS (LNAI), vol. 3488, pp. 332–341. Springer, Heidelberg (2005)

18. Li, Y., Ning, P., Wang, X.S., Jajodia, S.: Discovering calendar-based temporal association rules. Data and Knowledge Engineering 44(2), 193–218 (2003)
19. Mahanta, A.K., Mazarbhuiya, F.A., Baruah, H.K.: Finding Locally and Periodically Frequent Sets and Periodic Association Rules. In: Pal, S.K., Bandyopadhyay, S., Biswas, S. (eds.) PReMI 2005. LNCS, vol. 3776, pp. 576–582. Springer, Heidelberg (2005)
20. Mahanta, A.K., Mazarbhuiya, F.A., Baruah, H.K.: Finding calendar-based periodic patterns. Pattern Recognition Letters 29(9), 1274–1284 (2008)
21. Ozden, B., Ramaswamy, S., Silberschatz, A.: Cyclic association rules. In: Proceedings of 14th International Conference on Data Engineering, pp. 412–421 (1998)
22. Roddick, J.F., Spiliopoulou, M.: A survey of temporal knowledge discovery paradigms and methods. In: IEEE TKDE, pp. 750–767 (2002)
23. Verma, K., Vyas, O.P., Vyas, R.: Temporal Approach to Association Rule Mining Using T-Tree and P-Tree. In: Perner, P., Imiya, A. (eds.) MLDM 2005. LNCS (LNAI), vol. 3587, pp. 651–659. Springer, Heidelberg (2005)
24. Wu, X., Zhang, C., Zhang, S.: Efficient mining of both positive and negative association rules. ACM Transactions on Information Systems 22(3), 381–405 (2004)
25. Zimbrao, G., de Souza, J.M., de Almeida, V.T., de Silva, W.A.: An algorithm to discover calendar-based temporal association rules with item's lifespan restriction. In: Proceedings of 2nd Workshop on Temporal Data Mining (2002)

Chapter 14
The Mamdani Expert-System with Parametric Families of Fuzzy Constraints in Evaluation of Cancer Patient Survival Length

Elisabeth Rakus-Andersson

Abstract. Strict analytic formulas are the tools usually derived for determining the formal relationships between a sample of independent variables and a variable which they affect. If we cannot formalize the function tying the independent and dependent variables then we will utilize some expert-system control actions. We often adopt their fuzzy variants developed by Mamdani, Sugeno and Takagi. Fuzzy expert-system algorithms are furnished with softer mechanisms, when comparing them to crisp versions. An efficient action of these softer mechanisms depends on the proper fuzzification of variables. At the stage of fuzzifying the variable levels we will prove some parametric expressions, which rearrange one function to several forms needed by the expert-system algorithm. The general parametric equation of membership functions allows creating arbitrary lists without any intuitive assumptions.

The fuzzy expert-system algorithms are particularly adaptable to support medical tasks to solve. These tasks often cope with uncertain premises and conclusions. From the medical point of view it would be desirable to prognosticate the survival length for patients suffering from gastric cancer. We thus formulate the objective of the current chapter as the utilization of the Mamdani fuzzy control actions as a methodology adapted for the purpose of making the survival prognoses.

Keywords: Parametric s-functions, Mamdani expert-system, estimation of survival length.

14.1 Introduction

In this chapter we will study the Mamdani methodology assumptions to support the approximation of the survival length. In the Mamdani model we wish to treat

Elisabeth Rakus-Andersson
Blekinge Institute of Technology, 37179 Karlskrona, Sweden
e-mail: Elisabeth.Andersson@bth.se

the survival length as a dependent variable, which is affected by some biological parameters. Strict analytic formulas are the tools usually derived for determining the formal relationships between a sample of independent variables and a variable which they affect.

If we cannot mathematically formulate some connections between the set of independent variables and the dependent variable then we will test the Mamdani procedure action.

Fuzzy set theory allows us to describe complex systems by using our knowledge and experience in transparent English-like rules. It does not need complex mathematical equations and system modelling that governs the relation between inputs and outputs.

Expert-knowledge designs together with assumptions of fuzzy set theory have given rise to the creation of fuzzy control procedures and its technical applications, see e.g., [1–2, 10–11, 14–16, 21–22, 24–25].

Experience-based rules constitute the crucial part of fuzzy control models, which have found many adherents to apply them in order to support solutions of complex systems not characterized by formally stated structures. The most popular types of fuzzy control systems are recognized as the Mamdani control [11], the Sugeno control [21–22] and Takagi-Sugeno control [15, 25].

Fuzzy methodology of Mamdani has also been tested in medicine [3, 7, 8].

Generally, the actions of the fuzzy expert systems consist of three basic parts: fuzzification process of input and output variables, processing procedures and defuzzification of the final fuzzy output set. In the processing part of the expert controlsystems we link the linguistic terms of input variables to the linguistic output state. Levels of independent variables, expressed as fuzzy sets, usually constitute the inputs in the model. Anyway, the choice of the output level pattern is a more sophisticated case. We place a fuzzified level of the output variable in the logical rules assisting the Mamdani methodology. In Sugeno and Sugeno-Takagi control systems the output levels are derived as functions. To find formulas of these functions we should use real data that come from some samplings showing relationships between collections of independent variables and a dependent variable. In logical IF-THEN rules of Mamdani system we mainly engage, instead, some experience in order to bind the levels of the rule antecedent to the rule consequence. As we are not given by dense data samples, which report gastric cancer patients, we have decided to adapt the Mamdani control as a methodology to evaluate the survival length for the patients. The surgeon's expertise will support the creation of the logical rules. By making the communication with the physician much easier we have divided the input and output variables into levels of intensity, expressed verbally.

The verbal descriptions of levels should be comprehensive in mathematical terms. We thus assign to the levels fuzzy sets characterized by families of membership functions. To implement the functions we decide a number of them in the partition of a reference set. The length of the reference set is the second data value used in an algorithm generating fuzzy constrains [17–19]. The design of functions is affected by two parameters, which results in the initiation of arbitrary numbers of uniform and symmetric membership function patterns. This technique of solving the first step of the Mamdani methodology (fuzzification of variables) is easily adaptable to the problem of prognosticating the survival length for patients who suffer from gastric cancer [19–20, 26–27].

The evaluation of survival length was already accomplished by statistical methods. In the first trials of survival approximation a survival curve from censored data was introduced [9]. The model was used in cancer patient examinations to estimate the length of living [13]. The Cox regression [4] of life length prediction was developed in such studies as logistic Cox regression [23]. The statistics-based models predicting the survival were compared by Everitt and Rabe-Hesketh [6] who found such model disadvantages as the lack of normal distribution or missing values among survival times.

In our study we test the Mamdani procedure to obtain a crisp value being the prediction of survival length. We make the length dependent on two clinical markers "*age*" and "*CRP-value*" due to the physicians' advice. The selection of *CRP* and age, as representative markers of post-surgical survival in cancer diseases, has been suggested due to the latest investigations revealing associations of these indices with the progression of disease in many cancer types [5, 12].

14.2 Making Fuzzification of Input and Output Variable Entries by Parametric *s*-Functions

The Mamdani model [11] is applied in research to the judgement of some relationships between a collection of independent variables and the dependent of them variable when we cannot formalize the functional connection among them.

The typical Mamdani system normally consists of three parts. The first part refers to the fuzzification process of input and output variables. These are first linguistically differentiated in levels. The levels are listed as names, which are demonstrated by fuzzy sets with assigned to them appropriate membership functions.

A fuzzy set, say, A in the universe X is a collection of elements assisted by the membership degrees that are computed by means of the membership function $\mu_A : X \to [0,1]$. Therefore A is denoted as $A = \{(x, \mu_A(x)), x \in X\}$. A is called normal if at least one element in the set A is assigned to the membership degree equal to 1. The support of A is a non-fuzzy set that consists of elements accompanying by membership degrees greater than 0.

The second part contains processing procedures. We rely on own experience when we prepare rules to link the linguistic terms of input variables to the output variable state. The rules employed in the model are constructed as IF-THEN statements. On the basis of the rules, verbally formulated and actual for individual tested values of independent variables, we estimate their mathematical consequences expressed as a collection of fuzzy sets. The creation of a sampling of all consequence sets results in one final consequence fuzzy set and terminates the action of the second step of the Mamdani methodology.

The last step of the Mamdani system is to defuzzify the final fuzzy output set being the result of the second stage. We adopt the centre of gravity (COG) method to convert the fuzzy set into a crisp value corresponding to the initial crisp input data.

We are expected to prove the Mamdani fuzzy system in order to evaluate the survival length in patients with diagnosis "*gastric cancer*". We have selected this system as the most effective one. The effectiveness of the Mamdani procedure is supported here by the surgeon's experience, which constitutes the most essential factor in the verbal communication to state the contents of IF-THEN rules. Otherwise, in the Sugeno model some output functions are desired. As our samplings of the patients' experimental data are rather poor, then we will not to take a risk to obtain improper formulas of output functions.

The period of survival is affected by two biological parameters X = "*age*" and Y = "*CRP-value*", which are selected as the most essential markers of making the prognosis. We cannot formally derive a function, which relates the independent variables X = "*age*" and Y = "*CRP-value*" to the dependent variable Z = "*survival length*"; therefore we will adapt the Mamdani system that supports estimation of dependent values in spite of the lack of a formula concerning $z = f(x, y)$, $x \in X$, $y \in Y$, $z \in Z$.

All variables will be differentiated into levels, which are expressed by lists of terms. The terms from the lists are represented by fuzzy sets, restricted by the parametric s-functions lying over the variable domains $[x_{\min}, x_{\max}]$, $[y_{\min}, y_{\max}]$ and $[z_{\min}, z_{\max}]$ respectively.

In conformity with the physician's suggestions we introduce five levels of X and Y as

X = "*age*" = { X_1 = "*very young*", X_2 = "*young*", X_3 = "*middle-aged*", X_4 = "*old*", X_5 = "*very old*"},

Y = "*CRP-value*" = { Y_1 = "*very low*", Y_2 = "*low*", Y_3 = "*medium*", Y_4 = "*high*", Y_5 = "*very high*"}

and we suggest seven levels of Z in the form of a collection

Z = "*survival length*" = { Z_1 = "*very short*", Z_2 = "*short*", Z_3 = "*rather short*", Z_4 = "*medium*", Z_5 = "*rather long*", Z_6 = "*long*", Z_7 = "*very long*"}.

To accomplish a formal mathematical design of level restrictions let us study the special own technique of their implementations [18–20].

In general, we suggest that the linguistic list of terms is converted to a sampling of fuzzy sets $L_1,\ldots,L_m$, where m is an odd positive integer greater or equal to 5. Each term is represented by the corresponding fuzzy set, whose constraint is supposed to be created as the common formula depending on the l^{th} value, where $l = 1,\ldots,m$. We assume that supports of restrictions $\mu_{L_l}(w)$, $l = 1,\ldots,m$, will cover parts of the reference set $L = [\min(L_1), \max(L_m)]$, $w \in L$. We introduce $E = |L|$ as the length of L.

We divide all expressions L_l in three groups, namely, a family of "*leftmost*" sets $L_1,\ldots,L_{\frac{m-1}{2}}$, the set $L_{\frac{m+1}{2}}$ "*in the middle*" and a collection of "*rightmost*" sets $L_{\frac{m+3}{2}},\ldots,L_m$. To design the membership functions of L_l the s-class function

$$s(w,\alpha,\beta,\gamma)=\begin{cases}0 & \text{for} \quad w\le\alpha,\\ 2\left(\frac{w-\alpha}{\gamma-\alpha}\right)^2 & \text{for} \quad \alpha\le w\le\beta,\\ 1-2\left(\frac{w-\gamma}{\gamma-\alpha}\right)^2 & \text{for} \quad \beta\le w\le\gamma,\\ 1 & \text{for} \quad w\ge\gamma,\end{cases} \tag{1}$$

will be adopted. The point (α, 0) starts the graph of the s-function, whereas the point (γ, 1) terminates this graph. The parameter β is found as the arithmetic mean of α and γ. In $w=\beta$ the s-function reaches the value of 0.5.

When designing parameters of each class function we want to consider the possibility to obtain the equal lengths of these parts of L_l's supports, which assist membership values greater than or equal to 0.5. The parts are regarded as the important representatives of fuzzy sets as they possess the largest index of the relationship to the set. We thus determine the breadth of each L_l to be $\frac{E}{m}$ on the membership level equal to 0.5.

Let us first design the parameters of the membership function "*in the middle*". The function of $L_{\frac{m+1}{2}}$ is constructed as a π-function

$$\pi(w)=\begin{cases}s(w,\alpha_1,\beta_1,\gamma_1=\alpha_2) & \text{for} \quad w\le\gamma_1=\alpha_2,\\ 1-s(w,\alpha_2=\gamma_1,\beta_2,\gamma_2) & \text{for} \quad w\ge\gamma_1=\alpha_2.\end{cases} \tag{2}$$

We suppose that $L_{\frac{m+1}{2}}$ will be a normal fuzzy set in $\gamma_1=\alpha_2=\frac{E}{2}$.

In order to guarantee the breadth $\frac{E}{m}$ on the membership level 0.5, function $L_{\frac{m+1}{2}}$ should take $\beta_1=\frac{E}{2}-\frac{E}{2m}=\frac{E(m-1)}{2m}$ and $\beta_2=\frac{E}{2}+\frac{E}{2m}=\frac{E(m+1)}{2m}$. Since $L_{\frac{m+1}{2}}$ is expected to preserve the uniform and symmetric shape then $\alpha_1=\frac{E}{2}-2\frac{E}{2m}=\frac{E(m-2)}{2m}$ and $\alpha_2=\frac{E}{2}+2\frac{E}{2m}=\frac{E(m+2)}{2m}$. We state $L_{\frac{m+1}{2}}$'s formula as

$$\mu_{L_{\frac{m+1}{2}}}(w)=\begin{cases}0 & \text{for} \quad w\le\frac{E(m-2)}{2m},\\ 2\left(\frac{w-\frac{E(m-2)}{2m}}{\frac{E}{m}}\right)^2 & \text{for} \quad \frac{E(m-2)}{2m}\le w\le\frac{E(m-1)}{2m},\\ 1-2\left(\frac{w-\frac{E}{2}}{\frac{E}{m}}\right)^2 & \text{for} \quad \frac{E(m-1)}{2m}\le w\le\frac{E}{2},\\ 1-2\left(\frac{w-\frac{E}{2}}{\frac{E}{m}}\right)^2 & \text{for} \quad \frac{E}{2}\le w\le\frac{E(m+1)}{2m},\\ 2\left(\frac{w-\frac{E(m+2)}{2m}}{\frac{E}{m}}\right)^2 & \text{for} \quad \frac{E(m+1)}{2m}\le w\le\frac{E(m+2)}{2m},\\ 0 & \text{for} \quad w\ge\frac{E(m+2)}{2m}.\end{cases} \tag{3}$$

For the "*leftmost*" family $L_1,\ldots,L_{\frac{m-1}{2}}$ we make suggestions that the top segments of functions lying on the membership level 1 will have the same lengths. Moreover, the last "*left*" function $L_{\frac{m-1}{2}}$ should have the intersection point with "*in the middle*" function on the membership level 0.5.

Each upper segment of L_t, $t = 1,\ldots,\frac{m-1}{2}$, will be thus equal to $\frac{\frac{E}{2}}{\frac{m-1}{2}+1}=\frac{E}{m+1}$. Particularly, $\alpha_{L_{\frac{m-1}{2}}}=\frac{E(m-1)}{2(m+1)}$ after multiplying the length of the distinct upper segment by the number of the last left function. We have already found $\beta_1=\frac{E(m-1)}{2m}$ of the "*in the middle*" function $L_{\frac{m+1}{2}}$. Due to the previously made assumption the functions $L_{\frac{m-1}{2}}$ and $L_{\frac{m+1}{2}}$ should intersect each other in point $\left(\frac{E(m-1)}{2m},0.5\right)$; therefore $\beta_{L_{\frac{m-1}{2}}}=\frac{E(m-1)}{2m}$. The difference between $\alpha_{L_{\frac{m-1}{2}}}$ and $\beta_{L_{\frac{m-1}{2}}}$ is evaluated to be $\frac{E(m-1)}{2m(m+1)}$. To find a uniform slope of $L_{\frac{m-1}{2}}$ we determine $\gamma_{L_{\frac{m-1}{2}}}=\beta_{L_{\frac{m-1}{2}}}+\frac{E(m-1)}{2m(m+1)}=\frac{E(m-1)(m+2)}{2m(m+1)}$.

Since the beginning of $L_{\frac{m-1}{2}}$ is planned to be placed in (min(L_1), 1) then $\mu_{L_{\frac{m-1}{2}}}(w)=1-s\left(w,\frac{E(m-1)}{2(m+1)},\frac{E(m-1)}{2m},\frac{E(m-1)(m+2)}{2m(m+1)}\right)$.

The membership function of $L_{\frac{m-1}{2}}$ is thus expanded as

$$\mu_{L_{\frac{m-1}{2}}}(w)=\begin{cases}1 & \text{for} \quad w\le\frac{E(m-1)}{2(m+1)},\\ 1-2\left(\frac{w-\frac{E(m-1)}{2(m+1)}}{\frac{E(m-1)}{m(m+1)}}\right)^2 & \text{for} \quad \frac{E(m-1)}{2(m+1)}\le w\le\frac{E(m-1)}{2m},\\ 2\left(\frac{w-\frac{E(m-1)(m+2)}{2m(m+1)}}{\frac{E(m-1)}{m(m+1)}}\right)^2 & \text{for} \quad \frac{E(m-1)}{2m}\le w\le\frac{E(m-1)(m+2)}{2m(m+1)},\\ 0 & \text{for} \quad w\ge\frac{E(m-1)(m+2)}{2m(m+1)}.\end{cases} \tag{4}$$

All constraints characteristic of the "*leftmost*" family of fuzzy sets will be given after inserting parameter $\delta(t)=\frac{2}{m-1}\cdot t$, $t = 1,\ldots,\frac{m-1}{2}$, in (4) to form it as [17]

$$\mu_{L_t}(w)=\begin{cases}1 & \text{for} \quad w\le\frac{E(m-1)}{2(m+1)}\delta(t),\\ 1-2\left(\frac{w-\frac{E(m-1)}{2(m+1)}\delta(t)}{\frac{E(m-1)}{m(m+1)}\delta(t)}\right)^2 & \text{for} \quad \frac{E(m-1)}{2(m+1)}\delta(t)\le w\le\frac{E(m-1)}{2m}\delta(t),\\ 2\left(\frac{w-\frac{E(m-1)(m+2)}{2m(m+1)}\delta(t)}{\frac{E(m-1)}{m(m+1)}\delta(t)}\right)^2 & \text{for} \quad \frac{E(m-1)}{2m}\delta(t)\le w\le\frac{E(m-1)(m+2)}{2m(m+1)}\delta(t),\\ 0 & \text{for} \quad w\ge\frac{E(m-1)(m+2)}{2m(m+1)}\delta(t).\end{cases} \tag{5}$$

Parameter $\delta(t)$ takes the value of 1 for $t = \frac{m-1}{2}$, which means that $\delta(\frac{m-1}{2})$ in (5) has no influence on the shape of the last left function. However, the introduction of $\delta(t)$ in (5) induces the narrowing effects in the supports of the other left function shapes. To preserve the same lengths of upper segments corresponding to membership 1 and middle segments attached to membership 0.5 we adjust $\delta(t)$, assisting the left function L_t, to be equal to $\frac{1}{\frac{m-1}{2}}$ multiplied by the function number t.

In order to start the implementation of the "*rightmost*" family functions let us note that the first right function $\mu_{L_{\frac{m+3}{2}}}(w)$ should possess $\mu_{L_{\frac{m-1}{2}}}(w)$'s inverted shape.

We generate the membership function of $L_{\frac{m+3}{2}}$ by

$$\mu_{L_{\frac{m+3}{2}}}(w) = \begin{cases} 0 & \text{for} \quad w \le E - \frac{E(m-1)(m+2)}{2m(m+1)}, \\ 2\left(\frac{w-\left(E-\frac{E(m-1)(m+2)}{2m(m+1)}\right)}{\frac{E(m-1)}{m(m+1)}}\right)^2 & \text{for} \quad E - \frac{E(m-1)(m+2)}{2m(m+1)} \le w \le E - \frac{E(m-1)}{2m}, \\ 1-2\left(\frac{w-\left(E-\frac{E(m-1)}{2(m+1)}\right)}{\frac{E(m-1)}{m(m+1)}}\right)^2 & \text{for} \quad E - \frac{E(m-1)}{2m} \le w \le E - \frac{E(m-1)}{2(m+1)}, \\ 1 & \text{for} \quad w \ge E - \frac{E(m-1)}{2(m+1)}. \end{cases} \tag{6}$$

The function of $L_{\frac{m+3}{2}}$ is symmetrically inverted to the function of $L_{\frac{m-1}{2}}$ over interval [min(L_1), max(L_m)].

Hence, the membership function $\mu_{L_{\frac{m-1}{2}}}(w) = 1 - s\left(w, \frac{E(m-1)}{2(m+1)}, \frac{E(m-1)}{2m}, \frac{E(m-1)(m+2)}{2m(m+1)}\right)$ will be changed into $\mu_{L_{\frac{m+3}{2}}}(w) = s\left(w, E - \frac{E(m-1)(m+2)}{2m(m+1)}, E - \frac{E(m-1)}{2m}, E - \frac{E(m-1)}{2(m+1)}\right)$.

To generate the "*rightmost*" family of sets $L_{\frac{m+3}{2}},\ldots,L_m$ we need to create a new parameter $\varepsilon(t) = 1 - \frac{2}{m-1}(t-1)$, $t = 1,\ldots,\frac{m-1}{2}$, which will be inserted in (6). The construction of $\varepsilon(t)$, when comparing to the creation of $\delta(t)$, is authorized by the fact that $t = 1$ should be followed by $\varepsilon(1) = 1$, whereas $t = \frac{m-1}{2}$ is helped by $\varepsilon(\frac{m-1}{2}) = \frac{2}{m-1}$. Formula (7) constitutes a common base for deriving membership functions

$$\mu_{L_{\frac{m+3}{2}+t-1}}(w) = \begin{cases} 0 & \text{for } w \le E - \frac{E(m-1)(m+2)}{2m(m+1)}\varepsilon(t), \\ 2\left(\frac{w-\left(E-\frac{E(m-1)(m+2)}{2m(m+1)}\varepsilon(t)\right)}{\frac{E(m-1)}{m(m+1)}\varepsilon(t)}\right)^2 & \text{for } E - \frac{E(m-1)(m+2)}{2m(m+1)}\varepsilon(t) \le w \le E - \frac{E(m-1)}{2m}\varepsilon(t), \\ 1-2\left(\frac{w-\left(E-\frac{E(m-1)}{2(m+1)}\varepsilon(t)\right)}{\frac{E(m-1)}{m(m+1)}\varepsilon(t)}\right)^2 & \text{for } E - \frac{E(m-1)}{2m}\varepsilon(t) \le w \le E - \frac{E(m-1)}{2(m+1)}\varepsilon(t), \\ 1 & \text{for } w \ge E - \frac{E(m-1)}{2(m+1)}\varepsilon(t). \end{cases} \tag{7}$$

The functions of fuzzy sets $L_1,...,L_m$ intend to maintain the same distances on the membership level 0.5. This property allows assigning to $L_1,...,L_m$ the relevant parts of their supports possessing the same length. The relevant parts of fuzzy sets consist of the sets' elements that reveal the membership degree values greater than or equal to 0.5. When forming the supports of the same length, in turn, we warrant the partition of $[\min(L_1),\max(L_m)]$ in equal subintervals standing for L_l levels, $l = 1,..,m$. Apart from that, the "*leftmost*" and "*rightmost*" functions also keep the same distances on the membership level 1. This feature provides us with a harmonious arrangement of function shapes.

All steps of the discussed algorithm, which initiates three sets of membership functions corresponding to a list of terms, can be sampled in the block scheme. We need to follow the steps of the scheme together with formulas (3), (5) and (7) to write the excerpt of a computer program. We emphasize that the only data, used in the algorithm, are the length of the reference set and the number of functions. We do not need to specify the sets' borders in the process of the program initialization, as most of programmers do, since the borders are computed automatically by formulas (3), (5) and (7). The steps of the algorithm flow chart are sampled in Fig. 1.

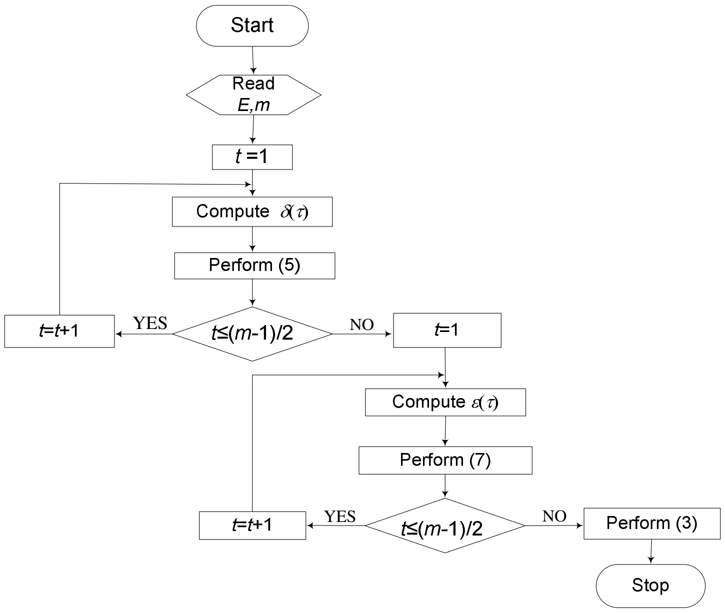


Fig. 1 The flow chart of the $L_1,\ldots,L_m$ implementation

The procedure discussed above has given rise to the introduction of membership functions typical of levels of X, Y and Z which, in turn, represent "*age*", "*CRP*" and "*survival*".

For five levels of $X = [0, 100]$, $L = X$, $w = x$, $m = 5$, $E = 100$, the leftmost family is revealed by

$$\mu_{X_t}(x)=\begin{cases}1 & \text{for} \quad x\le 33.33(0.5t),\\ 1-2\left(\frac{x-33.33(0.5t)}{13.33(0.5t)}\right)^2 & \text{for} \quad 33.33(0.5t)\le x\le 40(0.5t),\\ 2\left(\frac{x-46.66(0.5t)}{13.33(0.5t)}\right)^2 & \text{for} \quad 40(0.5t)\le x\le 46.66(0.5t),\\ 0 & \text{for} \quad x\ge 46.66(0.5t),\end{cases} \tag{8}$$

for $t = 1, 2$ due to (5).

The rightmost family of X-levels, composed in conformity with (7) is stated as

$$\mu_{X_{4+t-1}}(x)= \begin{cases}0 \text{ for } x\le 100-46.66(1-0.5(t-1)),\\ 2\left(\frac{x-(100-46.66(1-0.5(t-1)))}{13.33(1-0.5(t-1))}\right)^2\\ \text{for } 100-46.66(1-0.5(t-1))\le x\le 100-40(1-0.5(t-1)),\\ 1-2\left(\frac{x-(100-33.33(1-0.5(t-1)))}{13.33(1-0.5(t-1))}\right)^2\\ \text{for } 100-40(1-0.5(t-1))\le x\le 100-33.33(1-0.5(t-1)),\\ 1 \text{ for } x\ge 100-33.33(1-0.5(t-1)),\end{cases} \tag{9}$$

for $t = 1, 2$.

The “*in the middle*” X-level “*middle-aged*” has, in accord with (3), the constraint

$$\mu_{X_3}(x)=\begin{cases}0 & \text{for } x\le 30,\\ 2\left(\frac{x-30}{20}\right)^2 & \text{for } 30\le x\le 40,\\ 1-2\left(\frac{x-50}{20}\right)^2 & \text{for } 40\le x\le 50,\\ 1-2\left(\frac{x-50}{20}\right)^2 & \text{for } 50\le x\le 60,\\ 2\left(\frac{x-70}{20}\right)^2 & \text{for } 60\le x\le 70,\\ 0 & \text{for } x\ge 70.\end{cases} \tag{10}$$

All levels of X are sketched in Fig. 2.

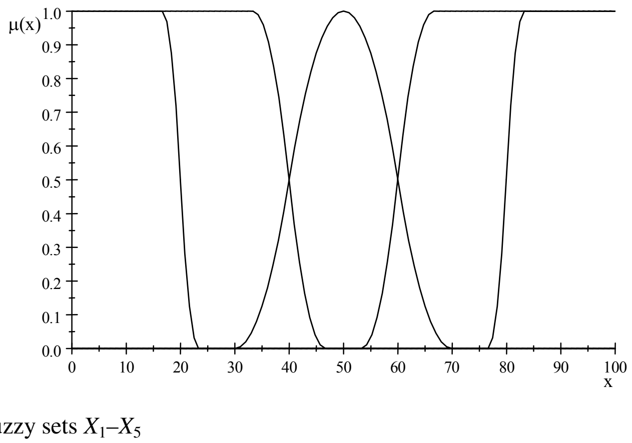


Fig. 2 The fuzzy sets X_1–X_5

By setting $Y = [0, 60]$, for $L = Y$, $w = y$, $E = 60$ and $m = 5$, we generate the leftmost Y-constraints

$$\mu_{Y_t}(y) = \begin{cases} 1 & \text{for } y \le 20(0.5t), \\ 1 - 2\left(\frac{y-20(0.5t)}{8(0.5t)}\right)^2 & \text{for } 20(0.5t) \le y \le 24(0.5t), \\ 2\left(\frac{y-28(0.5t)}{8(0.5t)}\right)^2 & \text{for } 24(0.5t) \le y \le 28(0.5t), \\ 0 & \text{for } y \ge 28(0.5t), \end{cases} \tag{11}$$

for $t = 1,2$.

For the same t-values the functions of Y's rightmost family are shaped by

$$\mu_{Y_{4+t-1}}(y) = \begin{cases} 0 \text{ for } y \le 60 - 28(1-0.5(t-1)), \\ 2\left(\frac{y-(60-28(1-0.5(t-1)))}{8(1-0.5(t-1))}\right)^2 \\ \text{for } 60 - 28(1-0.5(t-1)) \le y \le 60 - 24(1-0.5(t-1)), \\ 1 - 2\left(\frac{y-(60-20(1-0.5(t-1)))}{8(1-0.5(t-1))}\right)^2 \\ \text{for } 60 - 24(1-0.5(t-1)) \le y \le 60 - 20(1-0.5(t-1)), \\ 1 \text{ for } y \ge 60 - 20(1-0.5(t-1)), \end{cases} \tag{12}$$

whereas Y-level "*medium*" is established as

$$\mu_{Y_3}(y) = \begin{cases} 0 & \text{for} & y \le 18, \\ 2\left(\frac{y-18}{12}\right)^2 & \text{for} & 18 \le y \le 24, \\ 1-2\left(\frac{y-30}{12}\right)^2 & \text{for} & 24 \le y \le 30, \\ 1-2\left(\frac{y-30}{12}\right)^2 & \text{for} & 30 \le y \le 36, \\ 2\left(\frac{y-42}{12}\right)^2 & \text{for} & 36 \le y \le 42, \\ 0 & \text{for} & y \ge 42. \end{cases} \tag{13}$$

Fuzzy sets, corresponding to Y-levels are plotted in Fig. 3.

Finally, we utilize (3), (5) and (7) respectively to release membership functions of seven Z-levels. If $Z = [0, 6]$ then, for $L = Z$, $w = z$, $E = 6$ and $m = 7$, we will build the left restrictions

$$\mu_{Z_t}(z) = \begin{cases} 1 & \text{for} & z \le 2.25(0.33t), \\ 1 - 2\left(\frac{z-2.25(0.33t)}{0.64(0.33t)}\right)^2 & \text{for} & 2.25(0.33t) \le z \le 2.57(0.33t), \\ 2\left(\frac{z-2.89(0.33t)}{0.64(0.33t)}\right)^2 & \text{for} & 2.57(0.33t) \le z \le 2.89(0.33t), \\ 0 & \text{for} & z \ge 2.89(0.33t), \end{cases} \tag{14}$$

where $t = 1, 2, 3$.

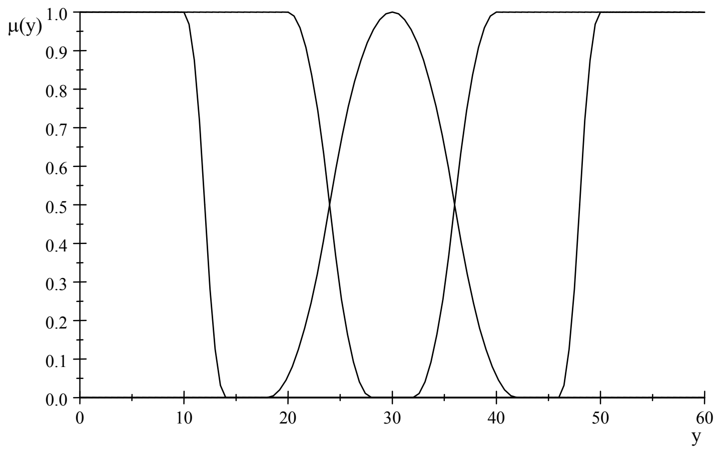


Fig. 3 The fuzzy sets Y_1-Y_5

The constrains of right Z-levels are results of

$$\mu_{Z_{5+t-1}}(z) = \begin{cases} 0 \text{ for } z \le 6-2.89(1-0.33(t-1)), \\ 2\left(\frac{z-(6-2.89(1-0.33(t-1)))}{0.64(1-0.33(t-1))}\right)^2 \\ \text{for } 6-2.89(1-0.33(t-1)) \le z \le 6-2.57(1-0.33(t-1)), \\ 1-2\left(\frac{z-(6-2.25(1-0.33(t-1)))}{0.64(1-0.33(t-1))}\right)^2 \\ \text{for } 6-2.57(1-0.33(t-1)) \le z \le 6-2.25(1-0.33(t-1)), \\ 1 \text{ for } z \ge 6-2.25(1-0.33(t-1)), \end{cases} \tag{15}$$

when setting $t = 1, 2, 3$.

The medium Z-level Z_4 possesses the membership function

$$\mu_{Z_4}(z) = \begin{cases} 0 & \text{for } z \le 2.14, \\ 2\left(\frac{z-2.14}{0.86}\right)^2 & \text{for } 2.14 \le z \le 2.57, \\ 1-2\left(\frac{z-3}{0.86}\right)^2 & \text{for } 2.57 \le z \le 3, \\ 1-2\left(\frac{z-3}{0.86}\right)^2 & \text{for } 3 \le z \le 3.43, \\ 2\left(\frac{z-3.86}{0.86}\right)^2 & \text{for } 3.43 \le z \le 3.86, \\ 0 & \text{for } z \ge 3.86. \end{cases} \tag{16}$$

The graphs of all Z-levels are collected in Fig. 4.

We emphasize the importance of the parametric design of functions. Instead of implementing seventeen formulas of the similar pattern, we have sampled all functions in three generic groups (left, right and in the middle). Any time we can involve the desired function in necessary computations by setting its t-number in the proper formula concerning *X*, *Y* or *Z*. Moreover, the mathematical scenario of membership functions is established in the formal and elegant designs, which can be segments of a computer program for the reason of their nature letting the creation of loops.

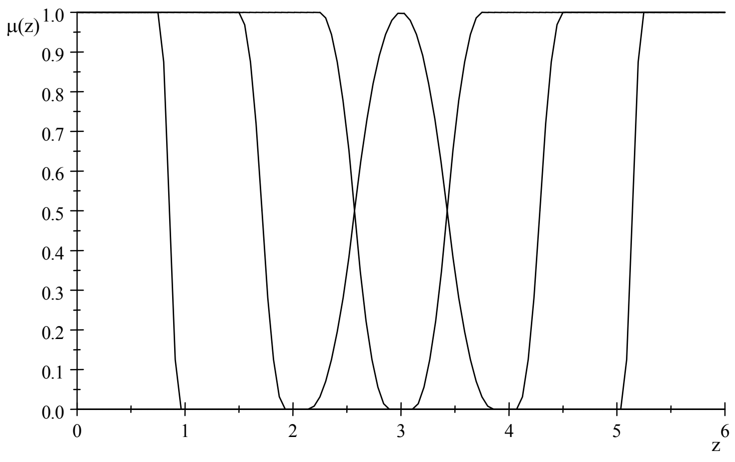


Fig. 4 The fuzzy sets Z_1-Z_7

14.3 The Rule Based Processing Part of Surviving Length Model

After the fuzzification procedure we are able to create the rule bases that link the states of the two input variables to the state of the output variable. We thus design a table, in which the entries are filled with terms of "*survival length*". To express the states of the survival length as logically as possible, we have also studied the behaviour of variables on the basis of biological data samplings. The cells of the table are characterized by subintervals of domains of *X* and *Y*.

We first estimate the survival length median in the samplings of the data corresponding to considered cells. The median value was set as *z* in the membership functions of all fuzzy sets listed in the *Z*-space. We select this fuzzy set Z_k, k = 1,…,7, as a representative of the cell, in which the membership degree of the median was largest. The technique of combining the human experience with data sets, obtained for discrete samples to make conclusions referring to continuous samples, is a modern branch of so-called "integration systems".

The estimations of survival length are sampled in Table 1.

Some entries in the table are empty, since the essential data was lacking for younger people. It rarely happens to find young patients with diagnosis "*gastric cancer*"; therefore we could not make any reliable conclusions concerning survival in this age group. For old individuals the period of 6 years of surviving has been established as an upper border.

Table 1 Estimation of "*survival length*" as a rule base

X_i/Y_j	very low	low	medium	high	very high
very young					
young					
middle-aged	long				
old	rather long	rather short	short	short	very short
very old	rather short	short	very short	very short	very short

Suppose that we would like to make the survival prognosis for (x, y), $x \in X$, $y \in Y$ – with other words we want to evaluate $z = f(x, y)$ when assuming that the *f*-formula is not developed.

Furthermore, x often belongs to more than one fuzzy sets X_i, i = 1,...,5, with different membership degrees equaling $\mu_{X_i}(x)$. Element y associated to x can be a member of some fuzzy sets Y_j, j = 1,...,5, in which it takes membership degrees $\mu_{Y_j}(y)$.

By means of IF-THEN statements grounded on the basis of Table 1, we can determine the contents of rules by attaching the pair of input variable levels to a level of the output variable according to

$$\text{Rule } R_{(x,y):r}\text{: If } x \text{ is } X_{i:r} \text{ and } y \text{ is } Y_{j:r}\text{, then } z \text{ is } Z_{k:r}, \tag{17}$$

where r is the rule number. The expressions $X_{i:r}$, $Y_{j:r}$ and $Z_{k:r}$ denote the fuzzy sets X_i, Y_j and Z_k assisting rule number r. $R_{(x,y):r}$ is determined for actual x and y.

To evaluate the influence of the input variables on the output consequences we need an estimate $\alpha_{(x,y):r}$ computed by performing the minimum operation

$$\alpha_{(x,y):r} = \min\left(\mu_{X_i:r}(x), \mu_{Y_j:r}(y)\right) \tag{18}$$

for each $X_{i:r}$ and $Y_{j:r}$ concerning the choice of (x,y).

We use $\alpha_{(x,y):r}$ and the minimum operator to determine consequences of all rules $R_{(x,y):r}$ for the output. Fuzzy sets $R_{(x,y):r}^{conseq}$, stated in the output space Z, will have the membership functions

$$\mu_{R_{(x,y):r}^{conseq}}(z) = \min\left(\alpha_{(x,y):r}, \mu_{Z_k:r}(z)\right). \tag{19}$$

In the last step of the processing part we aggregate the consequence sets $R_{(x,y):r}^{conseq}$ in one common set $conseq_{(x,y)}$ allocated in Z over a continuous interval [z_0, z_n]. To derive the membership function of $conseq_{(x,y)}$ we prove the action of the maximum operator in the form of

$$\mu_{conseq_{(x,y)}}(z) = \max_r \left(\mu_{R_{(x,y):r}}{}^{conseq}(z)\right). \tag{20}$$

14.4 Defuzzification of the Output Variable

In order to assign a crisp value z to the selected pair (x, y) we defuzzify the consequence fuzzy set in Z. We will thus indicate the expected value of the survival length for a gastric cancer patient whose age x and *CRP*-value y have been examined.

As a defuzzification rule we select the centre of gravity method (COG). This model of computing is clearly interpretable. We expand COG as

$$z = f(x, y) =$$

$$\frac{\int_{z_0}^{z_n} z \cdot \mu_{conseq_{(x,y)}}(z)dz}{\int_{z_0}^{z_n} \mu_{conseq_{(x,y)}}(z)dz} = \frac{\int_{z_0}^{z_1} z \cdot \mu_{conseq_{(x,y)}}(z)dz + \cdots + \int_{z_{n-1}}^{z_n} z \cdot \mu_{conseq_{(x,y)}}(z)dz}{\int_{z_0}^{z_1} \mu_{conseq_{(x,y)}}(z)dz + \cdots + \int_{z_{n-1}}^{z_n} \mu_{conseq_{(x,y)}}(z)dz} \tag{21}$$

with the inner borders $z_1,\ldots,z_{n-1}$ being either z-coordinates of intersection points between adjacent branches of the $conseq_{(x,y)}$ membership function or characteristic support values of fuzzy sets included in $conseq_{(x,y)}$.

14.5 The Survival Length Prognosis for a Selected Patient

Suppose that we examine a 79-year-old patient, whose *CRP*-value is 25. His diagnosis is determined by a physician as "*gastric cancer*". We wish to estimate theoretically the expected value of his survival length by proving the algorithm sketched in previous sections. The information is confidential and used only by the physician.

Let $x = 79$ and $y = 25$. Age 79 belongs to fuzzy set X_5 = "*very old*". We want to find the membership degree of 79 in "*very old*", which belongs to the rightmost family of X; therefore we put $t = 2$ in (9) to induce

$$\mu_{X_5}(x) = \begin{cases} 0 & \text{for} \quad x \le 76.665, \\ 2\left(\frac{x-76.665}{6.67}\right)^2 & \text{for} \quad 76.665 \le x \le 80, \\ 1-2\left(\frac{x-83.335}{6.67}\right)^2 & \text{for} \quad 80 \le x \le 83.335, \\ 1 & \text{for} \quad x \ge 83.335. \end{cases}$$

As $79 \in [76.665, 80]$ then $\mu_{X_5}(79) = 2\left(\frac{79-76.665}{6.67}\right)^2 = 0.245.$

Age 79 is also a member of X_4 = "*old*" with the membership degree $\mu_{X_4}(79)=1$, due to (9) when t = 1.

Value y = 25 is located in Y_2 ="*low*" with $\mu_{Y_2}(25)=0.281$ and in Y_3 = "*medium*" with $\mu_{Y_3}(25)=0.347$. To evaluate the values of membership degrees in respective fuzzy sets in Y we have been referred to formula (11) to reach Y_2 and to formula (13) to get Y_3.

In accordance with (17) and available information from Table 1 we establish the rules, which connect the states of the input variables to the output variable levels. We list the rules in the following order:

$R_{(79,25):1}$: IF x = 79 is X_4 = "*old*" and y = 25 is Y_2 = "*low*", THEN z = "*survival length*" will be Z_3 = "*rather short*",

$R_{(79,25):2}$: IF x = 79 is X_4 = "*old*" and y = 25 is Y_3 = "*medium*", THEN z = "*survival length*" will be Z_2 = "*short*",

$R_{(79,25):3}$: IF x = 79 is X_5 = "*very old*" and y = 25 is Y_2 = "*low*", THEN z = "*survival length*" will be Z_2 = "*short*",

$R_{(79,25):4}$: IF x = 79 is X_5 = "*very old*" and y = 25 is Y_3 = "*medium*", THEN z = "*survival length*" will be Z_1 = "*very short*".

To evaluate the influences of the input variables on the output consequences due to (18), we estimate $\alpha_{(79,25):r}$, r = 1,...,4 as quantities

$$\alpha_{(79,25):1}=\min\left(\mu_{X_4:1}(79),\mu_{Y_2:1}(25)\right)=\min(1,0.281)=0.281,$$
$$\alpha_{(79,25):2}=\min\left(\mu_{X_4:2}(79),\mu_{Y_3:2}(25)\right)=\min(1,0.347)=0.347,$$
$$\alpha_{(79,25):3}=0.245,\ \alpha_{(79,25):4}=0.245.$$

In conformity with formula (19) we obtain the fuzzy subsets of the consequences. Set $R_{(79,25):1}{}^{conseq}$ has a membership function

$\mu_{R_{(79,25):1}}{}^{conseq}(z)=\min\left(\alpha_{(79,25):1},\mu_{Z_3:1}(z)\right)=\min\left(0.281,\mu_{\text{"}rather\,short\text{"}}(z)\right)$ given by Fig. 5.

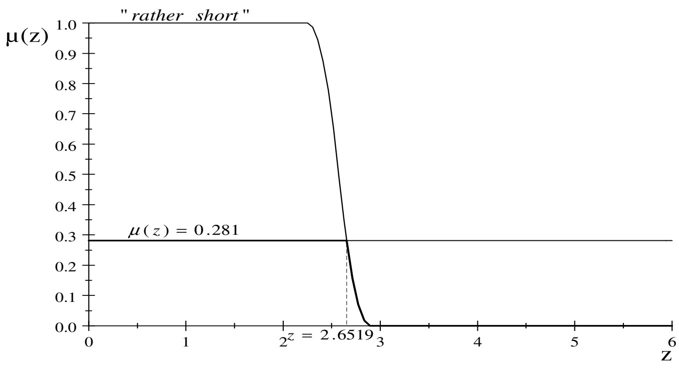


Fig.5 The fuzzy subset of consequence constructed due to $R_{(79,25):1}$

The next set $R_{(79,25):2}{}^{conseq}$ is characterized by a constraint $\mu_{R_{(79,25):2}}{}^{conseq}(z) = \min\left(\alpha_{(79,25):2}, \mu_{"Z_2:2"}(z)\right) = \min\left(0.347, \mu_{"short"}(z)\right)$ drawn in Fig. 6.

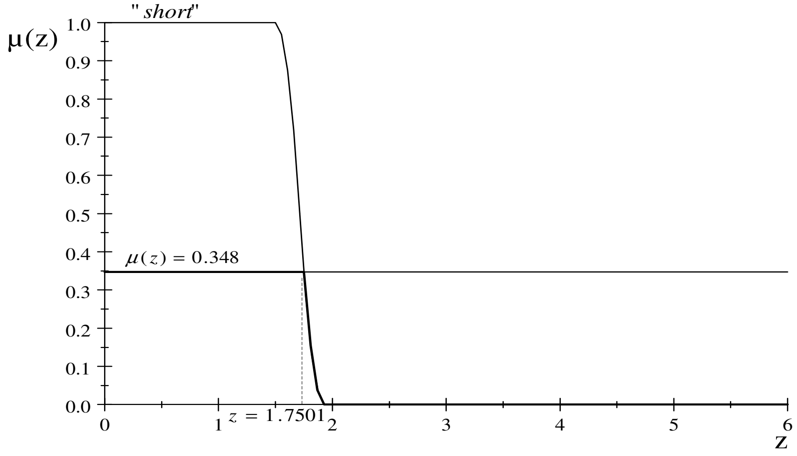


Fig. 6 The fuzzy subset of consequence constructed due to $R_{(79,25):2}$

The third set $R_{(79,25):3}{}^{conseq}$ possesses a function

$$\mu_{"R_{(79,25):3}"}{}^{conseq}(z) = \min\left(\alpha_{(79,25):3}, \mu_{Z_2:3}(z)\right) = \min\left(0.245, \mu_{"short"}(z)\right)$$

whose graph is revealed in Fig. 7.

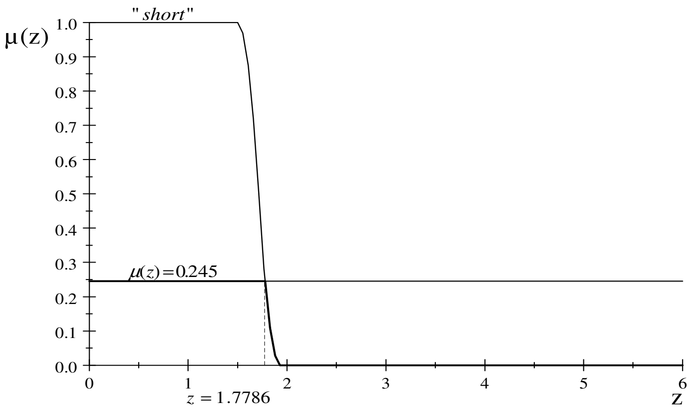


Fig. 7 The fuzzy subset of consequence constructed for $R_{(79,25):3}$

The last set $R_{(79,25):4}{}^{conseq}$ is restricted by a function $\mu_{R_{(79,25):4}}{}^{conseq}(z) = \min\left(\alpha_{(79,25):4}, \mu_{Z_1:4}(z)\right) = \min\left(0.245, \mu_{\text{"}very\ short\text{"}}(z)\right)$ plotted in Fig. 8.

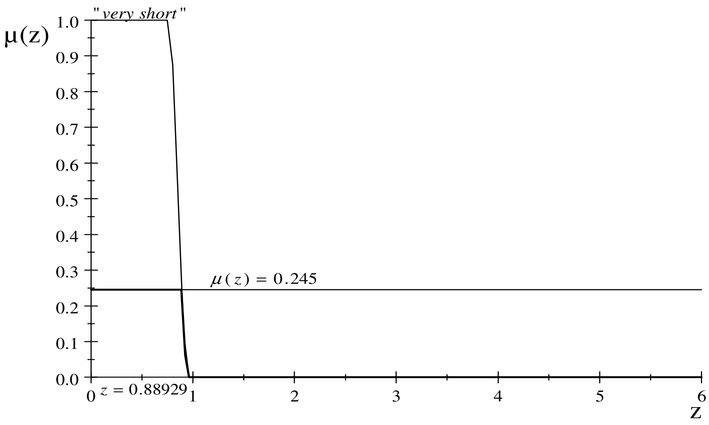


Fig. 8 The fuzzy subset of consequence constructed in accord to $R_{(79,25):4}$

When applying formula (20) we concatenate all $\mu_{R_{(x,y):r}}{}^{conseq}(z)$, $r = 1,2,3,4$, in order to determine a common consequence of rules (17) fitted for the pair (79, 25). The fuzzy subset of the universe Z will be thus yielded by its membership function

$$\mu_{conseq_{(79,25)}}(z) = \max_{1 \le r \le 4}\left(\mu_{R_{(79,25):r}}{}^{conseq}(z)\right).$$

The fuzzy set $conseq_{(79,25)}$ is aggregated in Fig. 9.

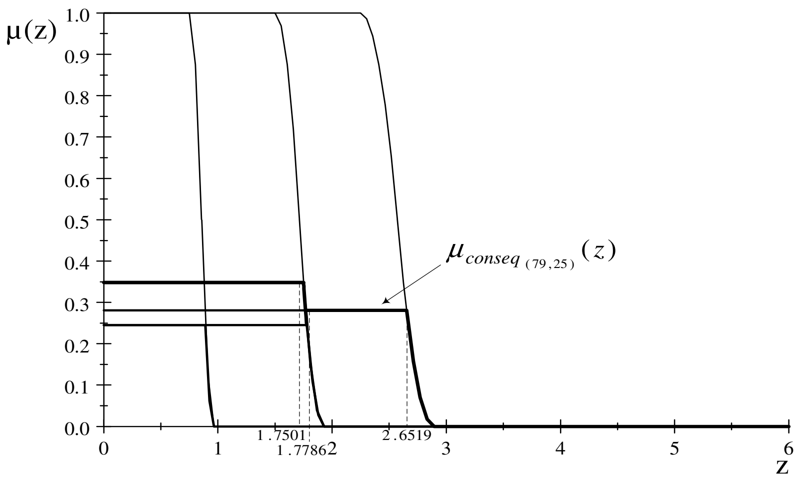


Fig. 9 The consequence set $conseq_{(79,25)}$ in Z

Formula (21) constitutes a basis of an estimation of the survival length expected when assuming "*age*" = 79 and "*CRP-value*" = 25. Over interval $[z_0, z_4]$ = [0, 2.8928], which contains characteristic points $z_0 = 0$, $z_1 = 1.7501$, $z_2 = 1.7786$, $z_3 = 2.6519$ and $z_4 = 2.8928$, we compute the z-prognosis

$$z = f(79,25) = \frac{\int_{0}^{1.7501} 0.348z\,dz + \int_{1.7501}^{1.7786} \left(2\left(\frac{7}{3}z-\frac{9}{2}\right)^2\right) z\,dz + \int_{1.7786}^{2.6519} 0.281z\,dz + \int_{2.6519}^{2.9828} \left(2\left(\frac{14}{9}z-\frac{9}{2}\right)^2\right) z\,dz}{\int_{0}^{1.7501} 0.348\,dz + \int_{1.7501}^{1.7786} \left(2\left(\frac{7}{3}z-\frac{9}{2}\right)^2\right) dz + \int_{1.7786}^{2.6519} 0.281\,dz + \int_{2.6519}^{2.9828} \left(2\left(\frac{14}{9}z-\frac{9}{2}\right)^2\right) dz} \approx 1.3.$$

For the patient who is 79 years old and has the *CRP*-value equal to 25, the theoretical estimated survival length is over one year. The result converges to the physician's own judgment made on the basis of his medical reports. For each pair (x,y) we can arrange new computations due to the Mamdani algorithm based on the expert-system rules to estimate the patient's period of surviving in the case of suffering from gastric cancer.

14.6 Conclusions

The fuzzy expert-systems are powerful methods, which mostly are applied to technologies checking complex processes by means of human experience. In this work we have proved that the expected values of patients' survival lengths can be estimated even if the mathematical formalization involving independent and dependent variables is unknown.

When adapting the Mamdani procedure to medical dependency assumptions we have supported the evaluation of the survival length, made so far by statistical tests.

Due to the results of performed computations we can conclude that the Mamdani algorithm demands a large number of operations in the processing phase. Nevertheless, we can always construct logical rules IF-THEN, which are based on variable levels and are assisted by fuzzy sets. Even if the data from point sets is lacking, it will be still possible to make a trial of designing membership functions for all levels of variables by relying on the human expertise.

The results brought by the Mamdani procedure encounter results coming from statistical experiments. Moreover, the Mamdani fuzzy method does not demand particular assumptions like normal distributions of the dependent variables.

In the future experiments we want to construct the computer program to cover the rectangle $[x_{\min}, x_{\max}] \times [y_{\min}, y_{\max}]$ with a surface, which allows to read off the desired value of z for an arbitrary pair (x, y). In that way we will solve the problem of the continuous evaluation of survival length as it has been recommended by our co-operating physicians. It will be desirable to introduce more independent variables to the model as well.

For the purpose of the Mamdani procedure we have initially designed families of fuzzy sets that are affected by parameters. In many algorithms the boundary values of membership functions, representing a certain list, are introduced as the initial data. We have used the pattern of the s-functions to model constrains of fuzzy sets without their predetermined start points and endpoints. For a large amount of functions a computer program is recommended to produce uniform and symmetric shapes of fuzzy restrictions. Due to the flow chart proposed, which constitutes the original model created by the author, we can easily convert the scheme to a sequence of program commands. We emphasize that the only data required by the model are restricted to the number of functions and the length of the reference set.

We hope that all transformations performed on the membership functions have demonstrated the logical and elegant design, which should be regarded as an advantage of the presented model.

In the end, we emphasize that the Mamdani methodology is supported by easily performed mathematical operations unlike, e.g., differential equations typical of other traditional expert-based systems.

Acknowledgment. The author thanks the Blekinge Research Board for the grant funding the current research. The author is also grateful to Associate Professor in Medicine Henrik Forssell for medical consulting of this report.

References

1. Al-Odienat, A.I., Al-Lawama, A.A.: The Advantages of PID Fuzzy Controllers over the Conventional Types. American Journal of Applied Sciences 5(6), 653–658 (2008)
2. Andrei, N.: Modern Control Theory: a Historical Perspective. Centre for Advanced Modelling and Optimization. Research Institute for Informatics, Romania (2005), `http://www.ici.ro/camo/neculai/history.pdf`
3. Chen, C.-T., Lin, W.-L., Kuo, T.-S., Wang, C.-Y.: Blood Pressure Regulation by Means of a Neuro-fuzzy Control System. In: The 18th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Amsterdam, pp. 1725–1726 (1996)
4. Cox, D.: Regression Models and Life Tables. J. Roy. Stat. Soc. B 4, 187–220 (1972)
5. Kim, D.-K., Oh, S.Y., Kwon, H.-C., Lee, S., Kwon, K.A., Kim, B.G., Kim, S.-G., Kim, S.-H., Jang, J.S., Kim, M.C., Kim, K.H., Han, J.-Y., Kim, H.-J.: Clinical Significances of Preoperative Serum Interleukin-6 and C-reactive Protein Level in Operable Gastric Cancer. BMC Cancer 9, 155–156 (2009)
6. Everitt, B., Rabe-Hesketh, S.: Analyzing Medical Data Using S-PLUS. Springer, New York (2001)
7. Hernández, C., Carollo, A., Tobar, C.: Fuzzy Control of Postoperative Pain. In: Proceedings of the Annual International Conference of the IEEE, pp. 2301–2303 (1992)
8. Isaka, S., Sebald, A.V.: An Adaptive Fuzzy Controller for Blood Pressure Regulation. In: The 11th Annual International Conference on IEEE Engineering in Medicine & Biology Society, pp. 1763–1764 (1989)
9. Kaplan, E., Meier, P.: Nonparametric Estimation from Incomplete Observations. Journal American Statistical Association 53, 457–481 (1958)
10. Ma, X.J., Sun, Z.Q., He, Y.Y.: Analysis and Design of Fuzzy Controller and Fuzzy Observer. IEEE Transactions on Fuzzy Systems 6(1), 41–51 (1998)
11. Mamdani, E.H., Assilian, S.: An Experiment in Linguistic Synthesis with a Fuzzy Logic Controller. Int. J. Man-Machine Studies 7, 1–13 (1973)

12. de Mello, J., Struthers, L., Turner, R., Cooper, E.H., Giles, G.R.: Multivariate Analyses as Aids to Diagnosis and Assessment of Prognosis in Gastrointestinal Cancer. Br. J. Cancer 48, 341–348 (1983)
13. Newland, R.C., Dent, O.F., Lyttle, M.N., Chapuis, P.H., Bokey, E.L.: Pathologic Determinants of Survival Associated with Colorectal Cancer with Lymph Node Metastases. A Multivariate Analysis of 579 Patients. Cancer 73(8), 2076–2082 (1994)
14. Nguyen, H.T., Prasad, N.R., Walker, C.L., Walker, E.A.: A First Course in Fuzzy and Neural Control. Chapman & Hall/CRC (2002)
15. Passino, K.M., Yurkovich, S.: Fuzzy Control. Addison-Wesley Longman Inc. (1997)
16. Preitl, S., Precup, R.E., Preitl, Z.: Development of Conventional and Fuzzy Controllers and Takagi-Sugeno Fuzzy Models Dedicated for Control of Low Order. Acta Polytechnica Hungarica 2(1), 75–92 (2005)
17. Rakus-Andersson, E.: Fuzzy and Rough Sets in Medical Diagnosis and Medication. Springer, Heidelberg (2007)
18. Rakus-Andersson, E.: Adjusted s-parametric Functions in the Creation of Symmetric Constraints. In: Proceedings of the 10th International Conference on Intelligent Systems Design and Applications, ISDA 2010, Cairo, Egypt, pp. 451–456 (2010)
19. Rakus-Andersson, E.: Approximate Reasoning in Cancer Surgery. In: Proceedings of the International Conference on Fuzzy Computation Theory and Applications, FCTA 2011, Paris, France, pp. 466–469 (2011)
20. Rakus-Andersson, E., Zettervall, H., Forssell, H.: Fuzzy Controllers in Evaluation of Sur-vival Length in Cancer Patients. In: Recent Advances in Fuzzy Sets, In: Intuitionistic Fuzzy Sets, Generalized Nets and Related Topics. Volume II: Applications, Polish Academy of Sciences, System Research Institute, Warsaw, pp. 203-222 (2011)
21. Sugeno, M.: An Introductory Survey of Fuzzy Control. Inf. Sci. 36, 59–83 (1985)
22. Sugeno, M., Nishida, M.: Fuzzy Control of Model Car. Fuzzy Sets and Systems 16(2), 103–113 (1985)
23. Sargent, D.J.: Comparison of Artificial Networks with Other Statistical Approaches. Cancer 91, 1636–1942 (2001)
24. Sutton, R., Towill, D.R.: An Introduction to the Use of Fuzzy Sets in the Implementation of Control Algorithms. IEEE Trans., UDC 510.54:62-519:629.12.014.5 (1985), paper no. 2208/ACS39
25. Takagi, T., Sugeno, M.: Fuzzy Identification of Systems and Its Applications to Modeling and Control. IEEE Transactions on Systems, Man and Cybernetics SMC-15(1), 116–132 (1985)
26. Zettervall, H., Rakus-Andersson, E., Forssell, H.: The Mamdani Controller in Prediction of the Survival Length in Elderly Gastric Patients. In: Proceedings of Bioinformatics 2011, Rome, pp. 283–286 (2011)
27. Zettervall, H.: Fuzzy and Rough Theory in the Treatment of Elderly Gastric Cancer Patients. Licentiate Dissertation, Karlskrona, Sweden (2011)

Chapter 15
Support Vector Machines in Biomedical and Biometrical Applications

Krzysztof A. Cyran, Jolanta Kawulok, Michal Kawulok,
Magdalena Stawarz, Marcin Michalak, Monika Pietrowska,
Piotr Widłak, and Joanna Polańska

Abstract. In the chapter, a background review material concerning applications of the kernel methods in computational biology and biometry is illustrated by the case studies concerning the proteomic spectra analysis to find diagnostic biomarkers and performing case-control discrimination as well as the face recognition problem, which is situated among the most investigated

Krzysztof A. Cyran
Institute of Informatics
e-mail: Krzysztof.Cyran@polsl.pl

Jolanta Kawulok
Institute of Informatics
e-mail: Jolanta.Kawulok@polsl.pl

Michal Kawulok
Institute of Informatics
e-mail: Michal.Kawulok@polsl.pl

Magdalena Stawarz
Institute of Informatics
e-mail: Magdalena.Stawarz@polsl.pl

Marcin Michalak
Institute of Informatics
e-mail: Marcin.Michalak@polsl.pl

Monika Pietrowska
Institute of Oncology
e-mail: M_Pietrowska@io.gliwice.pl

Piotr Widłak
Institute of Oncology
e-mail: Widlak@io.gliwice.pl

Joanna Polańska
Institute of Automatic Control
e-mail: Joanna.Polanska@polsl.pl

biometric methods. These case studies, representing the state-of-the-art in applications of the support vector machines (SVM) in biomedical and biometrical applications, are the examples of a research work conducted by computer scientists, bioinformaticians, and biostatisticians from the Faculty of Automatic Control, Electronics and Computer Science at Silesian University of Technology in a collaboration with clinicists from the Institute of Oncology in Gliwice, Poland.

15.1 Introduction

Support Vector Machines (SVM) were presented for the first time in 1992 as a technique for maximizing the margin that separates two classes [4]. There are two main models for building the SVM classifier, namely linear and nonlinear SVM. The first one assumes that there exists an optimal hyperplane in the data set feature space which separates objects from two classes. The latter uses a ,,kernel trick" to find optimal separating linear hyperplanes in higher dimensions that are nonlinear in the original feature space.

It is worth to decipher the notion of a ,,support vector". Let us assume that the number of vectors in the training set is n. In the case of two dimensional vectors belonging to two linearly separable classes only three vectors are needed to determine the margin: two of them define the hyperplane (a line for two-dimensional case), and the third one determines the margin width. If some other points (different from the mentioned three) were removed from the training set, the SVM classifier would give the same separating line. Therefore in other words these three vectors support the margin and are called ,,support vectors" (SVs).

In the original SVM algorithm there is no possibility to modify the number of SVs - only the margin width, determined by the so-called ε-intensive loss function is modifiable. However, amongst other SVM modifications it is worth to mention the v-SVM where v is the fraction of SVs from the entire training set [54]. Increasing v makes it possible to obtain a more complicated model that separates the classes better and decreasing v generates a less accurate, but more general model.

v-SVM and the original SVM are based on the assumption that the level of noise is uniform in the whole domain. This limitation is removed by the $par - v$-SVM model [16] where the margin with constant width is replaced with the margin defined by two functions: separating hyperplane and the new function $g(x)$. The local width of the margin equals to $2|f(x) - g(x)|$.

Based on the SVM, also a new branch of nonparametrical regression has been developed, called Support Vector Regression (SVR). Here, the separating hyperplane was substituted by the regression hyperplane and the separation margin was replaced by the notion of a regression tube. SVR may be described with the known sentence ,,through every three points on the plane the line can be drawn, provided to it will be thick enough". Then the regression function is the center of the ,,thick line" and the whole line is the regression tube. SVR was also defined in the classical way (ε-SVR [11]), as v-SVR [53] and $par - v$-SVR [16]. An interesting extension of SVR is presented in [9] where instead of a single regression function two bound functions (upper and lower) are introduced.

As all the mentioned SV algorithms are based on solving quadratic programming problem, there are lot of methods that substitute this step with some heuristics. DirectSVM [51] is an iterative algorithm which builds the support vector set incrementally. It combines the theorethical foundations of SVM and formulation of neural networks, which allows to avoid solving quadratic optimization problem. For this reason the time of computing is much shorter and it requires less computational efforts than original SVM method, affording similar results. Based on two simple heuristics this algorithm finds the set of candidate support vectors to determine parameters of the optimal hyperplane. The first heuristic assumes that if two elements belonging to the opposite classes are in the closest distance to each other, they are selected as an initial candidate support vector. If it occurs that they are not support vectors, the next couple is taken under consideration, etc. The second heuristic assumes that an element from the training set that maximally violates the current position of the hyperplane is the support vector. The algorithm finds this maximum violator in each iteration and then the orientation of candidate Support Plane has to be turned to make it a Support Vector.

The SimpleSVM algorithm [65] also starts off with the closest pair of points from the opposite classes by adding them to the candidate Support Vector set like DirectSVM [51]. If the inclusion of the new candidate SV to candidate set is prevented by other points already belonging to this set the algorithm uses a backtracking approach to cut them off from set. The iteration through the all dataset is repeated until no violators can be found to guarantee that Karush-Kuhn-Tucker conditions are satisfied. In order to insure data linear separability it use quadratic penalty formulation.

Another method called Local Incremental Learning of SVM (LISVM) is based on Radial Basis Function Kernel [49]. On-line learning requires actualization of the classification process results always when new data are added to the current dataset. LISVM, instead of learning all datasets again, assumes that it can be carried out only on a neighborhood of the new data and the training data can be updated by changing their weights, but only this subset of support candidates should be re-learning that lies in neighborhood of input. This subset is obtained based on the variation of the error estimate. Neighborhood is defined as limited area on which support vectors have a higher influence. The size of neighborhood has to be computed at each step, to determine if they use criterion coming from [10].

Incremental and Decremental Support Vector Machine Learning is the next online variant of SVM [7]. This algorithm gives the exact solutions of classification problem based on incremental, reversible procedure. It can be ensured by keeping Kuhn-Tucker conditions satisfied on the current training dataset while the new point is added. The decremental part of this method allows to obtain exact leave-one-out generalization effectiveness, which require only two passes through the data.

Through all the years Support Vector Machines have been successfully applied to solve such problems like face detection [38], time series prediction [6, 12, 60], text categorization [22], fuzzy systems learning [31], and many more.

SVMs have found many remarkable applications in bioinformatics and biometrics. Some prominent examples include protein classification [28], detection of the splice sites [50], or analysis of the gene expression profiles [29], including gene selection for microarray data, where a special type of SVM called the Potential SVM [19] has been successfully used for analysis of brain tumor data set [58], Lymphoma data set [57], and breast cancer data set [63].

Recently, the successful application of SVMs to analysis of proteomic spectra has been also achieved in our research group composed of computer scientists, bioinformaticians, and biostatisticians from the Faculty of Automatic Control, Electronics and Computer Science at Silesian University of Technology in a collaboration with clinicists from the Institute of Oncology in Gliwice, Poland. The detailed case study concerning this application is a subject discussed in Sect. 15.2. Here, let us give some introductory comments about the method used. An important issue in analyzing proteomic spectra is to find appropriate diagnostic biomarkers, which can help in early cancer detection. In order to find such biomarkers, it is necessary to extract and convert the mass spectra peaks. Moreover, the individual peaks are usually selected directly from the spectrum, but here they are modeled using Gaussian Mixture Decomposition. Correctness of the selected set of features can be verified using SVMs.

Such approach is presented in Sect. 15.2, which describes the experiments with data collected from a cohort of healthy people (a control group) and from patients suffering from the lung cancer. The proteomic spectra were obtained with MALDI-TOF method. The peaks were modeled to be composed of 100-900 (step=50) Gaussian components. Subsequently, two SVM's kernels (linear and Gaussian Radial Basis Function) and the different value of the box constraints for the soft margin (*bc* parameter)were investigated to find the most effective parameters. For the classification purposes 1–40 most differentiating properties derived from the Gaussian components were chosen based on t-Student test.

The validation was carried out using the method of multiple random cross-validation. Section 15.2 describes the whole cycle of classification, which is composed of: a) random selection of training sample, b) choice of the properties, c) learning the classifier, d) validation, e) error evaluation. Each cycle was repeated 400 times, and then the mean error was determined with its standard deviation. In the study three types of errors were determined, the total error, classification error in the control group, and the classification error in the group with cancer.

The second case study, which is in detail presented in Sect. 15.3, concerns face recognition. Face recognition is an approach which is one of the most regularly investigated biometric methods. Moreover, this is a biometric approach in which SVMs have found many interesting applications. As further explained in Sect. 15.3, automatic face recognition is a complex process that in general can be divided into four main stages, face detection, face normalization, feature extraction, and comparison of the feature vectors aimed at determining similarity between input facial images.

In practice, at every stage the classifiers such as SVM or artificial neural networks (ANN) can be applied to enhance the robustness of the process. The three most beneficial applications of SVM include: (1) Face detection, (2) Feature vectors comparison, and (3) Multi-method fusion. All these steps are demonstrated in Sect. 15.3. In particular, face detection, which is a sophisticated process composed of a quick preliminary selection of face candidates based on ellipse detection, followed by verification, is a subject of Sect. 15.3.4. In this approach, SVMs are used to determine whether a given image (actually a clipped region of the input image, normalized to fixed dimensions) is a face. Moreover, SVMs are used to improve the detection precision and the detected eye location is iteratively updated, to maximize the response.

Then, the feature vector comparison follows as illustrated in Sect. 15.3.5. This process depends on a specific feature extraction method and in many cases it is based on a distance metric defined in the feature space. However, the experimental results presented suggest that there is a potential for improvement using SVM. The classifier's task is to determine whether two given feature vectors (actually a vector being the difference between them) come from the same individual. This approach has been successfully applied for comparing the feature vectors obtained using different feature extraction methods, such as the Eigenfaces and elastic bunch graph matching.

Finally, the multi-method fusion is often beneficial when several different feature extraction methods are used. The similarity results obtained using individual feature extraction methods must be properly transformed into the final similarity measure. Here, SVMs take a similarity vector as an input and make the decision whether two input images were derived from the same individual.

In all aforementioned cases the representativeness of the training data must be of a particular concern. Large amounts of training data are often available, but not all of them are beneficial to finding the effective decision rules. This problem can be solved by randomly selecting the subsets of all available data to find the optimal. However, the genetic algorithms can be used here, as presented in Sect. 15.3.3 to optimize the training set, which occurred to be definitely more robust than the random selection.

15.2 Support Vector Machines Applied in the Classification of Mass Spectra

Clinical proteomic is an important field of proteomics which is the study of the proteome. All cells in an organizm have a common genome, but in the various types of cells, the different fragment of the genome is expressed. The proteome in contrast to the genome is dynamic and fluctuates depending on a combination of numerous internal and external factors. Therefore the changes in disease development can be monitored by proteome analysis. It has a great potential to contribute to medicine in terms of biomarker and therapeutic target discovery and also helps to detect diseases at their early stages [45,62].

The proteome is most frequently analyzed using mass spectrometry (MS) of the blood. The mass spectrometer returns the result in a very simple form of a spectrum which is represented by pairs of mass-to-charge ratios (m/z value) and corresponding intensity values. Then, the various methods of mathematical analysis are used to obtain considerable knowledge about the sample [1,40,73].

This fragment of chapter shows one of the methods for analyzing mass spectra. They are preprocessed, modeled as a Gaussian mixture and used in support vector machine (SVM) in order to distinguish two groups of people: those who suffer from lung cancer and a control (healthy) group. The data were delivered by the Department of Experimental and Clinical Radiobiology, Cancer Center and Institute of Oncology in Gliwice. Serum was isolated from blood taken from 49 healthy people and 38 people with lung cancer. Next, they were used in mass spectrometry. There were several obtained spectra for every person, because the serum isolation and subsequent MS analysis were repeated more than once for each individual [46].

15.2.1 MS Spectra Preprocessing

The process of gathering MS spectra involves many disturbances of biological and technical nature. As a result, the raw spectrum signal is noisy and must be preprocessed so as the spectra can be compared with each other. This process is illustrated in Fig. 1 and it consists of the following steps: 1) interpolation to common m/z axis, 2) removing outliers using Dixon test based on areas of the raw spectra, 3) averaging of technical repeats, 4) binning of neighboring points to reduce data complexity, 5) removal of the spectral area below baseline and 6) the total ion current (TIC) normalization which is a commonly adapted practice in the field [18,25,40].

The first step of the preprocessing was to standardize m/z axis. In the collected sample, the distance between each pair of successive m/z ratios was not constant, also it might be different for various samples. These differences were derived from that ion intensities, for each sample, was alone counted in

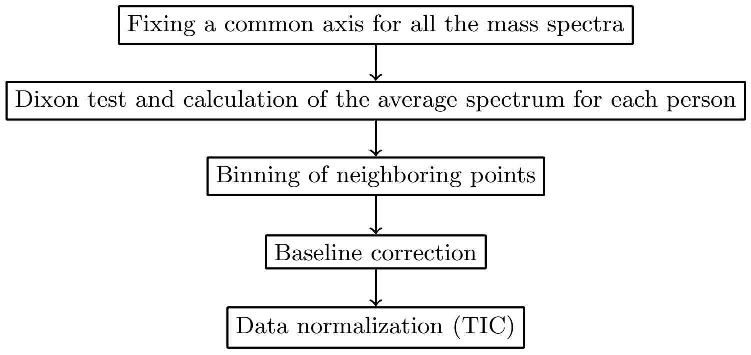

Fixing a common axis for all the mass spectra
Dixon test and calculation of the average spectrum for each person
Binning of neighboring points
Baseline correction
Data normalization (TIC)

Fig. 1 Data preprocessing

the MS detector. Because all spectra in the subsequent steps were processed together, a uniform scale was needed before the analysis was performed.

In order to eliminate accidental errors when isolating serum or during the MS process, these two steps of processes were repeated 4 times for the control group and 4–12 times for the group with cancer. In a set of repeated measurements one or more of the obtained values may differ considerably from the majority of the rest. In such a case it is important to eliminate those deviant values and not to include them in any subsequent calculation (e.g. the mean value or the standard deviation). For small repetitions, the Dixon's Q-test is used in order to detect outliers. This test is very simple with two limitations: the sample size should be greater than 3 and the population which is being sampled is assumed normal. Dixon's test checks significant differences between the extreme values (the smallest and the largest) and the rest of the sample [24].

Here the sample consist of a sum of intensity peaks and the sample size was a number of repetitions. Following this method, one repetition (spectrum) from 5 healthy people was rejected ($pval = 0.01$). Then, the average spectrum for each person was computed. In Fig. 2 an example of an average spectrum is shown. In this example, there was a single outlier, which had the largest value of the sum of intensity. As a result that outlier gave much bigger average value of peak intensities.

The MS detector is very sensitive to small changes of quantity of ions from an analizator. Therefore the detector returns many very detailed data. Their large amount would delay the computational processes, and therefore they were reduced by changing the discretization step. This was achieved by binning of the neighboring points. This process also renders a simple noise reduction and reduces the effects of minor observation errors. For analyzed mass spectra, 8 neighboring points were replaced with a single one, being their averaged intensity value.

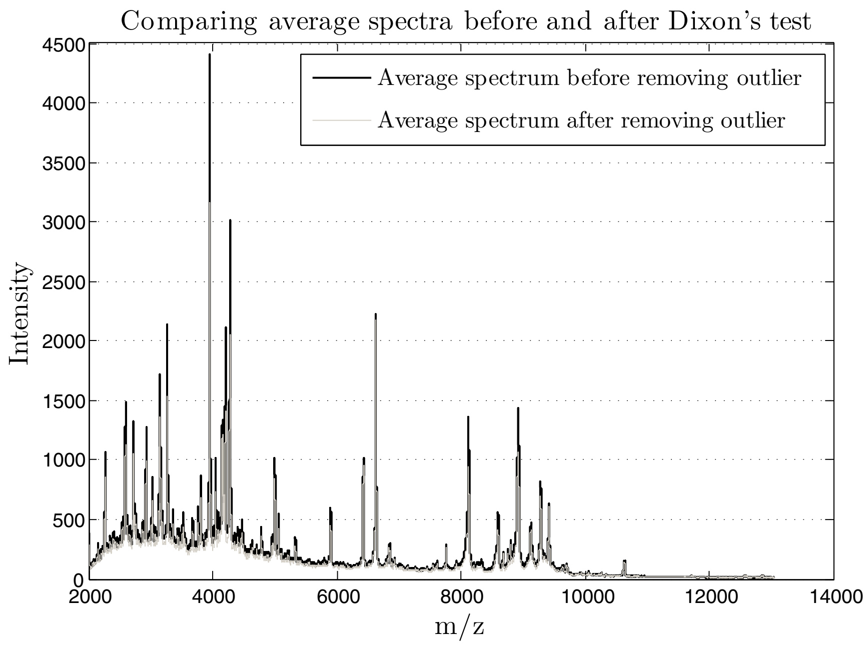


Fig. 2 Average spectrum before and after Dixon's test

One of the artifacts affecting the spectra is a shift of the signal, termed a spectrum baseline. It depends on the sample preparation, cocrystallization protein or peptide molecules with molecules of the matrix. Small molecular fragments from the process of degradation, desorption and collisions in the acceleration phase also produce background fluctuations. The noise in the spectrum consists of low frequency baseline. A general purpose of the baseline correction algorithm is to estimate (to find low frequency noise from the spectrum) and to remove the vast baseline artifact, in such a way that the shape line and peaks in the spectrum are not distorted. Without the baseline correction, the value of the baseline level could affect a considerable area of peaks causing difficulties in choosing the most significant peaks. In Fig. 3 examples of spectra before and after removing the baseline are shown.

There are some algorithms which remove the baseline, e.g. convex hull, moving average, spline, wavelet transform, monotone local minimum [33, 56, 66]. However the quality of baseline detection is difficult to assess.

Search of the 10-th local percentiles was used here. The percentiles estimate low frequency baseline, which is hidden among the high frequency noise and real signal peaks. The percentiles were fixed within multiple shifted windows of width 200 m/z and their values were assigned to the center of the window. The baseline was estimated using Piecewise Cubic Hermite Interpolation and it was removed afterwards.

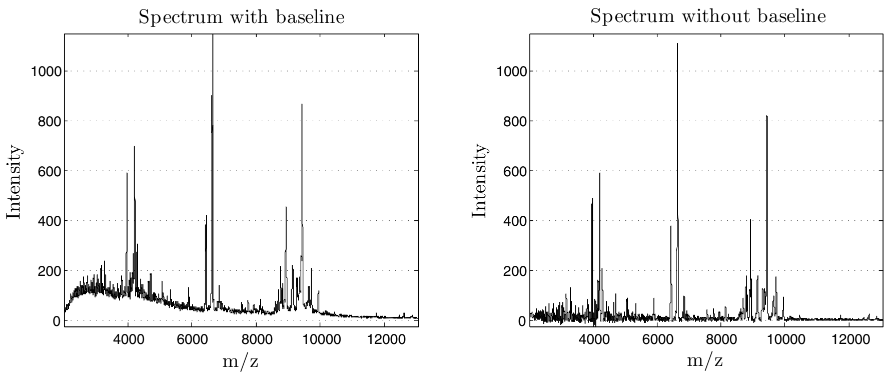


Fig. 3 The graph compares spectra before and after baseline correction

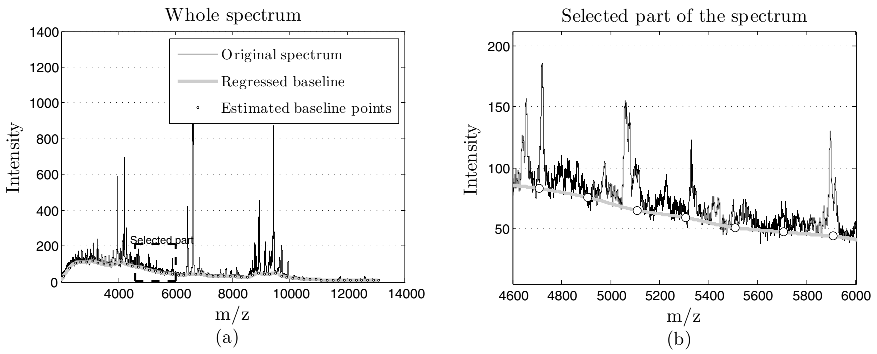


Fig. 4 The spectrum with the regressed baseline

In Fig. 3 an example of a spectrum is shown before and after baseline correction. The detailed assessment of the baseline is shown in Fig. 4. In (a) the whole spectrum after Dixon's test is presented along with the baseline regression based on the estimated baseline points. A selected part of the spectrum indicated in (a) is magnified in (b).

If many spectra are analyzed together, large differences in the intensities of the peaks may be essential. Unfortunately, the intensities of the peaks are very variable between experiments due to ionization efficiency or different total concentration of the substance. In order to eliminate this effect, the spectra are normalized. One of the method for normalizing the data is TIC (total ion current) method. Each peak intensity is divided by a ratio of the local intensity peaks sum to the mean intensity peaks sum for all spectra [25,36,45].

15.2.2 Preparing Spectra to Classification

In classification, the total number of measured data points is extremely large (about 10^4 after points binning). It is not possible to use whole spectra for classification, so feature selection is necessary. There are some existing methods for analyzing mass spectra after the preprocessing in order to select the features [35,40,72]. Most of the published algorithms detect and extract single peaks from the individual spectra. In this work, the average spectrum was modeled as a sum of Gaussian bell-shaped curves [45, 47]. The fitting is performed using Expectation-Maximization algorithm (EM) which maximizes the likelihood function [17,48]. In the work reported here, the Gaussian mixture density decomposition was used for 100, 150, ..., 850, 900 mixture components. In Fig. 5 a fragment of the average spectrum is shown along with its represent peaks by using Gaussian mixture decomposition with 3 different numbers of mixture components (K). The increasing number of components makes the model better fitted to the peaks. If the parameter K is too large, some components are unnecessary and noninformative – the model becomes

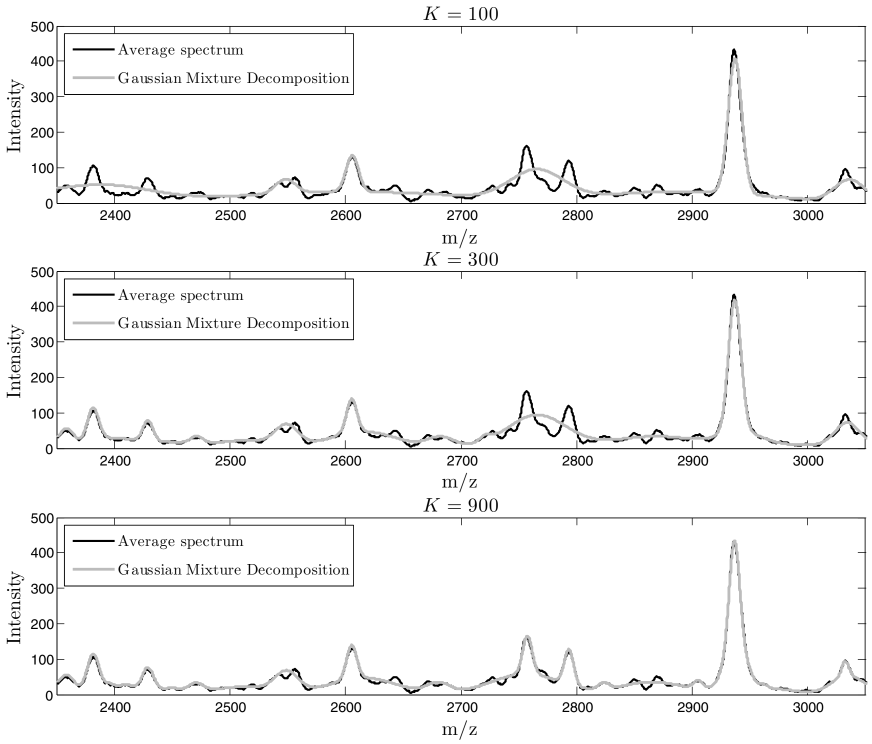


Fig. 5 Examples of average spectra fragment and gaussian mixture decomposition with 3 different numbers of mixture components (K)

overparametrized. On the other hand, when this parameter is small, several important peaks could be jointed together and the information is missing. Therefore the bayesian information criterion (BIC) was used to assess how the Gaussian mixture decomposition fits to the peaks [47,48]. This criterion introduces to a penalty for additional features. The log likelihood function with the BIC criterion takes the form:

$$\ell^{BIC} = \ell - \frac{1}{2}(3K-1)\ln\left(\sum_{n=1}^{N}\sum_{m-1}^{M} y_{m,n}\right), \tag{1}$$

where ℓ is the maximized value of the logarithm likelihood function for the estimated model, K is the number of components, $y_{m,n}$ is the n-th peak value for m-th person. The ℓ^{BIC} value is increasing along with the number of components up to about $K = 300$. For $K > 300$, ℓ^{BIC} started to be stabilized. Hence 300 mixture components were chosen for further analysis.

The Gaussian components were used to compute the feature vector (C) of spectra for each person. These spectral components were characterized by their intensities location on the m/z axis and standard deviation of a corresponding Gaussian:

$$C_m(k) = \sum_{n}^{N} \alpha_k y_{m,n} f(x_n;\mu_k,\sigma_k^2), \tag{2}$$

where α_k is the weight of a k-th component, $f(\cdot)$ is a probability density function of the x_n (m/z) value with μ (mean) and σ^2 parameters. In this case, $M = 87$ (number of people) input vectors are used for classification, each of them containing $K = 300$ (number of components) elements.

15.2.3 Classification

In this study, support vector machines (SVM) were used to classify the feature vectors obtained after Gaussian mixture decomposition. SVM classified each individual based on their feature vectors as being healthy or having lung cancer markers. It was investigated how the choice of the kernel and the box constraints for the soft margin influence the classification score obtained for different numbers of features.

To determine a classifier's quality it is necessary to determine a real probability of assigning a person to incorrect group. This value is unknown, therefore it has to be estimated experimentally using a validation set. This probability is estimated based on the percentage of wrong assignments within the whole testing sample. The total error (δ), as well as false negative rate (δ_{FN}) and false positive rate (δ_{FP}) are obtained as

$$\delta = \frac{FP+FN}{TP+TN+FP+FN} \quad \delta_{FP} = \frac{FP}{TN+FP} \quad \delta_{FN} = \frac{FN}{TP+FN} \tag{3}$$

where: TP is the number of sick people who were correctly diagnosed as sick, FP is the number of healthy people who were incorrectly diagnosed as sick, TN is the number of healthy people who were correctly diagnosed as healthy and FN is the number of healthy people who were incorrectly diagnosed as sick. Relationships among these terms are shown in Table 1.

Table 1 Accuracy of Diagnostic Test

		CONDITION	
		Disease	Non disease
TEST OUTCOME	Disease	True positive (TP)	False positive (FP)
	Non disease	False negative (FN)	True negative (TN)

The classification process, which is shown below, was repeated 400 times to estimate the average of 3 types of error rate (total error, false positive rate, false negative rate) and their standard deviation. The first step of classification was to select randomly the training and validation sets. The size of both sets was equal. Then, the features for the classification were selected based on t-test performed for the training set [73]. This test investigates the significance of the difference between the means of two groups [24]. For each feature, statistical significance ($pval$) was computed and the features with the smallest value of $pval$ were selected. Using selected features, the classifier was trained from the learning set and then it was tested on the validation set. The schemat of the classification proces is shown in Fig. 6.

A potential problem with using t-statistics is that this test can be used if the samples are normally distributed or if their number is sufficiently large. Thus the features were checked by the Lilliefors test which is used to test the null hypothesis that data come from a normally distributed population [32]. This test showed that 169 features in the control group and 185 features in the group with cancer had normal distribution ($pval = 0.01$). An example of this problem is shown in Fig. 7. When the data points introduce a curvature in the plot, an assumption of normality is not justified. In such a case the features in its original form could not be used in t-test. To correct the features, the Box-Cox transformation was used. This transformation transforms non-normally distributed data to a set of data that has approximately normal distribution [5]. The Box-Cox transformation is a family of power transformations. After it was applied, the number of features which did not have normal distribution ($pval = 0.01$) was reduced to only 1 in the control group and 12 in the group with cancer. An example of normal probability plot for feature with and without Box-Cox transformation is shown in Fig. 8. After this transformation the data points appear linear.

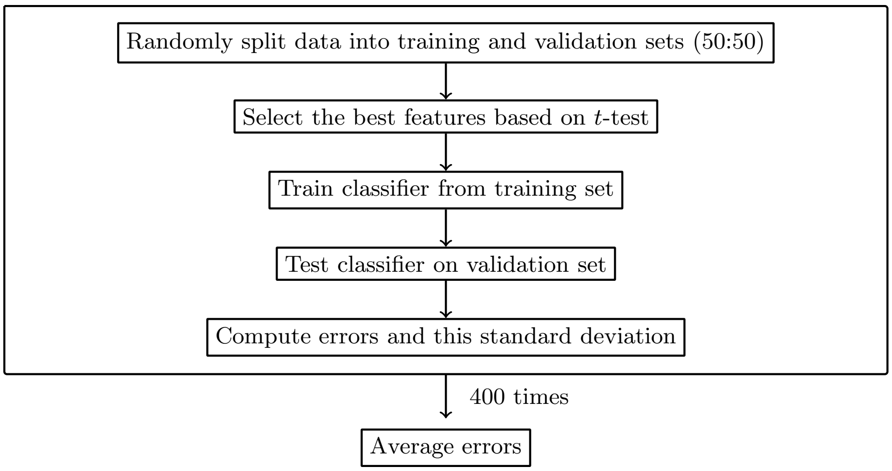

Randomly split data into training and validation sets (50:50)
Select the best features based on t-test
Train classifier from training set
Test classifier on validation set
Compute errors and this standard deviation
400 times
Average errors

Fig. 6 Classification process

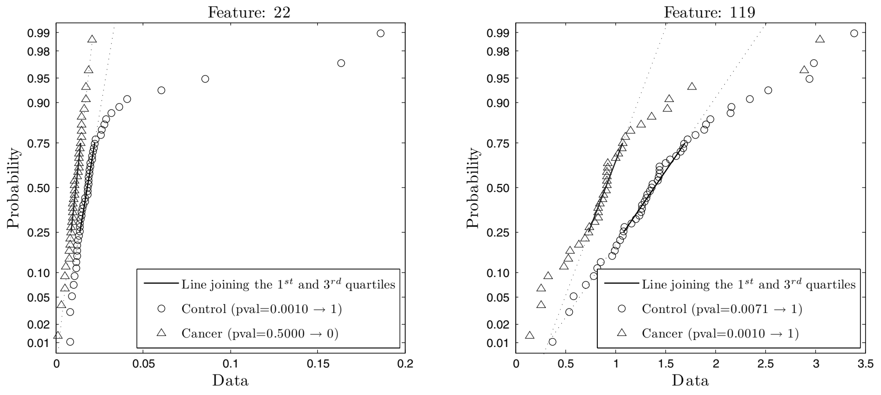


Fig. 7 The normal probability plot for the 22-th and 119-th features.

The Box-Cox transformation was used only to select the features using t-test and later the original features were classified. The feature indices were ordered by their ascending value of statistical significance and placed in a vector $F = (f_1, f_2, ..., f_{300})$. A vector F' contained the features indices which were used later in SVM. The number of features that are used in the classification cannot be greater than the size of the learning group. In this work, the training group contained 44 elements, so the maximum c value was set to 40. Here two methods were used to select features from the vector F.

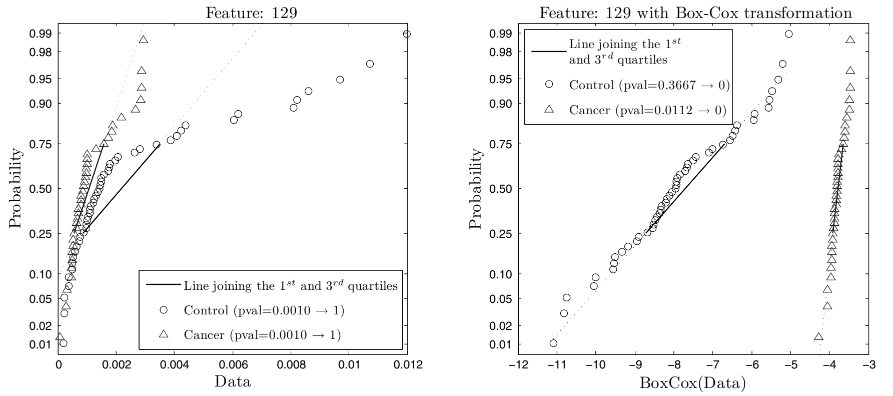
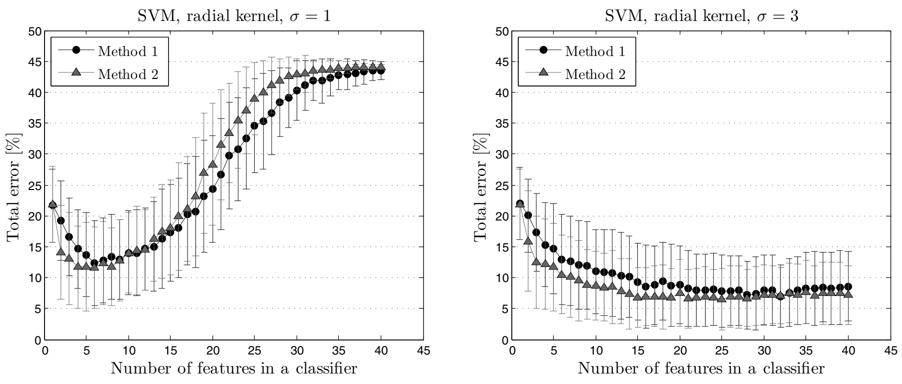


Fig. 8 The normal probability plot for the 129-th feature without and with Box-Cox transformation.

1. The 1-st method selected c first, features from the vector, so the consecutive vector of selected features was $F' = (f'_1, f'_2, ..., f'_c)$, where $f'_j = f_j$. The *pval* value for the features that are correlated with each other can be very similar and the correlated features do not contribute anything to the classification,
2. An alternative method was proposed in order to eliminate correlation between selected features. The correlation between neighboring features was checked using Pearson product-moment correlation coefficient [37]. The coefficient α, which determined if the features were closely connected, was set to 0.01. So, as a first feature always f_1 $(f'_1 = f_1)$ was taken, and then f_2 was verified if is correlated with f'_1. If not, it was accepted $(f'_2 = f_2)$, otherwise the next feature was checked until finding no relationship. After second feature selection, the subsequent correlations were tested only with the latest selected feature. To conclude, f'_{j+1} feature was checked whether it was correlated with f'_j. In this way, the best c features were chosen by checking only adjacent correlations.

In Fig. 9, the total error with standard deviation is shown, where different kernels and two methods of the feature selection are used. It can be seen in the chart, that regardless of the classifier parameters, the best results are obtained when the neighboring correlation is verified. The minimum total error is smaller and achieved using fewer features than if the features are selected in turn. Therefore it might be concluded that, when the feature correlation is checked, then similar features are not used in SVM, so a smaller number of features are needed to achieve minimum error.

In this study, not only was it checked how the number of features influences the error, but also how the box constraints for the soft margin (*bc* parameter) and the SVM kernel affect the classification score. Fig. 10 and Table 2 show how these parameters influence correctness of the classification. It might be



(a) SVM with radial kernel — σ=1 (b) SVM with radial kernel — σ=3

Fig. 9 The graph compares two methods of selecting features for SVM. The Method 1 is without checking the correlation between features and Method 2 is with checking the correlation.

noted that the parameter defining box constraints for the soft margin does not affect the final result significantly. Hence, it is better to take to use a large value of this parameter. In addition, the bigger this parameter is, the less support vectors we used in classifier (Fig. 11). A large number of support vectors may cause SVM be over-matched to the learning set. So there were no differences observed in the classification score for different values of the box constraints parameter, but they can affect the generalization capabilities.

With the σ value increase in SVM with the radial kernel, the minimum classification error decreases, but this minimum appears simultaneously at a larger number of features used for classification. Thus, the higher sigma (σ) is, the more features are needed to achieve the stability of the error. It is asymptotically convergent to a total error value obtained for random label assignment i.e. to $100\% \cdot 19/43 = 44.19\%$. The classification results for the radial SVM are getting closer to the results of the linear SVM with the increase of σ. The error in linear SVM does not increase when the number of parameters is bigger. This is due to the fact that adding new features to the classifier with a linear kernel does not contribute anything to the model, because the separating hyperplane is parallel to the dimensions defined by the irrelevant features.

Concluding from the results, the best number of features equals about 6–8. These values seem to be correct, if it is considered that the number of features should not exceed an order of magnitude less than a learning group size. In Table 3 the error values are shown for three different numbers of features. They do not vary much, but here the linear SVM is the best, not only in terms of the total error, but also for the false negative rate. This

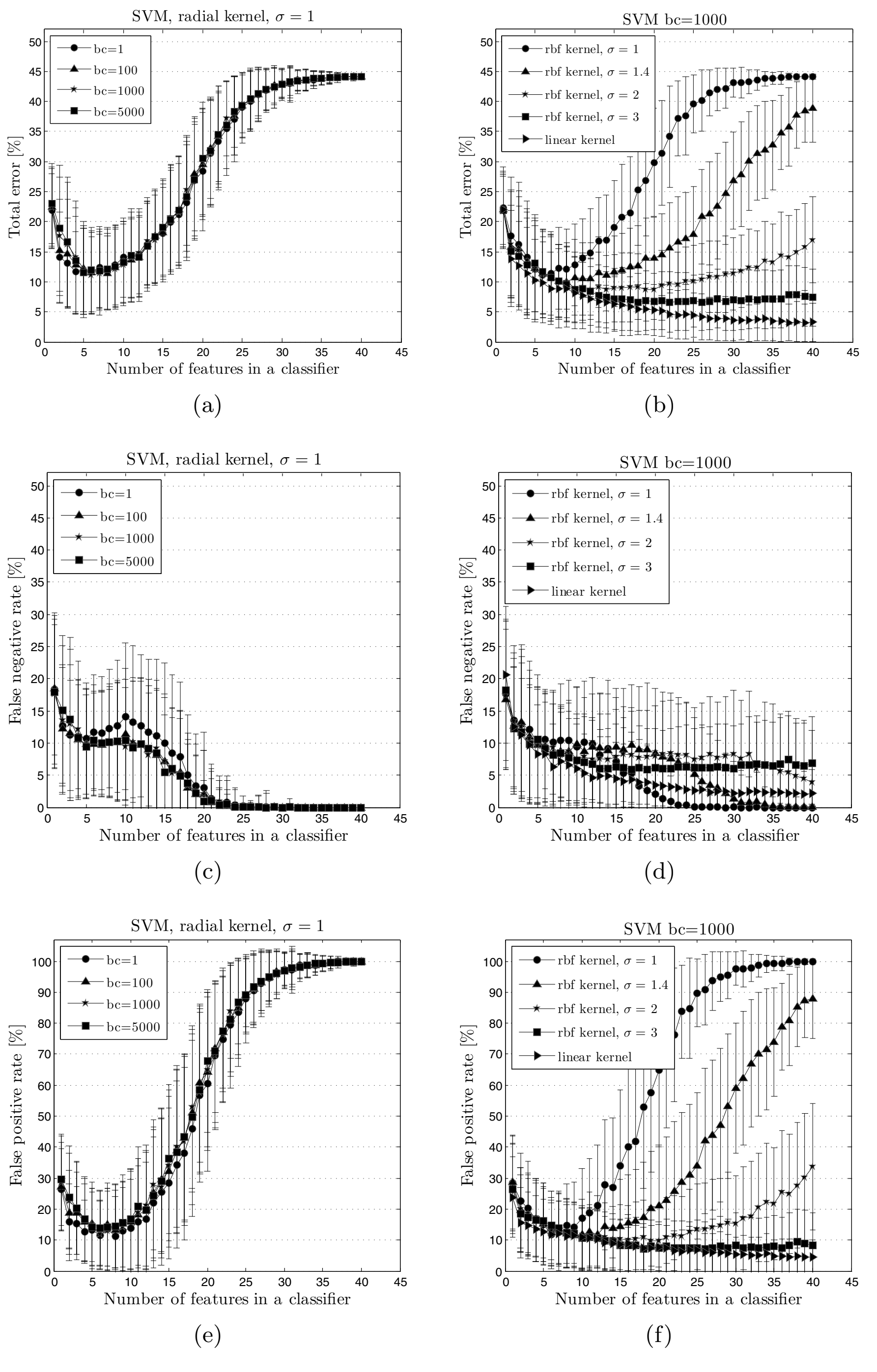


Fig. 10 Influence SVM parameter on quality of the classification

Table 2 Classification error for SVM with different parameter

bc	σ=1.0	σ=1.2	σ=1.4	σ=1.6	σ=1.8	σ=2.0	σ=2.6	σ=3.0	σ=3.6	linear
					Number of features for minimal total error					
1	6	6	7	4	9	11	18	16	20	40
100	8	6	7	10	12	11	21	25	29	37
500	6	8	10	10	11	18	17	29	26	40
1000	6	9	9	10	14	19	21	18	29	39
5000	5	7	8	11	15	14	20	24	27	40
					Total error [%]					
1	11.63	10.57	10.26	10.05	9.40	8.65	7.39	7.05	6.17	3.00
100	11.47	10.69	10.22	9.84	8.89	8.53	6.97	6.53	5.66	3.22
500	11.60	10.56	10.19	9.73	9.02	8.41	7.01	6.15	5.69	3.14
1000	11.06	10.66	9.96	9.49	8.99	8.71	7.13	6.59	5.64	3.19
5000	11.51	10.57	10.25	9.51	9.10	8.30	7.35	6.36	5.44	3.02
					FNR [%]					
1	11.64	10.50	9.82	9.26	9.41	9.14	7.93	7.22	6.61	2.21
100	10.06	9.21	8.33	8.49	7.54	7.71	6.44	5.70	5.69	2.16
500	10.38	9.42	8.56	8.50	7.80	7.17	6.50	6.20	5.55	2.11
1000	10.49	9.41	8.65	8.13	8.46	7.81	6.45	6.14	5.38	2.06
5000	9.42	9.15	8.71	7.81	7.64	7.57	6.76	5.66	5.16	1.95
					FPR [%]					
1	11.62	10.66	10.82	11.05	9.38	8.03	6.71	6.84	5.62	4.00
100	13.25	12.55	12.59	11.55	10.59	9.58	7.63	7.58	5.63	4.57
500	13.16	12.01	12.25	11.28	10.57	9.99	7.64	6.08	5.87	4.43
1000	11.79	12.25	11.62	11.22	9.67	9.84	8.00	7.16	5.97	4.61
5000	14.16	12.37	12.20	11.66	10.96	9.21	8.11	7.25	5.80	4.37

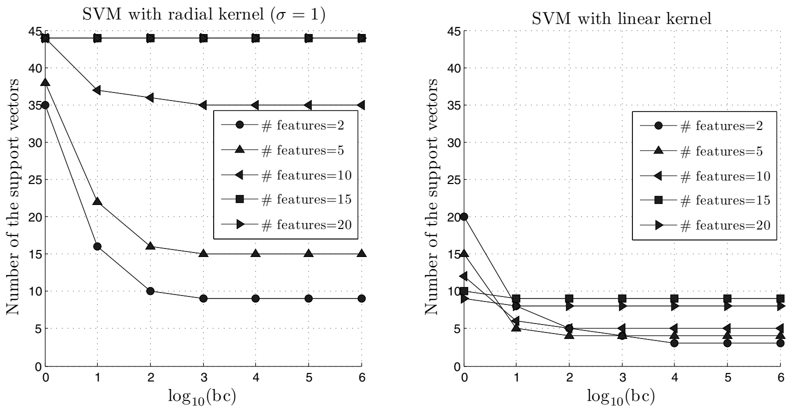


Fig. 11 Influence of number of selected features and box constraints for the soft margin (bc) on the number of support vectors which is used in SVM.

Table 3 The 3 types of errors [%] for 6, 7, 8 numbers of features (c)

c		σ=1.0	σ=1.2	σ=1.4	σ=1.6	σ=1.8	σ=2.0	σ=3.0	σ=3.6	linear
6	Tot. Error	11.94	11.33	10.86	11.55	10.36	11.85	11.37	11.92	9.71
	FNR	10.47	10.09	8.89	10.00	7.75	9.69	8.76	9.11	7.31
	FPR	13.79	12.89	13.36	13.51	13.66	14.59	14.67	15.47	12.74
7	Tot. Error	11.83	10.57	10.88	10.09	10.67	10.15	10.67	10.48	9.17
	FNR	9.95	9.15	9.01	8.14	9.13	8.48	8.26	8.42	7.11
	FPR	14.21	12.37	13.24	12.55	12.62	12.25	13.71	13.09	11.78
8	Tot. Error	12.05	10.98	10.25	10.22	10.30	10.01	10.03	10.66	8.80
	FNR	10.12	9.05	8.71	8.75	9.02	7.92	7.94	8.34	6.74
	FPR	14.49	13.42	12.20	12.08	11.91	12.66	12.67	13.58	11.39

is very important for the diagnostics, because it is essential to decrease the probability that a person with cancer is diagnosed as healthy.

15.3 Support Vector Machines Applied to Human Face Recognition

Face recognition [3, 14, 41, 42, 61, 68, 75] is one of the most investigated biometric methods which identify individuals on the basis of the human body features. Although face recognition systems have been extensively developed for the last 20 years, existing solutions perform definitely worse than humans do and below the expectations shared several years ago. This limited effectiveness is mainly due to the high variability of the human face resulting from expression variations and different lighting conditions [42]. Moreover, it is worth noting that the human face is one of the most important objects which we learn to analyze since the beginning of our lives. This is yet another reason why the existing algorithms cannot compete with the accuracy achieved by human observers. However, the main advantage of face recognition compared to alternative, often more reliable biometric methods, is a very low level of required interaction between the system and the individual who is being recognized. This allows face recognition systems to be used for various applications, ranging from entertainment to access control and surveillance tracking.

Computer vision still appears to be far from developing fully automatic face recognition systems, but during the last decade the algorithms improved significantly. This was confirmed by Face Recognition Vendor Tests (FRVT) [42] conducted by National Institute of Standards and Technology (NIST) in 2000, 2002 and 2006. Here, we present how face recognition algorithms can be improved using support vector machines. More specifically, we show three most beneficial applications that support the face recognition process at different stages.

15.3.1 Face Recognition Process

Automatic face recognition is a complex process that in general can be divided into four main steps illustrated in Fig. 12. These are: a) face detection, b) face normalization, c) facial feature extraction, and d) comparison and classification of the feature vectors.

Face detection [71, 74] determines whether any human faces are visible in the input image, and specifies their location. Location of frontal faces is usually defined by a pair of points representing the eye centers.

After that, every detected face is subject to normalization. This step includes geometric normalization (i.e. scaling, rotation and clipping) to a fixed size so that the eyes are always in the same position. Furthermore, the normalization may include more advanced operations aimed at reducing the impact of lighting conditions, occlusions, and facial expression changes. Here, we normalize the lighting variations by performing histogram equalization, which is a commonly adopted practice in face recognition [75].

After normalization, the facial image is processed to extract its distinctive features represented by a feature vector. There have been many methods developed for this task which adapt different approaches. In general, they can be divided into those that are holistic [2, 3, 61], using the entire face and those that make use of local features [68]. Feature extraction method should also define a similarity measure between the feature vectors in order to determine how similar any two given faces are. Sometimes multiple feature extraction methods are applied to boost the system performance. In such a case multiple similarities produced by different methods must be converted into a final similarity score.

The outlined core process makes it possible to complete four categories of the identification task [14], namely:

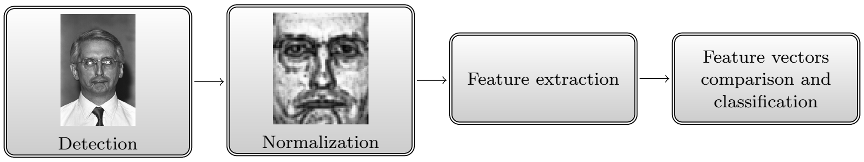


Fig. 12 Face recognition process.

1. Classification of a face image into one of a set of predefined classes (for face identification a single class contains images of the same individual).
2. Solving the known-unknown problem (i.e. determining whether an image belongs to one of the predefined classes or to none of them).
3. Verification (the image identity is claimed and the system determines if this statement is true or false).
4. Full identification which is a composition of the known-unknown and classification problems.

Face recognition algorithms can also be applied for solving related tasks such as gender classification [69], emotion or age estimation [67], etc. It is also worth noting that face detection is often considered as a separate task itself. At the current stage of development, detecting faces is much more reliable than identifying individuals and it has already found first successful applications in the industry, e.g. in the digital cameras.

At every stage of the face recognition process classifiers such as support vectors machines or artificial neural networks can be applied to improve the performance. In this chapter three examples of most beneficial SVM's applications are presented and discussed:

1. *Verifier in face detection.* Face detection is usually a sophisticated process itself. We adapted technique composed of a quick preliminary selection of face candidates based on ellipse detection, followed by the verification step [26]. Here, SVM serves as a verifier which determines whether a given image (actually a clipped region of the input image, normalized to fixed dimensions) is a face. This is further explained in Sect. 15.3.4.
2. *Feature vector comparison* depends on a specific feature extraction method, and in many cases is based on a distance metric defined in the feature space. However, the experimental results have confirmed that there is a potential for improvement using SVM. The classifier's task is to determine whether two given feature vectors (actually a vector being the difference between them) have been derived from the same individual. This approach, presented in Sect. 15.3.5, has been successfully applied for comparing the feature vectors obtained using the Eigenfaces [61] feature extraction method.
3. *Multi-method fusion* is often beneficial when several different feature extraction methods are used. The similarity results obtained using individual feature extraction methods must be transformed into the final similarity measure. Here, SVM takes a similarity vector as an input and makes the decision whether two input images were derived from the same individual. This application is described in Sect. 15.3.6.

15.3.2 Evaluation Protocol

Face recognition algorithms were evaluated on the basis of the classification task specified earlier in this chapter, following protocol described in [15].

The validation set was split into a gallery (G) which contains one image per individual, and a probe set (Q) for which the classification is performed. Classification score of a single sample $q \in Q$ is concerned with a term of *rank* (r). It is an integer ranging from 1 to the cardinality of the gallery. A sample is recognized with n-th rank when there are n samples in the gallery which similarity (S) to q is greater than or equal to the similarity between q and a sample in the gallery g_k, which class identifier $\mathrm{id}(g_k)$ is identical to an identifier of q:

$$r(q) = \sharp \left\{ (g_i, g_k) \in G : S(g_i, q) \geq S(g_k, q), \mathrm{id(g_k)} = \mathrm{id(q)} \right\} . \tag{4}$$

Hence, if a sample is classified correctly, its classification rank $r = 1$. Classification effectiveness for a given set is a percentage of correctly classified images. Moreover, face recognition performance is also presented in a form of cumulative match curves (CMC) which show the classification score obtained for subsequent ranks.

In cases when the feature extraction process was evaluated (Sects. 15.3.5 and 15.3.6), in order to prevent the propagation of face detection errors, we used hand-labeled eye locations during the geometrical normalization.

Face detection score used in Sect. 15.3.4 was evaluated on the basis of detection precision. In literature the detection score is often expressed as the detection rate and false positive rate. The first one is a percentage of faces that were correctly detected in the analyzed set of images, while the latter is a percentage of non-faces within the detected faces. This is a proper metric as long as the aim is to count the faces in images or give a rough approximation of face location. However, for face analysis purposes not only does it matter whether the face was found, but how precisely it was located. If the detection precision error may be propagated further, the detector's evaluation should take the precision into account. In the presented approach it was achieved by calculating a relative detection error (δ_d) for a single face as

$$\delta_d = \frac{\Delta_l + \Delta_r}{2D} , \tag{5}$$

where Δ_l and Δ_r are the distances in pixels between real and detected positions for the left and right eye, and D is the distance between real positions of the eyes. For false negatives and false positives the relative error is set to 1, and an average relative error computed for the entire test set is considered as a final detection score.

15.3.3 Selecting SVM Training Set

In all of the aforementioned SVM applications the training data are generated in automatic or semi-automatic way. This means that huge amounts of training data are available, but their representativeness and correctness cannot be guaranteed. Not all of the data in the original training sets are beneficial to finding the effective decision rules. The dependence between the training set size and training time was extensively studied [8,23,55] and different methods have been proposed to decrease the training time. However, it has been also shown that the number of support vectors grows linearly with the training set size [59]. This means that using entire training data sets may affect not only the training time, but the classification time and effectiveness as well.

There exists a possibility that in case of huge training data sets, their subsets are more representative than the whole set, especially if the data were acquired in an automatic, uncontrolled way. It is worth noting that in practice SVM training is identical to selecting a small subset of the whole training set. These selected vectors are termed the support vectors and are used for the classification. Hence, it may be concluded that only a small part of the original training set is relevant for training. Reduced support vector machines [30] benefit from this observation by selecting a subset of the training set randomly. Here we outline how the training set can be optimized using genetic algorithms. This delivers much better results than using the entire training data set and is definitely more robust than selecting the subset randomly.

15.3.3.1 Genetic Algorithms

Genetic algorithms [13] (GA) implement a heuristic approach which supports the search for solutions close to the optimal using an evolutional strategy. At first, a population of individuals characterized by their genotypes is generated and it is later transformed using three genetic operators: selection, mutation and crossover. A genotype contains a set of parameters to be optimized and defines unambiguously a single solution.

The population $\boldsymbol{P}_0$ is initialized with a set of individuals $\{p_i^{(0)}\}$, whose genotypes are generated randomly. This creates the first generation. After that, every individual is assessed and its fitness $\eta(p)$ is determined. A subsequent generation is obtained by transforming the current population using three operators:

1. *Selection.* This operation rejects the individuals with low fitness, and those which remain are further processed using two subsequent operators.
2. *Mutation* is a random modification to the genotype of an individual and it is aimed at sustaining the genotype variety. Otherwise there would be a risk that the individuals within every generation become too similar to each other. This would result in finding a local minimum of the optimization problem.
3. *Crossover* operates on two individuals whose genotypes are randomly merged into a new one. This creates a subsequent generation of individuals.

In general it is expected that the maximal fitness increases between subsequent generations. This process is iteratively repeated to obtain an optimal solution (i.e. a sufficiently well-fitted individual). A stop condition may vary from case to case.

Genetic algorithms are particularly useful in those areas where standard optimization techniques cannot be applied, but the effectiveness of a single solution can be determined easily and clearly. The parameters like population size, initialization method and stop condition, as well as the detailed operator's rules must be defined according to specific conditions of the optimization problem.

15.3.3.2 Training Set Optimization Using Genetic Algorithms

We found genetic algorithms particularly useful for optimizing the SVM training sets in problems related to face recognition. Here, cardinality of the training data sets is usually large, but quality of the data is low. Sometimes the data cannot be labeled with high confidence which will be explained later for the specific cases. Hence, the process of training subset selection occurs crucial to the final effectiveness rendered by SVM.

We have utilized genetic approach as follows. A genotype of a single individual defines the content of the training data. An individual is understood as a subset of the entire training set. Number of vectors in each class is constant for every individual, so the training set size is a fixed parameter.

Fitness of each individual is determined based on the classification effectiveness measured for the validation set independent from the training set. If the classifier is used to solve a specific task which can be unambiguously evaluated, the evaluation score is treated as the fitness (e.g. face detection score). The individuals with the highest fitness are selected to create the subsequent generation. This process of creating an individual and determining its fitness is illustrated in Fig. 13.

Mutation is performed by random changes to the training subsets of the individual. Some vectors in the genotype are randomly substituted with others from the original training set. Mutation parameter ξ defines the percentage of the vectors in every genotype which are modified. For all of the investigated cases we used $\xi = 10\%$.

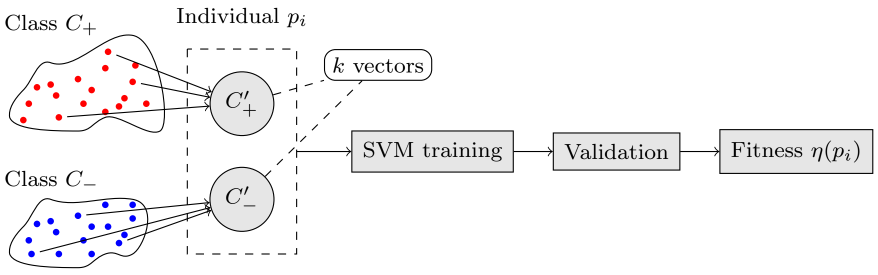


Fig. 13 Process of selecting and evaluating an individual using genetic algorithm.

In order to obtain a crossover of two individuals, the sets defined by their genotypes are at first summed within each class and then k vectors are selected in each class to form a new individual.

The described process is iteratively repeated as long as the maximal effectiveness is increasing in subsequent generations or until the effectiveness reaches a desired level.

In the presented cases we used labeled training data sets and we did not allow the vectors to change their labels (a vector $v \in C_+$ cannot be selected to C_- during the optimization). However, it is worth noting that this procedure can also be adapted for cases where the training data are unlabeled or label changing may be allowed due to soft labeling.

Examples presented later in this chapter demonstrate that following this approach, it is much easier to determine a proper subset of the training set. Theoretically, the same subset can be found by randomized or brute-force search, but in case of huge training sets it may be hardly feasible.

15.3.4 Face Detection

As it was stated earlier in this chapter, face detection is often considered as a separate task which has found a broad range of applications, including the surveillance tracking, human-computer interfaces and entertainment purposes. In this context it is regarded as the first, but a very important step of automatic face recognition [41,75]. Here not only is it important to give a rough approximation of face location (which is usually sufficient for most of the applications), but the faces must be located with high precision. This is essential for proper image normalization. If the eyes are imprecisely located, then after normalization, the faces may be tilted or misplaced, which usually makes further analysis ineffective. The presented experimental validation demonstrates that the face detection precision has critical influence on the face recognition score.

An extensive survey on the existing face detection methods is presented in [74]. Among the most popular and robust algorithms, the majority of them utilizes classifiers such as SVM or neural networks. They work by learning face appearance from a representative training set to make it possible to determine whether a given image region contains a face [39,52,61]. This is often effective, but the whole image must be analyzed with a varying size of the scanning window, which seriously affects the performance and limits real-time applications. Such basic detectors have been optimized using cascades of classifiers [64] which speed up the searching process significantly.

Here, we focus on a double-level approach [26], in which the face candidates are first selected using ellipse detection and then verified by SVM classifier. This makes it possible to detect faces quickly and precisely, which is essential for face recognition applications. This approach is outlined in Fig. 14.

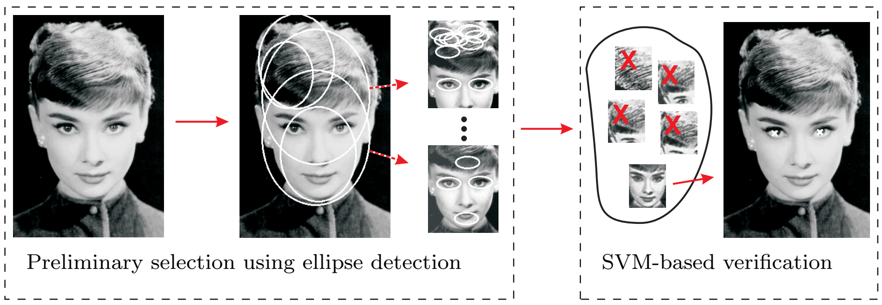


Fig. 14 Double-level face detection process.

In the work presented here the preliminary selection is realized using an ellipse detector. We benefit from the observation that human heads can be usually represented by vertically-oriented ellipses. Furthermore, the eye sockets also are characterized by horizontally-oriented elliptical shapes. We have adapted an ellipse detector proposed by Maio and Maltoni [34] for face localization. In their approach ellipses are detected in a directional image with generalized Hough transform to find approximate face location. After that, we repeat ellipse detection inside the primarily detected ellipses to find candidates for eye sockets. The eye-socket candidates are paired using simple heuristic rules to obtain face candidates which are normalized and are subject to verification. More details of the ellipse detection procedure are explained in our earlier work [26].

After the candidates have been detected, they are normalized and verified to reject all of the false cases and accept exclusively facial images. The normalization includes histogram equalization and scaling to a fixed size of 16×19 pixels in such a way, that the eyes are in fixed positions afterwards. Eye positions are determined as centers of the eye-socket ellipses. Subsequent rows of a normalized image are stacked to form a 304-dimensional vector that is further processed by SVM.

SVM is trained with two classes of vectors representing a) real faces and b) false face candidates. Here, we used RBF kernel with $\sigma = 0.7$ on the basis that it delivered the best results. After training, the classifier is capable of determining whether a candidate is a face or not. Furthermore, in order to increase the detection precision we modify the original eye locations with a following iterative procedure. At every i-th step of the procedure the current eye position $E^{(i)}$ is modified in eight directions by a step s_i. For every modified eye position, a face image is normalized and verified. If the verifier's response for a position $E_n^{(i)}$ (where $n \in \{1, 2, ..., 8\}$ determines the direction) is greater than the response in the current position $E^{(i)}$, it is moved to the new position ($E^{(i+1)} = E_n^{(i)}$). If the current position gives the maximal response, the step is decreased ($s_{i+1} = s_i/2$), and the position remains unchanged ($E^{(i+1)} = E^{(i)}$). The process, executed for every eye separately, is started with a step $s_0 = 4$ and repeated until it equals 1. As a result, the eye positions are moved to a local maximum of the verifier's response. This position is closer to the real location than a central point of the eye-socket ellipse.

The speed of this double-level face detector is dependent on the minimal size of heads which are expected to be detected and does not depend strongly on the input image size. Assuming that the smallest size of the faces which are supposed to be detected is such that their height is at least 20% of the largest input image dimension, the detector can work at 20 frames per second. This is definitely sufficient for real-time applications, and usually it is not expected to detect smaller faces for recognition purposes (they would be too small to be recognized correctly).

In Fig. 15 some examples of the images belonging to face and non-face classes are presented. These candidates were extracted from a BioID face database [21] which contains 1521 gray level images presenting frontal faces on a complex background. Ellipse-detection-based preliminary selection extracted 3612 face candidates which were classified by an expert to form the original training set. It may be observed from the figure that although in majority of cases the labeling may be done with high confidence, some candidates present faces which are not centrally located. They were assigned to the non-face class. Such examples may be very helpful to reject imprecisely located faces during the iterative precision improvement procedure. However, sometimes an expert may not be certain whether such an image would improve or deteriorate the detection score. It is presented later in this section than GA-based training set optimization is therefore very helpful here.

Experimental validation was conducted using 3657 frontal face images from Feret database [43]. Some examples of them are presented in Fig. 16. Also, face recognition performance was evaluated for different face detection scores in order to demonstrate the effect of face detection error propagation.

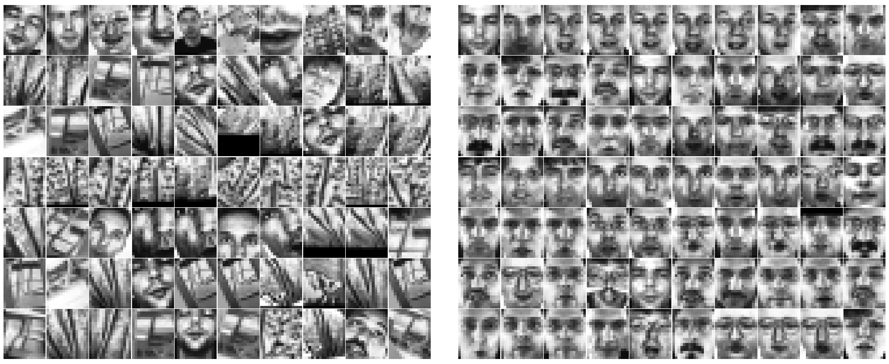


Fig. 15 False (a) and true (b) face candidates for SVM training.

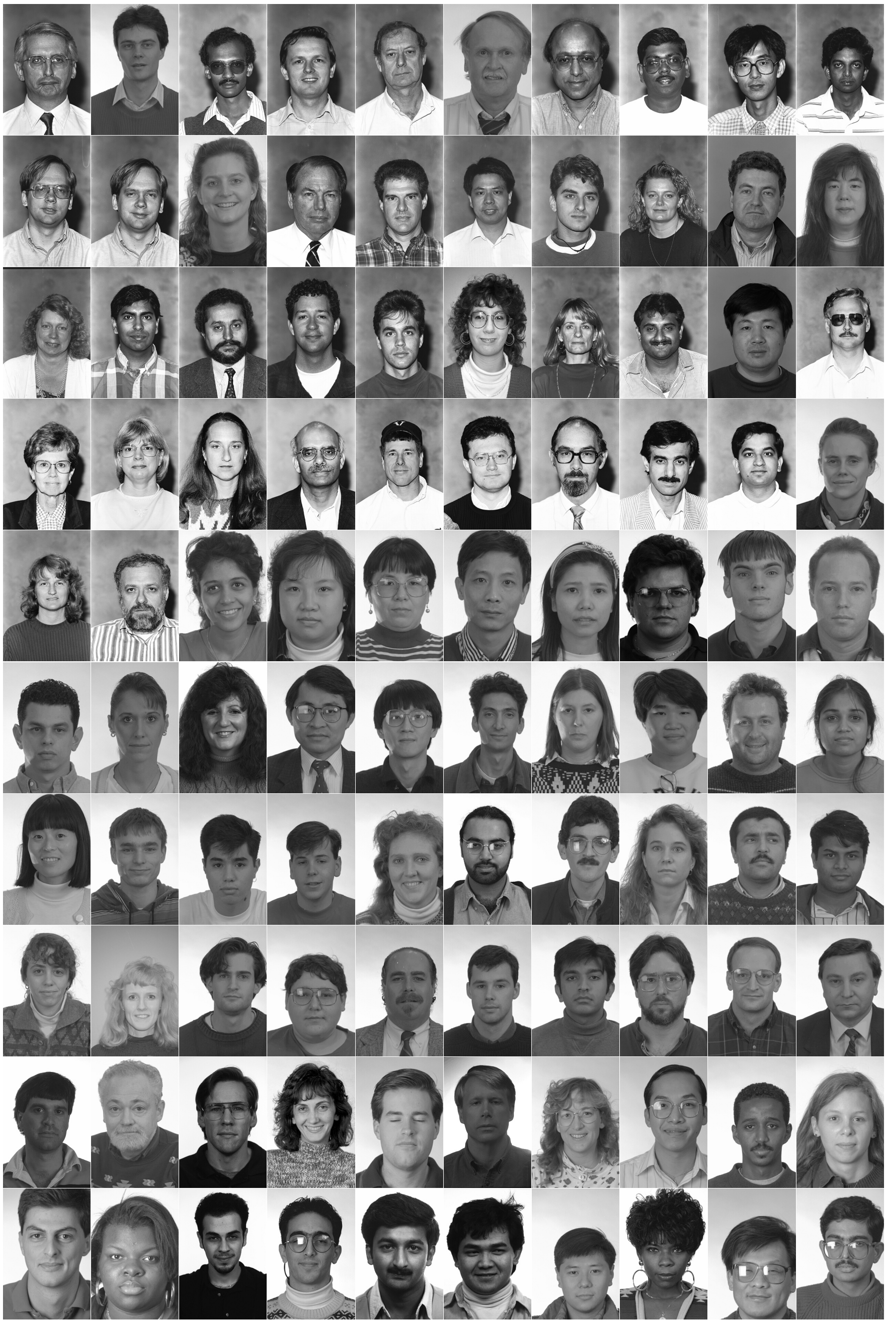


Fig. 16 Examples of images from Feret face database [43].

In this example we have compared three alternative approaches to SVM training:

1. SVM trained with all available images (entire training set used).
2. SVM trained with randomly chosen subsets of the entire training set. Every subset was formed out of 200 facial images and 200 non-facial images which examples are presented in Fig. 15.
3. Training set optimization using genetic algorithm as outlined earlier in Sect. 15.3.3.

Comparison of the detection error obtained for these three cases are demonstrated in Fig. 17. In the figure we present the smallest detection error obtained at subsequent generations of the GA-based optimization procedure, averaged over 10 different runs. Error bars indicate the standard deviation between the score obtained at different runs.

Evaluation of a single individual involves SVM training and running face detection for all of the images in the evaluation set. This is a time-consuming operation, so in our experiments we limited the size of the population to 10 individuals. Despite this limitation the obtained results are much better than using random selection of the training subset, and stable result is achieved after 10-15 generations. Moreover, it may be observed from the standard deviation that the final score was similar in all runs. For fair comparison we have performed 200 different random selections which equals the number of tests executed during a single GA run. The best result obtained with the random selection is presented in the figure. Taking a subset of the original training set is superior to using the entire data set for SVM training (the obtained score is also marked in the figure).

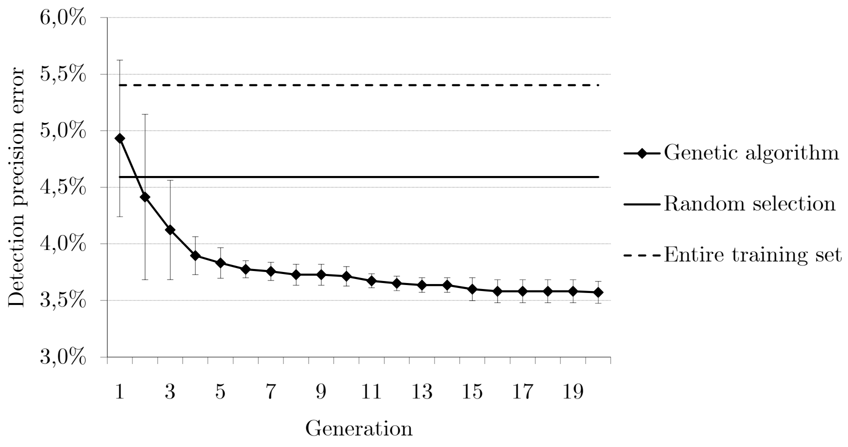


Fig. 17 Detection precision error in subsequent generations of training sample optimization.

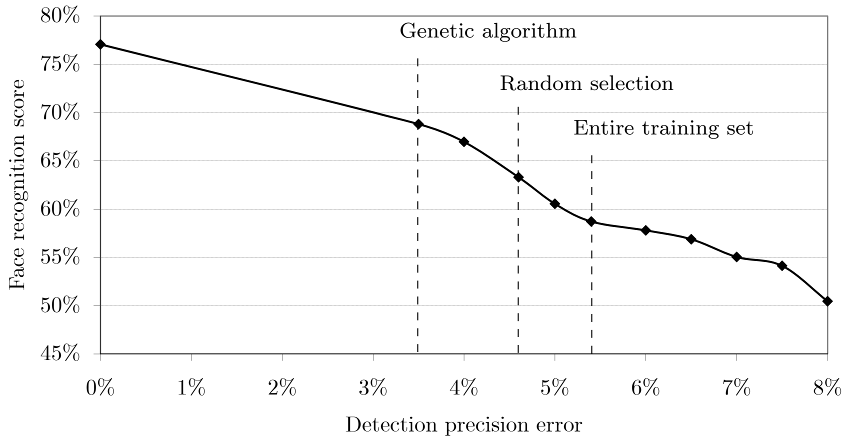


Fig. 18 Dependence between detection precision error and face recognition score.

The differences between the errors obtained using the investigated methods may appear small, but they have large impact on face recognition performance. The dependence between face detection precision error and face recognition score is illustrated in Fig. 18. In this case the Eigenfaces [61] feature extraction method was used. The detection error equal to 0% means the real eye positions pointed by an expert. It can be seen from the figure that the difference in detection precision error between the GA-based optimization and random selection equals over 5%, and it grows to as much as 10% compared to SVM trained with the entire training set. This demonstrates how important the training subset optimization is in case of face verification.

15.3.5 Feature Vectors Comparison

Second application of SVM to automatic face recognition is concerned with comparing feature vectors obtained using the Eigenfaces method [61]. This method was one of the first attempts to face recognition and is based on principal component analysis (PCA). It has many drawbacks which limit its effectiveness [27], but it has been widely investigated and is frequently used for research purposes.

The method requires a training stage which is performed for M normalized face images. Every image contains N pixels and is treated as an N-dimensional vector. First, a covariance matrix $\boldsymbol{C}$ is computed:

$$\boldsymbol{C} = \frac{1}{M} \sum_{i=1}^{M} \left(\mathbf{x}_i - \boldsymbol{\mu}\right) \left(\mathbf{x}_i - \boldsymbol{\mu}\right)^T , \qquad (6)$$

where $\mathbf{x}_i$ is an i-th normalized face image vector, and $\boldsymbol{\mu}$ is an average face vector. Subsequently, the covariance matrix is subjected to the eigen decomposition $\boldsymbol{C} = \boldsymbol{\Phi}\boldsymbol{\Lambda}\boldsymbol{\Phi}^T$ where $\boldsymbol{\Lambda} = diag\,(\lambda_1, \ldots, \lambda_N)$ is the matrix with the ordered eigenvalues along the diagonal and $\boldsymbol{\Phi} = [\boldsymbol{v}_1| \ldots |\boldsymbol{v}_N]$ is the matrix with the correspondingly ordered eigenvectors as columns. A face space has far less dimensions than the input space, and a relatively small number $(n << N)$ of the eigenvectors with the highest eigenvalues create an orthogonal basis for the n-dimensional face space. Here, the dimensionality was reduced to $n = 150$.

After training, features can be extracted from any normalized face image, utilizing n eigenvectors with the highest eigenvalues. The feature extraction is performed by projecting a normalized face image onto the face space:

$$\boldsymbol{\nu} = \boldsymbol{\Phi}^T(\mathbf{x} - \boldsymbol{\mu}), \tag{7}$$

where $\boldsymbol{\nu}$ is the feature vector and $\mathbf{x}$ is a normalized face image. Following the conventional approach, the similarity between the feature vectors is computed on the basis of their Euclidean or Mahalanobis distance in the feature space. Alternative distance metrics can also be applied for this purpose, which has been studied in [70].

Classifiers are often used to determine identity based on the feature vector without computing similarity between them. In such a case every class of this classifier represents a single individual [20]. This requires a multi-class classifier be learned with feature vectors extracted from people whose images are intended to be recognized. Although SVM is a two-class learning machine, it can be used here to distinguish each class from all of the remaining samples. Hence, for K individuals, K classifiers must be learned. Such a solution brings good results, but two serious limitations must be considered: 1) many feature vectors are required for every person whose face is to be enrolled to the system database, and 2) adding a new person to the gallery requires retraining all of the classifiers. This limits the system scalability and stands in contrast with the conventional approach where a single image is sufficient to register a new individual (the only action is to add the extracted feature vector to the gallery).

Here, we adapted another approach [44] which reduces the aforementioned K class problem to a two-class one. A pair of feature vectors $\boldsymbol{\nu}_1$ and $\boldsymbol{\nu}_2$ are subtracted from each other to obtain a difference vector $(\boldsymbol{\nu}_\Delta)$ which is processed by SVM. Every i-th element of the difference vector is obtained as

$$\nu_{\Delta i} = |\nu_{1i} - \nu_{2i}| \,. \tag{8}$$

Two classes of such feature vector pairs can be distinguished, namely: a) intra-class pairs $C_W = \{(\boldsymbol{\nu}_1, \boldsymbol{\nu}_2) : \mathrm{id}(\boldsymbol{\nu}_1) = \mathrm{id}(\boldsymbol{\nu}_2)\}$ and b) inter-class pairs $C_B = \{(\boldsymbol{\nu}_1, \boldsymbol{\nu}_2) : \mathrm{id}(\boldsymbol{\nu}_1) \neq \mathrm{id}(\boldsymbol{\nu}_2)\}$. Hence, SVM determines whether a given pair of feature vectors have been extracted from the same individual or from two different persons. Following this approach, the classifier can be learned only once based on two sets of pairs created from a training set, and adding new vectors does not require SVM be retrained. Also, a single image is sufficient to register a new person to the gallery.

The main problem is concerned with training set representativeness and number of available samples in each class. For a training set containing M images of K individuals (K_i images for every i-th individual), there are $\sharp C_B = \sum_i K_i(K_i - 1)/2$ intra-class pairs and $\sharp C_W = M(M-1)/2 - \sharp C_B$ inter-class pairs. This means that for a large number of classes there will be much more intra-class pairs than inter-class ones ($\sharp C_B << \sharp C_W$). Hence, training set reduction is essential here, which can be supported using GA-based optimization.

For training we used 761 face images for 275 individuals from the FRGC Version 1 database [41]. From this set 286066 inter-class and 3114 intra-class pairs were extracted. Every pair was transformed into a difference vector (8) that can be processed by SVM. Two kernel functions have been used: 1) 3^{rd} degree polynomial and 2) linear kernel. RBF kernel was not effective for this application. This can be explained by a linear nature of the problem (the longer the difference vector is, the bigger chance that it is an inter-class difference).

In this case it was virtually impossible to train SVM with all of the available vectors due to large and unequal cardinality of the opposite classes in the training set. Therefore, we trained SVM with 100 intra-class and 100 inter-class difference vectors selected from the entire training set. Similarly to the face verification case, the training samples were selected in two ways: randomly and using genetic algorithm. The number of single classification tests was equal in both cases. For GA optimization the fitness was determined based on the first-rank classification score obtained for the validation set. Population size was 10 and the presented results were averaged over 10 independent runs. Classification score achieved after subsequent generations is presented in Fig. 19 for the polynomial kernel and in Fig. 20 for the linear kernel. GA training set optimization delivered much better results than those achieved using random selection.

The best scores obtained for the polynomial and linear kernels are compared in Fig. 21 in a form of CMC curves. The achieved classification scores are compared with the effectiveness obtained following the conventional methods in Table 4. It can be seen from the table that using SVM trained with GA-optimized set, the face recognition score is higher than that obtained using Euclidean or Mahalanobis distances. GA-based optimization made it possible to find a representative training set which reduced the classification error significantly. It can also be observed that although the best results achieved using two alternative kernels are very close to each other, it is the linear kernel which is less sensitive to the selected subset. While the gain of using GA over random selection is almost 8% for the polynomial kernel, it is just 2.6% for the linear one. GA helps to find a better subset, but the gain is not that large here.

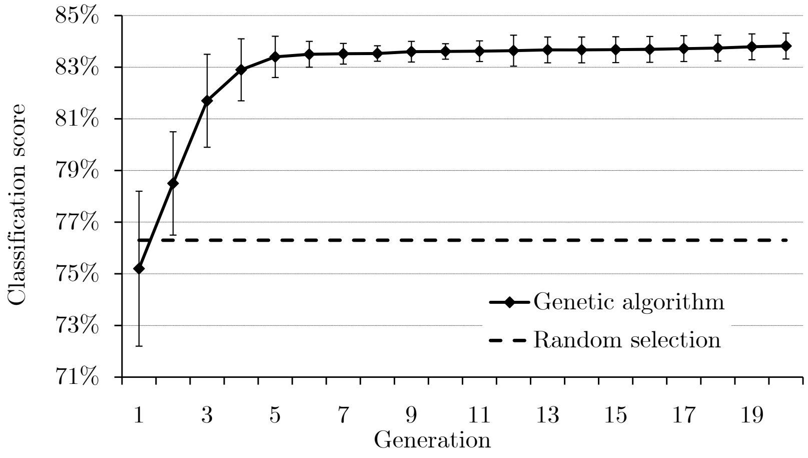


Fig. 19 First-rank classification score in subsequent generations for SVM with polynomial kernel.

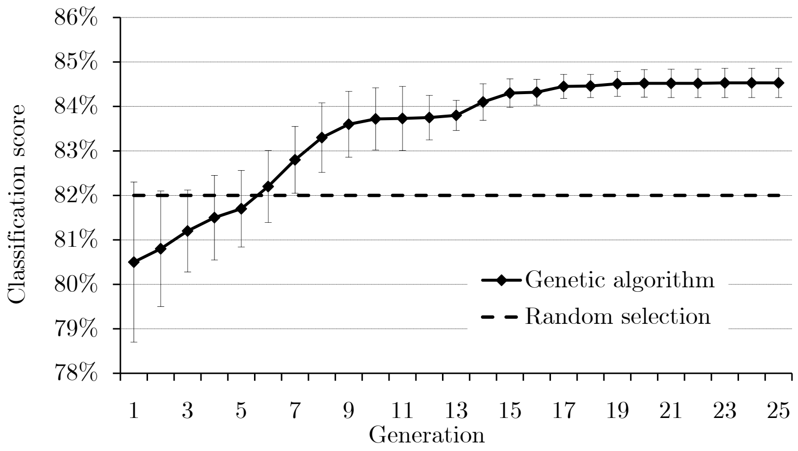


Fig. 20 First-rank classification score in subsequent generations for SVM with linear kernel.

Table 4 First-rank classification score for various feature vector comparison methods.

SVM-based comparison		
Kernel type	Random selection	Genetic algorithm
Polynomial	76.3 %	84.0 %
Linear	82.0 %	84.6 %
Euclidean-distance-based comparison		77.1%
Mahalanobis-distance-based comparison		79.2%

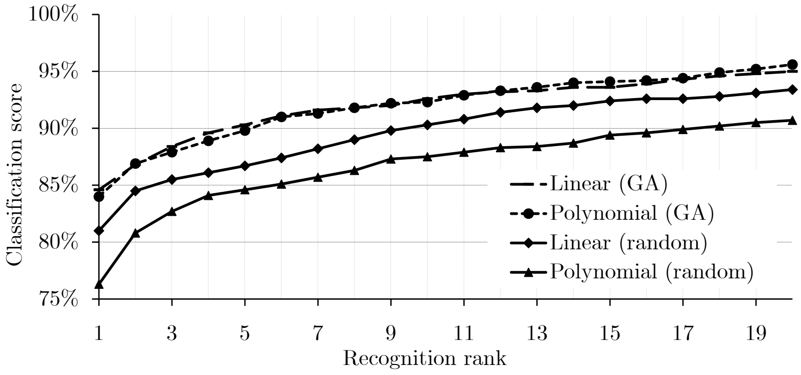


Fig. 21 Cumulative match curve for SVM trained using GA-optimized and randomly-selected training set.

The overall conclusion is that the SVM-based feature vector comparison is more effective than the conventional approach. Using linear SVM is equivalent to assigning weights to the feature space dimensions, where the weights are obtained as subsequent elements of the separating hyperplane normal vector. Hence, SVM training may be used to obtain the weights, and the similarity score between two feature vectors may be obtained as a weighted average of their difference vector elements. This is beneficial for the performance, as computing a weighted average is much faster than running the SVM classification.

15.3.6 Multi-method Fusion

The last example of applying SVM to automatic face recognition, presented in this chapter, is concerned with the multi-method fusion. This task is aimed at generating a final similarity score S from n partial similarities s_i, delivered by each feature extraction method. Here, we performed fusion of four different feature extraction methods, i.e.: 1) the Eigenfaces with Euclidean metric, 2) the Eigenfaces with Mahalanobis metric, 3) the Fisherfaces method [3], which is based on linear discriminant analysis, and 4) elastic bunch graph matching (EBGM) [68]. Although using several different feature extraction methods generates an additional time cost, it makes it possible to improve the overall classification score. Basically, taking into account low dimensionality of the input vectors, the problem's nature appears to be linear – the lower partial similarities are, the lower final similarity should be. Following this assumption, an improvement can be theoretically achieved by computing a weighted mean of partial similarities:

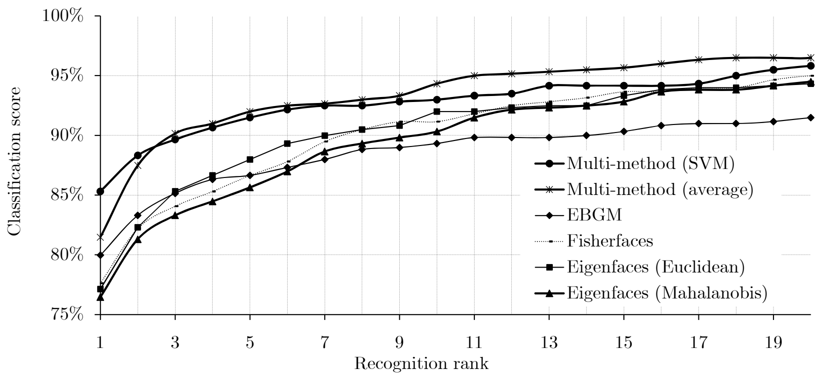


Fig. 22 Cumulative match curves for single feature extraction methods and their fusion using averaging and SVM-optimized weights.

$$S = \sum_{i=1}^{n} w_i s_i, \quad \sum_{i=1}^{n} w_i = 1, \tag{9}$$

where w_i is a weight of an i-th feature extraction method. To render that process effective, the most challenging task is to determine optimal weights.

Here, we used SVM for classifying n-th dimensional vectors $\mathbf{s}$ of partial similarities ($n = 4$ for the presented case). In training we have adapted a similar procedure to that presented in the previous section, i.e. we split all available similarity vectors generated from a training set into two categories: intra-class and inter-class similarity vectors. After training, SVM determines whether a given vector of partial similarities is intra- or inter-class.

Contrary to the cases outlined earlier, only linear kernel occurred effective for this application, and GA-based training set optimization did not render any gain compared to random subset selection (we selected $k = 200$ vectors to each class from the entire training set). This fact is in favor of the assumption that this is a linear problem of low complexity, and SVM occurred very effective here to optimize the weights w_i. The results of face classification score are presented in Fig. 22 as CMC curves. We show the performance of four basic methods, fusion obtained without weights optimization (for $w_i = 1$), and using the weights obtained after SVM training. It may be noticed that first-rank scores of the statistical face recognition methods (i.e. the Fisherfaces and both variants of the Eigenfaces) are very close to each other, and EBGM performs definitely better. Averaging the similarity score improves the overall result by 1.5% compared to the best single method (i.e. EBGM), and using the optimized weights – by 5.3%. This example clearly explains that the multi-method fusion may improve face recognition performance at a cost of additional processing. This is yet another area where SVM has been found very useful.

15.4 Conclusions

In this chapter two cases of support vector machines application were presented and deeply analyzed. Naturally, the range of potential areas, in which SVM may be found helpful, is definitely larger than that demonstrated here. However, our intention was to present the methodology of using and tuning this valuable tool in artificial intelligence.

First, SVM was used for classifying feature vectors extracted from proteomic mass spectra. It has been explored whether SVM can be trained to make distinction between patients suffering from early lung cancer and a healthy control group. Thus SVM with different kernels was used to classify the mass spectra after preprocessing. The features were selected for the classification based on t-test. It been also shown that elimination of the correlated features improved the final classification score, as well as it reduced the required number of features to SVM. When a small value of the radial kernel (σ) was selected, the minimum total error was obtained for 6–8 features. However when the sigma value increased, first the minimum error was achieved for a very large number of features. Furthermore, for a sizeable values of sigma and for the linear SVM the value of the minimal error stabilized without re-growth after its decrease.

There are many possible directions for future works in the presented field. For example, the presented method can be further developed to find discriminative features in MS spectra which would make it possible to differentiate between various cancers. Also, it may be investigated whether different types of cancers have common features that can be observed in MS spectra. In general, this would contribute to increasing the importance of the proteomics in the cancer diagnostics.

Furthermore, three selected cases of applying support vector machines to face recognition were analyzed in details. On the basis of experimental results it was demonstrated that face recognition performance can be significantly improved. Naturally, these are just a few examples out of a wide spectrum of SVM's applications to this field. Finally, it is worth noting that the presented methods are not limited to face recognition and can also be applied to familiar problems in image and signal processing.

References

1. Aebersold, R., Mann, M.: Mass spectrometry-based proteomics. Nature (422) (2003)
2. Bartlett, M., Movellan, J., Sejnowski, T.: Face recognition by independent component analysis. IEEE Transactions on Neural Networks **13**, 1450–1464 (2002)
3. Belhumeur, P., Hespanha, J., Kriegman, D.: Eigenfaces vs. Fisherfaces: recognition using class specific linear projection. IEEE Transactions on Pattern Analysis and Machine Intelligence **19**, 711–720 (1997)

4. Boser, B.E., Guyon, I.M., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: Proceedings of the fifth annual workshop on Computational learning theory, pp. 144 – 152. Pittsburgh (1992)
5. Box, G.E.P., Cox, D.R.: An analysis of transformations. JSTOR **62**(2), 211–252 (1964)
6. Cao, L.J., Tay, F.E.H.: Support vector machine with adaptive parameters in financial time series forecasting. IEEE Transactions on Neural Networks **14**(6), 1506 – 1518 (2003)
7. Cauwenberghs, G., Poggio, T.: Incremental and decremental support vector machine learning. In: Advances in neural information processing systems 13: proceedings of the 2000 conference, p. 409. The MIT Press (2001)
8. Chapelle, O.: Training a support vector machine in the primal. Neural Computation **19**(5), 1155–1178 (2007)
9. Chen, X., Yang, J., Liang, J., Ye, Q.: Smooth twin support vector regression. Neural Computing and Applications pp. 1–9 (2010). URL http://dx.doi.org/10.1007/s00521-010-0454-9
10. Cristianini, N., Shawe-Taylor, J.: An Introduction to Support Vector Machines and other kernel-based learning methods. Cambridge university press (2000)
11. Drucker, H., Burges, C.J.C., Kaufman, L., Smola, A., Vapnik, V.: Support vector regression machines. In: M.C. Mozer, M.I. Jordan, T. Petsche (eds.) Advances in Neural Information Processing Systems 9, pp. 155 – 161. MIT Press, Cambridge (1997)
12. Fernández, R.: Predicting time series with a local support vector regression machine. In: Proceedings of the ECCAI Advanced Course on Artificial Intelligence (ACAI 99) (1999)
13. Goldberg, D.: Genetic Algorithms in Search, Optimization, and Machine Learning. Addison-Wesley Publishing Co. (1989)
14. Gong, S., McKenna, S., Psarrou, A.: Dynamic Vision From Images to Face Recognition. Imperial College Press (1999)
15. Grother, P., Micheals, R., Phillips, P.J.: Face recognition vendor test 2002 performance metrics. In: Proceedings of the Fourth International Conference on Audio-Visual Based Person Authentication (2003)
16. Hao, P.Y.: New support vector algorithms with parametric insensitive/margin model. Neural Networks **23**, 60–73 (2010)
17. Hastie, T., Tibshrani, R., Friedman, J.: Clinical Proteomics: From Diagnosis to Therapy. Springer-Verlag (2001)
18. Hilario, M., Kalousis, A., Pellegrini, C., Mller, M.: Processing and classification of protein mass spectra. Bioinformatics **25**(3), 409–449 (2006)
19. Hochreiter, S., Obermayer, K.: Gene selection for microarray data. In: B. Scholkopf, K. Tsuda, J. Vert (eds.) Kernel Methods in Computational Biology, pp. 319–355. MIT Press (2004)
20. Huang, J., Blanz, V., Heisele, B.: Face recognition using component-based svm classification and morphable models. In: Proceedings of the First International Workshop on Pattern Recognition with Support Vector Machines, SVM '02, pp. 334–341. Springer-Verlag (2002)
21. Jesorsky, O., Kirchberg, K.J., Frischholz, R.W.: Robust face detection using the hausdorff distance. In: Audio and Video based Person Authentication - AVBPA, pp. 90–95. Springer (2001)
22. Joachims, T.: Text categorization with support vector machines: Learning with many relevant features. In: Machine Learning: ECML-98 (LNCS), vol. 1398, pp. 137–142 (1998)

23. Joachims, T.: Training linear svms in linear time. In: T. Eliassi-Rad, L.H. Ungar, M. Craven, D. Gunopulos (eds.) KDD, pp. 217–226. ACM (2006)
24. Kanji, G.K.: 100 statistical tests, 3 edn. SAGE Publications Ltd (2006)
25. Karpievitch, Y.V., Hill, E.G., Smolka, A.J., Morris, J.S., Coombes, K.R., Baggerly, K.A., Almeida, J.S.: Prepms: Tof ms data graphical preprocessing tool. Bioinformatics **23**(2), 264–265 (2007)
26. Kawulok, M., Szymanek, J.: Algorithm for precise frontal face detection. Studia Informatica **30**, 341–354 (2009)
27. Kawulok, M., Wu, J., Hancock, E.R.: Supervised relevance maps for increasing the distinctiveness of facial images. Pattern Recognition **44**(4), 929–939 (2011)
28. Kin, T., Kato, T., Tsuda, K.: Protein classification via kernel matrix completion. In: B. Scholkopf, K. Tsuda, J. Vert (eds.) Kernel Methods in Computational Biology, pp. 261–274. MIT Press (2004)
29. Krishnapuram, B., Carin, L., Hartemink, A.: Gene expression analysis: Joint feature selection and classifier design. In: B. Scholkopf, K. Tsuda, J. Vert (eds.) Kernel Methods in Computational Biology, pp. 299–317. MIT Press (2004)
30. Lee, Y., Huang, S.: Reduced support vector machines: A statistical theory. Neural Networks, IEEE Transactions on **18**(1), 1–13 (2006)
31. Leski, J.: On support vector regression machines with linguistic interpretation of the kernel matrix. Fuzzy Sets and Systems **157**, 1092–1113 (2006)
32. Lilliefors, H.L.: On the kolmogorovsmirnov test for normality with mean and variance unknown. JASA **62**, 399–402 (1967)
33. Liu, Q., Krishnapuram, B., Pratapa, P., Liao, X., Hartemink, E., Carin, L.: Identification of differentially expressed proteins using maldi-tof mass spectra. In: Asilomar Conference: Biological Aspects of Signal Processing (2003)
34. Maio, D., Maltoni, D.: Real-time face location on gray-scale static images. Pattern Recognition **33**, 1525–1539 (2000)
35. Morris, J.S., Coombes, K.R., Koomen, J., Baggerly, K.A., Kobayashi, R.: Feature extraction and quantification for mass spectrometry in biomedical applications using the mean spectrum. Bioinformatics **21**(9), 1764–1775 (2005)
36. Na, S., Paek, E.: Quality assessment of tandem mass spectra based on cumulative intensity normalization. J Proteome Res. **5**(12), 3241–3248 (2006)
37. Olofsson, P.: Probability, Statistics, and Stochastic Processes. John Wiley & Sons (2005)
38. Osuna, E., Freund, R., Girosi, F.: Training support vector machines: an application to face detection. Computer Vision and Pattern Recognition, IEEE Computer Society Conference on pp. 130–136 (1997)
39. Osuna, E., Freund, R., Girosi, F.: Training Support Vector Machines: an application to face detection. In: Proc. IEEE Conf. Computer Vision and Pattern Recognition, pp. 130–136 (1997)
40. Petricoin, E.F., Ardekani, A.M., abd Peter J. Levine, B.A.H., Fusaro, V.A., Steinberg, S.M., Mills, G.B., Simone, C., Fishman, D.A., Kohn, E.C., Liotta, L.A.: Use of proteomic patterns in serum to identify ovarian cancer. The Lancet **359**, 527–577 (2002)
41. Phillips, P., Flynn, P., Scruggs, T., Bowyer, K., Chang, J., K.Hoffman, Marques, J., Min, J., Worek, W.: Overview of the Face Recognition Grand Challenge. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, vol. 1, pp. 947–954 (2005)

42. Phillips, P., Grother, P., Micheals, R., Blackburn, D., Tabassi, E., Bone, J.: Face Recognition Vendor Test 2002: Evaluation Report. NISTIR 6965 (2003)
43. Phillips, P., Wechsler, H., Huang, J., Rauss, P.: The FERET database and evaluation procedure for face recognition algorithms. Image and Vision Computing J **16**(5), 295–306 (1998)
44. Phillips, P.J.: Support vector machines applied to face recognition. In: Advances in Neural Information Processing Systems 11, pp. 803–809. MIT Press (1999)
45. Pietrowska, M., Marczak, L., Polanska, J., Behrendt, K., Nowicka, E., Walaszczyk, A., Chmura, A., Deja, R., Stobiecki, M., Polanski, A., Tarnawski, R., Widlak, P.: Mass spectrometry-based serum proteome pattern analysis in molecular diagnostics of early stage breast cancer. J Transl Med. (2009). 7:60
46. Pietrowska, M., Marczak, L., Suwinski, R., Stobiecki, M., Polanska, J., Polanski, A., Widlak, P., Gawkowska-Suwinska, M., Drosik, A., Walaszczyk, A.: Application of mass spectrometry-based serum proteome pattern analysis in identification of lung cancer patients. J Thorac Oncol **5**(5, Suppl 1), S60 (2010). Abstract book, 2nd European Lung Cancer Conference, Geneva, Switzerland, 28 April-1 May 2010
47. Polanska, J., Widnak, P., Rzeszowska-Wolny, J., Kimmel, M., Polanski, A.: Gaussian mixture decomposition of time-course dna microarray data. In: Mathematical Modeling of Biological Systems, Volume I, Modeling and Simulation in Science, Engineering and Technology, pp. 351–359. Birkhuser Boston (2007)
48. Polanski, A., Kimmel, M.: Bioinformatics. Springer (2007)
49. Ralaivola, L., dAlché Buc, F.: Incremental support vector machine learning: A local approach. Artificial Neural NetworksICANN 2001 pp. 322–330 (2001)
50. Ratsch, G.: Accurate splice site detection for caenorhabditis elegans. In: B. Scholkopf, K. Tsuda, J. Vert (eds.) Kernel Methods in Computational Biology, pp. 277–298. MIT Press (2004)
51. Roobaert, D.: DirectSVM: A fast and simple support vector machine perceptron. In: Neural Networks for Signal Processing X, 2000. Proceedings of the 2000 IEEE Signal Processing Society Workshop, vol. 1, pp. 356–365. IEEE (2000)
52. Rowley, H., Baluja, S., Kanade, T.: Neural network-based face detection **20**(1), 23–38 (1998)
53. Schlkopf, B., Bartlett, P.L., Smola, A.J., Williamson, R.C.: Shrinking the tube: A new support vector regression algorithm. In: M.J. Kearns, S.A. Solla, D.A. Cohn (eds.) Advances in Neural Information Processing Systems 11, pp. 330–336. The MIT Press (1999)
54. Schlkopf, B., Smola, A.J., Williamson, R.C., Bartlett, P.L.: New support vector algorithms. Neural Coputation **12**(5), 1207–1245 (2000)
55. Shalev-Shwartz, S., Srebro, N.: Svm optimization: inverse dependence on training set size. In: Proceedings of the 25th international conference on Machine learning, ICML '08, pp. 928–935. New York, NY, USA (2008)
56. Shin, H., Sampat, M.P., Koomen, J.M., Markey, M.K.: Wavelet-based adaptive denoising and baseline correction for maldi tof ms. OMICS **14**(3), 283–295 (2010)
57. Shipp, M., Ross, K., Tamayo, P., Weng, A., Aguiar, R., Kutok, J., Gaasenbeek, M., M.Angelo, Reich, M., Ray, T., Pinkus, G., Koval, M., Last, K., Norton, A., Mesirov, J., Lister, T., Neuberg, D., Lander, E., Aster, J., Gloub, T.: Diffuse large b-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning. Nature Medicine **8**(1), 68–74 (2002)

58. S.L.Pomeroy, Tamayo, P., Gaasenbeek, M., Sturla, L., Angelo, M., McLaughlin, M., Kim, J., Goumnerova, L., Black, P., Lan, C., Allen, J., Zagzag, D., Olson, J., Curran, T., Wetmore, C., Biegel, J., Poggio, T., Mukherjee, S., Rifkin, R., Califano, A., Stolovitzky, G., Louis, D., Mesirov, J., Lander, E., Golub, T.: Prediction of central nervous system embryonal tumour outcome based on gene expression. Nature **415**(24), 436–442 (2002)
59. Steinwart, I.: Sparseness of support vector machines. J. Mach. Learn. Res. **4**, 1071–1105 (2003)
60. Tay, F.E.H., Cao, L.J.: Modified support vector machines in financial time series forecasting. Neurocomputing **48**(1-4), 847–861 (2002)
61. Turk, M., Pentland, A.: Face Recognition Using Eigenfaces. In: Proceedings of IEEE Conference on Computer Vision and Pattern Recognition, pp. 586–591 (1991)
62. Van Eyk, J.E., Dunn, M.J.: Clinical Proteomics: From Diagnosis to Therapy. Wiley-VCH (2008)
63. Veer, L.V., Dai, H., van de Vijer, M., He, Y., Hart, A., Mao, M., Peterse, H., van der Kooy, K., Marton, M., Witteveen, A., Schreiber, G., Kerkhoven, R., Roberts, C., Linsley, P., Bernards, R., Friend, S.: Gene expression profiling predicts clinical outcome of breast cancer. Nature **415**(24), 530–536 (2002)
64. Viola, P., Jones, M.: Robust real-time face detection. International Journal of Computer Vision **57**(2), 137–154 (2004)
65. Vishwanathan, S., Murty, N., et al.: SSVM: a simple SVM algorithm. In: Neural Networks, 2002. IJCNN'02. Proceedings of the 2002 International Joint Conference on, vol. 3, pp. 2393–2398. IEEE (2002)
66. Wagner, M., Naik, D., Pothen, A.: Protocols for disease classification from mass spectrometry data. PROTEOMICS **3**(9), 1692–1698 (2003)
67. Wilhelm, T., Bohme, H.J., Gross, H.M.: Classification of face images for gender, age, facial expression, and identity. In: ICANN (1), pp. 569–574 (2005)
68. Wiskott, L., Fellous, J., Kruger, N., Malsburg, C.: Face recognition by Elastic Bunch Graph Matching. Tech. Rep. IR-INI 96-08, Ruhr-Universitat Bochum, Germany (1996)
69. Wu, J., Smith, W.A.P., Hancock, E.R.: Facial gender classification using shape-from-shading. Image Vision Comput. **28**(6), 1039–1048 (2010)
70. Yambor, W., Draper, B., Beveridge, R.: Analyzing PCA-based face recognition algorithms: Eigenvector selection and distance measures. Empirical Evaluation Methods in Computer Vision (2002)
71. Yang, M.H., Kriegman, D., Ahuja, N.: Detecting faces in images: A survey. IEEE Trans. Pattern Analysis and Machine Intelligence **24**, 34–58 (2002)
72. Yu, W., He, Z., Liu, J., Zhao, H.: Improving mass spectrometry peak detection using multiple peak alignment results. J Proteome Res. **7**(1), 123–129 (2008)
73. Yu, W., Wu, B., Huang, T., Li, X., Williams, K., Zhao, H.: Statistical methods in proteomics. In: H. Pham (ed.) Springer Handbook of Engineering Statistics, pp. 623–638. Springer (2006)
74. Zhang, C., Zhang, Z.: A survey of recent advances in face detection. Tech. rep., Microsoft Research (2010)
75. Zhao, W., Chellappa, R., Phillips, P., Rosenfeld, A.: Face recognition: A literature survey. Tech. Rep. CARTR-948, Center for Automation Research, University of Maryland, College Park (2000)

Chapter 16
Workload Modeling for Multimedia Surveillance Systems

Mukesh Saini, Pradeep K. Atrey, and Mohan S. Kankanhalli[⋆]

Abstract. A multimedia surveillance system has to sustain high computational loads. Such a system needs to intelligently learn the characteristics of the workload that it is going to handle. The knowledge of the workload also provides a strong basis for design and optimization of the system components. To efficiently use this knowledge, we need an analytical model of the workload. The traditional multimedia workload models used in other domains are not appropriate for surveillance systems. In other domains, the workload characteristics are mainly derived from the statistical properties of the data, whereas in the case of surveillance, the semantics play a dominant role in determining the processing needs. In this chapter, we discuss popular workload models from other domains and explore their applicability to surveillance systems. We find that none of those models describe the workload accurately in surveillance context. Following this observation, we propose a novel Markov chain based formal model of multimedia workload for surveillance systems. Different states of the Markov chain meticulously capture the variability of the workload. The model is validated with real surveillance data. Subsequently, we describe performance analysis of a real surveillance system based on the proposed model.

16.1 Introduction

Multimedia surveillance systems are required to keep up with high computational load in order to analyze a large number of media streams. The analysis tasks, in

Mukesh K. Saini · Mohan S. Kankanhalli
School of Computing, National University of Singapore, Singapore
e-mail: {mksaini,mohan}@comp.nus.edu.sg

Pradeep K. Atrey
Department of Applied Computer Science, The University of Winnipeg, Canada
e-mail: p.atrey@uwinnipeg.ca

[⋆] Authors would like to thank the Canadian Bureau for International Education and the Natural Sciences and Engineering Research Council of Canada for their support in this work.

particular the computer vision algorithms, put huge processing demands on the system. This workload adversely affects the performance of the system. Therefore, a study of the effects of the workload on the evaluation of surveillance systems is paramount.

Performance evaluation has gained significant attention in the surveillance research community recently. Consequently, various efforts have been made towards investigating evaluation techniques to measure the performance of surveillance related algorithms [18]. For algorithmic evaluation these techniques may be sufficient; however, these are not adequate for system level analysis [18]. Current surveillance systems accomplish complex tasks with various levels of processing [7], and it is not feasible to implement the systems just for sake of evaluation. Also, for a true understanding of system behavior, a large number of experiments need to be conducted. This is hard to achieve in many cases, such as with wide area surveillance systems. It would take a long time and significant amount of efforts to repeat a whole set of experiments even with a slight change in system parameters. In summary, the performance of a surveillance system needs to be evaluated using an appropriate analytical model.

Analytical modeling helps in evaluating the system with respect to the workload it must handle. Hence, to successfully design a surveillance system, we need to know the characteristics of the workload. This forms the basis of design and optimization of the system components. Furthermore, to efficiently use this knowledge we need to have a formal model of the workload. Since the simulators are generally based on analytical models, in this work we will focus on the workload characterization appropriate for analytical evaluation. Having a formal workload model for multimedia surveillance systems offers the following advantages:

- The computer resources are shared by different system components and need a scheduling policy. A workload model is required for evaluating the given scheduling policy.
- Similarly, workload characteristics are also necessary for various energy saving schemes and run-time system adaption.
- In a collaborative environment, when one processor reaches a processing bottleneck, it can distribute some of the work to neighboring processors. The successful implementation of such interactions requires a good understanding of the nature of the workload the processors are handling.

To the best of our knowledge, there have been only a few works on formal modeling of surveillance workloads [21]. In this work the time between two events is assumed to be exponentially distributed, and queuing network models are used for performance analysis. There are three problems with this approach: 1) the application of the model is limited to scenarios where only one event occurs at a time, 2) the processing time may vary drastically depending on the environment, but the model does not capture this variability, and 3) the assumption of Poisson arrivals is true only in certain scenarios, such as ATM kiosks. In places where more activities occur the events are generally correlated.

There has been much work on workload modeling in the areas of e-commerce [15, 28], web servers [8, 2], embedded systems [22], operating systems [1], and networks [29, 19]. In these works, the systems logs are collected and clustering techniques are used to classify the workloads. The classification is generally based on the information extracted from data logs. For instance, in web server traffic modeling, the processing time of the queries is analyzed against the time to estimate the distribution. The surveillance task analysis needs mainly depend on the semantic content of the workload, which eventually depends on the scenario the sensors are monitoring. For example, the processing time for a tracking system mainly depends on the number of targets being tracked and is not an absolutely random phenomenon. This high correlation between the semantics of operating scenarios and the processing needs have not been explicitly considered in the modeling techniques in other domains. This motivates us to investigate a workload characterization model to augment the performance evaluation of multimedia surveillance systems.

16.1.1 Issues in Workload Characterization

Workload consists of a set of tasks. These tasks have a number of attributes, such as processing demands, memory demands, coordination requirements, dependencies on other tasks, etc. While modeling all the attributes is out of the scope of this chapter, we have chosen following important characteristics [7, 14]:

- **Task arrival rate** The rate at which tasks arrive at the system. It is generally measured as inter arrival times.
- **Processing demand** The amount of processing requested by each task. It is measured as the service time required for each task.
- **Memory demand** The task at hand needs to be stored in main memory for fast execution. It may cause additional memory requirements on the system in terms of data storage, etc. This is measured as the buffer memory requirement.

16.1.2 Contributions Summary

The main contributions in this chapter are as follows:

- We propose a Markov chain based workload model[1] to evaluate a surveillance system. The proposed model captures the variability of the workload in its different states. The timing information of the workload is preserved in the transition matrix, and the correlation with the operating environment is exploited to make it tunable to different scenarios. The model can be applied in different surveillance scenarios with minimal user inputs.
- To demonstrate the efficacy of the proposed model, we describe performance evaluation of a real surveillance system. The model is validated by conducting experiments with a real time surveillance system.

[1] A preliminary version of the proposal model with initial results was published in [20]

16.1.3 Chapter Organization

To begin with, surveillance system itself is not a clearly defined term. We give a overview of a typical surveillance system in Section 16.2. Then we discuss popular workload models from other domains and explore their applicability to the surveillance domain in Section 16.3. It is found that none of those models accurately describes the workload in surveillance context. Following this observation, we propose a novel formal model for surveillance workload modeling in Section 16.4. Afterwards, we demonstrate performance evaluation of a real surveillance system and show how to choose the model parameters in Section 16.5. Experimental results are provided in Section 16.6. We conclude the chapter in Section 16.7.

16.2 Surveillance System

In this section our main goal is to identify typical components of a surveillance system and generate a system level view. A system level view describes what components are in the system and how they communicate. A typical surveillance system has the following components: sensors, networks, and computers (memory, processor, etc.).

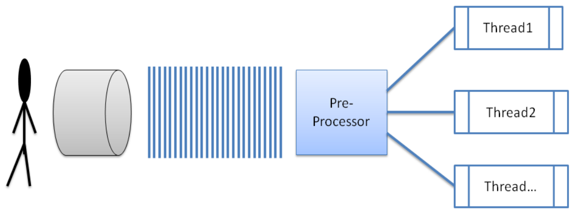


Fig. 1 A block diagram of a typical surveillance system.

It is found that a typical surveillance system achieves its goals by performing the following tasks [10, 6, 16]:

- Target detection: The surveillance system has to detect people, vehicles, and many other kinds of objects. We group them together and label them as targets. Target detection consists of two tasks: background subtraction followed by blob detection.
- Target recognition and tracking: For each detected object, a recognition task is performed to determine if it is a new object or one that was already in the sensor coverage area.
- Information fusion and activity analysis: Finally, the tracking information from multiple sensors is combined to perform multisensor tracking and activity analysis.

The block diagram of this hypothetical system is shown in Figure 1. Note that the above description is for a typical surveillance system. Particular architectures can be realized with slight modifications in the described system. For example, abandoned baggage can be detected by analyzing the trajectory information of various targets.

16.3 Previous Work

The earlier work on surveillance performance evaluation measured the workload in terms of events with exponential inter-arrival times [21]. This assumption is true in only specific scenarios where the task is defined in terms of events. Furthermore, in real applications it is hard to determine the events as there can be multiple events happening simultaneously when multiple objects are present. For a more general and realistic solution, we have chosen to define a frame as a task unit. The frame arrival rate and size is usually constant and depends mainly on sensor specifications. The processing time for each frame varies over time, depending on the environmental conditions. For example, a surveillance system with the goal of tracking and activity analysis has to track each target individually; hence, the processing time depends on the number targets. As we could not find much work on surveillance workload modeling, this section outlines similar works in other domains to highlight our contributions.

There has been many works on workload modeling with variable processing demands in other fields. In synthetic workload generation, the actual workload traces are collected to estimate the empirical probability distributions of workload parameters [4, 26]. Similarly, in works [3, 4, 5], the workload is divided into different clusters, and representative samples are derived for each cluster. While these models perform reasonably well for generating synthetic workloads for simulation purposes, they are not appropriate for mathematical analysis.

Alternatively, the stochastic workload models are often adopted to represent the workload. The most common approach is to estimate the probability distribution functions of parameters [9, 21]. The problem with this approach is that generally it is very hard to estimate adequate distribution for multi-class data. Also, these models completely omit the dynamic characteristics of the workload. Surveillance workload consists of different types of frames with widely varying resource demands, not only between types, but also within each type. As a consequence, the single class models [21] are not suitable for surveillance workload.

Maxiaguine et al. [14] proposes *wcet* (Worst Case Execution Time) and *bcet* (Best Case Execution Time) to characterize the workload for real time embedded systems. The model is good to understand extreme behavior, still, it does not capture the dynamic characteristics of the workload and is hard to use for analytical evaluation. A Markov chain based model is proposed by Song et al. [25] which preserves the dynamic behavior of the workload in different states. The work mainly focuses on the parallel computers where the states are determined based on the number of nodes requested by the task. Yet, the model does not accommodate the semantic concepts like the target based dependence on processing time. In other approaches [23, 12], each task is modeled as a Markov Chain for enterprise application servers. Such models are good for removing the redundancy in data logs; however, they are not appropriate for mathematical modeling of the system.

Table 1 A comparison with previous works

Work	Multi Class	Time Information	Tunable	Data Logs Required	Semantic Considered	Tractable	Complexity
Sreenivasan & Kleinman[26]	No	No	No	High	No	No	High
Feitelson [9]	No	No	No	Medium	No	Yes	Medium
Calzarossa & Ferrari [4]	Yes	No	No	High	No	No	High
Cadez et al. [3]	Yes	No	No	High	No	No	High
Maxiaguine et al. [14]	Yes	No	No	High	No	No	Medium
Song et al. [25]	Yes	Yes	No	No	Medium	No	Yes
Sharifimehr & Sadaoul [23]	Yes	No	No	High	No	No	High
Maksyagin [13]	Yes	No	No	Medium	No	No	Medium
Saini et al. [21]	No	No	No	High	No	Yes	Low
Proposed model	***Yes***	***Yes***	***Yes***	***Low***	***Yes***	***Yes***	***Low***

In contrast to the works described above, the proposed model preserves the timing characteristics of the workload in the states of a Markov chain, can be tuned to work in different scenarios with minimal user input, requires minimal workload trace to estimate parameters of Gaussian distribution, considers the semantics of the environment in terms of targets, is easy to calculate, and is mathematically tractable for performance analysis of the system. Table 1 summarizes the advantages of the proposed model.

16.4 Proposed Model

Figure 2 shows a topological view of a real surveillance system installations. The media streams are generated at sensors, transmitted to the processor over the network, and processed to accomplish the surveillance goal. From a system designer's perspective, cameras, network, and processors form the system; and the environment is the workload generator. However, modeling the workload at the environment level is hard and not efficient from the point of view of performance analysis. In this work, the cameras are considered part of the environment and they are the main entities that generate the workload. Note that, in this work, we do not model the network effect on inter-frame times. This is because in practice the cameras are generally connected through dedicated cables or a high speed Ethernet where the transmission time is negligible compared to the inter-frame delay. For larger surveillance systems, the workload exposed to the processor can be easily re-calculated once the system architecture has been finalized [11]. The main motivation of not including the network in workload modeling is to provide generality to the model.

In the proposed workload model, we use a Markov chain to represent the number of targets (usually people) in the environment. We define a target flow graph that can be easily constructed by observing the operating scenario. This graph is used to measure the transition probabilities and subsequently the steady state probabilities of the states. These states have been identified such that the workload shows similar behavior in each class and it can be abstracted using stochastic methods. Finally we formulate the characteristics of the workload in each class.

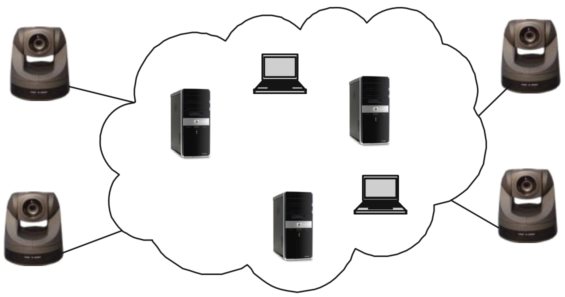


Fig. 2 The cloud represents the network. The processing units and the users are distributed over the network. Sensors are placed at the edge of the cloud to capture environmental information in the form of data frames.

16.4.1 Target Flow Graph (TFG)

The surveillance systems are generally designed to observe not only human behavior but also other type of objects such as abandoned baggage [24], vehicles [17], etc. In this work we use the term target to refer to humans and other objects [27], unless stated otherwise. Once the surveillance site has been fixed, we need to learn the dynamics of the environment. Actual workload is derived based on this knowledge and is represented as a target flow graph. The target flow graph TFG is constructed as a set of tuples:

Table 2 List of important symbols used in the chapter

χ	State transition matrix
p_{ij}	Transition probabilities
π_i	Steady state probabilities
l	Number of samples
m	Number of states
F_r	Frame rate
t_a	inter-frame time
t_c	State change overhead time
t_i^q	Processing time in i^{th} state
t_i^p	Processing demand in i^{th} state
t_i^q	Waiting time in i^{th} state
T_i^p	Average Processing demand in i^{th} state
T_i^w	Average waiting time in i^{th} state
T_i	Average response time in i^{th} state
T	Average response time
B	Buffer size in number of frames
m_c	Memory overhead due to state change

$$TFG = \{(ts_k, g_k) \mid k \in [1,l]\} \tag{1}$$

In Equation (1), g_k is the number of targets at ts_k^{th} time instant and l is the total number of samples. In other words, we represent the flow of targets in a time series format i.e. number of targets in the coverage area at sampling instances. High sampling rates and large observation times provide better model accuracy.

16.4.2 Markov Chain Construction

In a typical surveillance system, each target is individually tracked and the tracking results with other contextual information are stored for further analysis. As discussed earlier, unlike other domains, in the case of surveillance, the processing time and memory requirement for each frame mainly depend on the number of targets in the monitoring area. We capture this dependence in a Markov chain. There are two advantages of modeling workload as Markov chain:

1. Different states of a Markov chain can capture the variability of the workload. This is in contrast to the earlier model of surveillance workload which assumes a Poisson distribution and a single average event rate to characterize the workload [21].
2. The performance analysis can be done for each state and average values can be obtained by calculating the probabilistic sum of the performances for each state.

The number of states in a Markov chain is $m+1$ where m is given by the following equation:

$$m = max\{g_k \mid (ts_k, g_k) \in TFG, k \in [1,l]\} \tag{2}$$

Specifically, m represents the maximum number of people simultaneously expected in the monitoring area. Such information can be easily obtained from analysis of the environment where the surveillance system is to be installed. To capture the variability of the workload, we define different states of a Markov chain based on the number of targets. The set of states S of this Markov chain can be constructed as follows:

$$S = \{s_0, s_1, ...s_m \mid \forall(i,j) : i \neq j \wedge (i,j) \in [1,m], s_i = i, s_i \neq s_j\} \tag{3}$$

Figure 3 shows the states of the Markov chain for different frames of a PETS [18] video sequence. The images corresponding to these states are shown in Figure 4. There are two types of probabilities associated with a Markov chain: transition probability and steady state probability. The definitions of these probabilities in surveillance context follows:

Definition 1: Transition probability is the probability of finding a particular number of targets at the next time instant.

Definition 2: Steady state probability is the probability of finding a particular number of targets after the system has run for sufficient time.

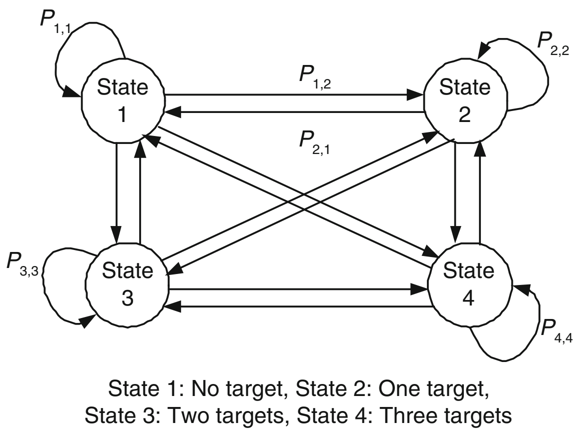

$P_{1,1}$
$P_{1,2}$
$P_{2,1}$
$P_{2,2}$
$P_{3,3}$
$P_{4,4}$
State 1
State 2
State 3
State 4
State 1: No target, State 2: One target,
State 3: Two targets, State 4: Three targets

Fig. 3 Different states of the Markov chain depending on the number of targets and transition probabilities.

The transition probabilities are represented in the form of a matrix χ. The elements of the transition matrix can be constructed as follows:

$$\chi = \{p_{ij} \mid p_{ij} = \frac{n_{ij}}{n_i}, (i,j) \in [1,m]\} \tag{4}$$

where n_{ij} and n_i are defined as

$$n_i = \mid \{g_k = i, (ts_k, g_k) \in TFG, k \in [1,l]\} \mid \tag{5}$$
$$n_{ij} = \mid \{g_k = i \wedge g_{k+1} = j, (ts_k, g_k) \in TFG, k \in [1, l-1]\} \tag{6}$$

These transition probabilities are of great use in system design. Once the system performance and resource requirements are known, these probabilities can be used for efficient resource management. However, the more important property of a Markov chain is the steady state probability of the states. Let $\Pi = (\pi_0, \pi_1, \ldots \pi_m)$ be the steady state probabilities of the states. The probabilities are calculated using one of the following two ways:

$$(\chi - I)\Pi = 0 \tag{7}$$
$$\Pi = \{\pi_i \mid \pi_i = p'_{ij}, p'_{ij} \in \chi^{Inf}, i = C, j \in [1,m]\} \tag{8}$$

In the above formulation, I is identity matrix, Inf is a large number (≈ 100), and C is any number between 0 and m. In fact, all the rows of the matrix χ^{Inf} will have similar values. The equation (7) gives the eigenvector of the transition probability matrix for the eigenvalue of 1.

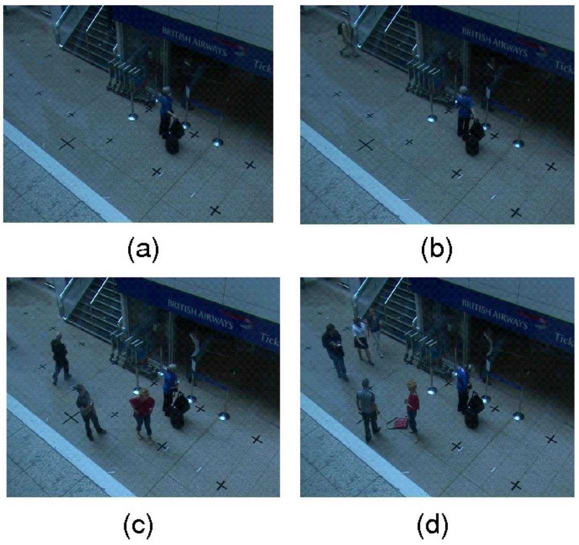


Fig. 4 Frames in different states of the Markov chain: (a) State 3 (b) State 4 (c) State 5 (d) State 8

In the following subsections we will model inter-arrival times, processing needs, and memory needs for each state. Note that in the proposed model, a task consists of a frame and the above described attributes are associated with each frame. For example, the processing demand is the time a particular frame is processed in order to extract the useful information.

16.4.3 Task Arrival

In the proposed model the tasks are generated by sensors in the form of frames. The rate at which these frames are captured by the camera determines the task arrival rate. Here we assume that the network used to transmit frames from sensors has a deterministic delay. The effects of the network behavior on the inter-frame delay can be studied separately and have been omitted from this chapter due to space constraints. The task arrival rate is the same as the frame rate of the sensors which is independent of the state of the Markov chain. Let F_r be the specified camera frame rate, then, the inter arrival time t_a is given by:

$$t_a = \frac{1}{F_r} \tag{9}$$

Since the sensors are generally connected to the processor either by dedicated analog cables or Ethernet switches, it is reasonable to assume the inter-frame time to be deterministic. To adapt the model to the scenarios where the frames need to travel through multiple routers before reaching the processor, the inter-frame time can be modeled separately. The other aspects of the model will still successfully apply in those scenarios.

16.4.4 Processing Demand

The time required to process a frame is called service time. More precisely, it is the time interval between the instances the system begins and ends processing a frame. The beginning time for a frame is the instant when the previous frame completes. If there is a buffer that stores frames to be processed, the service time determines the rate at which the frames are read from that buffer. There are two components of processing demand, the frame processing time and the state change overhead. Here, we first formulate the frame processing time and then investigate the effect of state change on the timing performance.

As mentioned earlier, the system deals with each target separately. This includes functionalities like detection, recognition, tracking, behavior analysis, etc. Therefore the amount of processing required by each task (frame) mainly depends on the state of the Markov chain. Though the processing times do not vary much within each state, there are random phenomena which can cause variation in processing time e.g. foreground size, the area of the targets (aka size of blob), etc. We capture this variability by modeling the processing time in each state as a Gaussian random variable. If we denote the processing time as t_i^q, its probability density function for the i^{th} state can be calculated as:

$$f(t_i^q) = \frac{1}{\sqrt{2\pi\sigma_i{}^2}} e^{-\frac{(t_i^q-\mu_i)^2}{2\sigma_i{}^2}} \tag{10}$$

where μ_i is the mean value and σ_i is the variance of the service times for i^{th} state. These values can be easily calculated for the given processor and camera specifications. Theoretical modeling of these values is very difficult since many hidden factors like memory type, cache speed, ambient temperature also affect these values; these should be determined empirically.

To validate the assumption that the processing time in each class is indeed Gaussian distributed, we conducted experiments and performed normality test on the observed data. The hypothesis that the service times are Gaussian distributed is found to be true after doing the test. The details of the test are provided in the experimental results section.

Whenever the number of targets in the frame changes, the system needs to perform some additional tasks to handle the newly appeared or just vanished target. In the system described in earlier sections, the additional work is thread creation and deletion. This may result in processing overheads resulting in additional timing delay. The state change probability P_i^c can be calculated by considering the transition probabilities:

$$P_i^c = \sum_{j=0,j\neq i}^{n} p_{i,j} \tag{11}$$

Here we add the probabilities based on our observation that state transitions are exclusive i.e. one state cannot transit to multiple states. The effective service time t_i^p is the sum of the frame processing time and thread creation overhead:

$$t_i^p = t_i^q + P_i^c t_c \tag{12}$$

where, t_c is average state change overhead. This value also depends of the system architecture and should be calculated empirically.

16.4.5 Memory Demand

If we assume that no data compression has been performed, the memory needed to store each frame is constant and completely depends on the resolution of the image. While frames are waiting to be processed in the buffer, the memory demand is equal to the frame size. However, there are some frames that can cause state change. Since the state change ultimately depends on the number of targets in the camera view, we incorporate the overhead due to state change in the task memory demand. When a new frame is picked-up for processing, there can be following three cases:

1. It does not change the state of the system. In this case it does not contribute to any additional memory requirement.
2. It changes to one of the higher states, here it causes generation of a new thread.
3. It changes the state of the system to a lower state, it reduces the memory requirement by one thread.

So the total memory requirement can be calculated as the sum of the memory required for a frame and the state change overhead. The transition probabilities can be used to calculate the state change probabilities. The total memory M^i for a frame of size F_s can be calculated as follows:

$$M^i = F_r + m_c\left(\sum_{j=i+1}^{n} p_{i,j} - \sum_{j=1}^{i-1} p_{i,j} \right) \tag{13}$$

When the frames are compressed at the sensor itself, the size of each frame should be determined based on the compression scheme. Yet, since more targets in a frame will result in larger foreground area, we expect that the frame memory to grow with the state value.

Once the frame processing starts, additional memory may be required for two purposes: thread management and storage of the target related information extracted from the frames. Again, this additional overhead depends on the system architecture.

16.5 Performance Evaluation

To demonstrate the efficacy of the proposed workload model, we calculate the performance of a real surveillance system and experimentally validate the results. The system consists of one camera connected to a processor by a dedicated line (Figure 5). The functionalities of the system are as described in Section 16.2.

For this system, we model the two most critical performance measures in surveillance context: system response time and frame drop probability. For the sake of clarity, let us first define these measures.

Definition 3: *System response time is the time period from the instant the frame arrives at the system to the time the system finishes extracting information from it and produces the results.*

Definition 4: *Frame drop probability is the probability of a frame arriving at the system and, finding the buffer full, being discarded.*

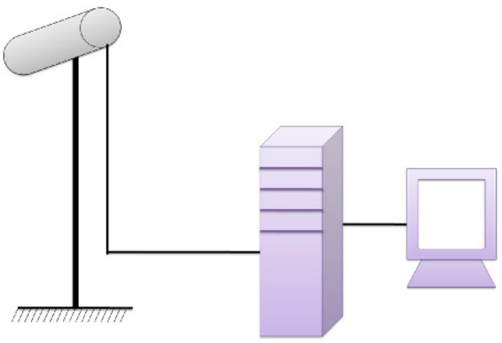


Fig. 5 The block diagram of the system implementation for performance evaluation and validation.

16.5.1 System Response Time

After the frame arrives at the system, it waits in the buffer for its turn to be processed. The system, depending on the content of the frame, applies analysis algorithms to extract useful information. Thus, there are two components of the response time t_i: a frame's processing demand (t_i^p) and waiting time in the buffer (t_i^w).

$$t_i = t_i^p + t_i^w \tag{14}$$

Equation (14) shows that the total response time is sum of two random variables. The following equation calculates the average values of both the random variables to measure the over all response time of the system:

$$T_i = T_i^p + T_i^w \tag{15}$$

The average processing demand, T_i^p, can be calculated by taking the expectation of t_i^p:

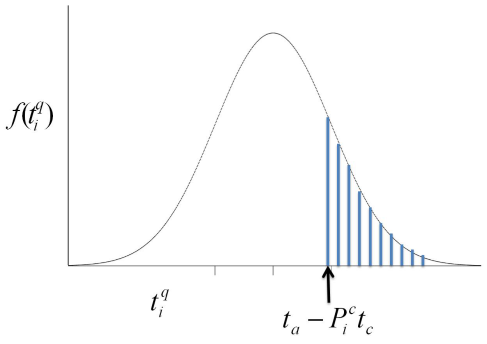


Fig. 6 The frames need to wait whenever $t_i^q > t_a - P_i^c t_c$, which is shown by lined area. The waiting time is $t_i^q - (t_a - P_i^c t_c)$.

$$\begin{aligned} T_i^p &= E(t_i^p) \\ &= E(t_i^q + P_i^c t_c) \\ &= E(t_i^q) + E(P_i^c t_c) \\ &= \mu + P_i^c t_c \end{aligned} \tag{16}$$

To calculate the average waiting time we need to identify the situations in which the frames need to wait. As shown in Figure 6, whenever the processing time exceeds a certain value, the frames need to wait. In other words, the frames need to wait whenever:

$$t_i^q > t_a - P_i^c t_c \tag{17}$$

Where t_a is inter-frame time which is given in Equation (9). This situation is illustrated in Figure 6 by the lined area. The exact waiting time would be $t_i^q - (t_a - P_i^c t_c)$. The average waiting time can be determined by calculating the probabilistic integration of the waiting times in that area:

$$T_i^w = \int_{t_a - P_i^c t_c}^{\infty} t_i^q f(t_i^q) dt_i^q \tag{18}$$

The overall average response time T can be calculated by performing probabilistic sum over all states:

$$T = \sum_{i=0}^{m} \pi_i T_i \tag{19}$$

16.5.2 Frame Drop Probability

The frame drop probability is very closely related to the processing time and the size of the buffer. Assume B is the buffer size in terms of the number of frames. So, the buffer can accommodate the frames received over time B/F_r. Now, if the processing time is such that the buffer gets full, the frames will be dropped. This is illustrated by the lined area in Figure 7. The system will drop frames when:

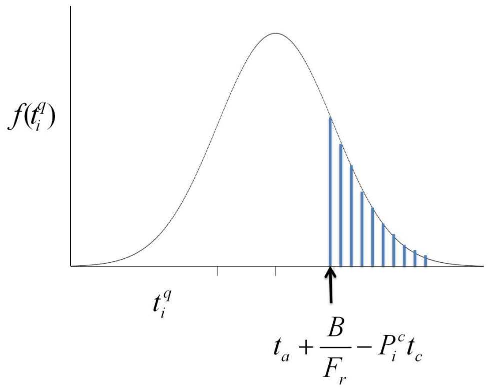


Fig. 7 The frames are dropped whenever $t_i^q > \frac{B}{F_r} + t_a - P_i^c t_c$ which is shown by lined area.

$$t_i^q > \frac{B}{F_r} + t_a - P_i^c t_c \tag{20}$$

This probability can be calculated using the probability density function of the processing time:

$$P_i = \int_{t_a + \frac{B}{F_r} - P_i^c t_c}^{\infty} f(t_i^q) dt_i^q \tag{21}$$

Now, similar to the average response time, the overall frame drop probability can be calculated as a probabilistic sum:

$$P = \sum_{i=0}^{m} \pi_i P_i \tag{22}$$

16.6 Experiments

For the evaluation of proposed model we have implemented a real surveillance system as shown in Figure 5. The system automatically detects and tracks targets. In this section we describe the hardware and software details of the implemented system and experimental results.

16.6.1 Implementation

The system is first trained to learn the background; anything that appears in the foreground is considered as the target. When there are multiple targets, the system tracks each of them individually until they disappear from the camera view. Whenever targets appear in the camera view, a new object is created. When the target leaves the camera view, the object is deleted and the tracking history is stored to disk. Table 3 lists the hardware and software details of the implemented system.

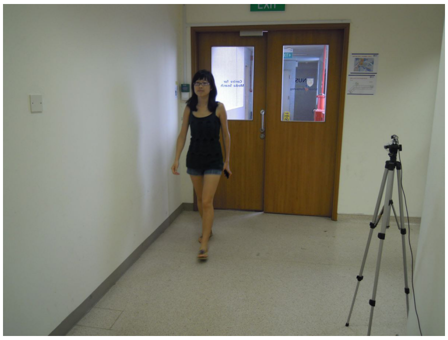


Fig. 8 System Installation.

Table 3 The specifications of the system.

Operating System	Microsoft Windows XP
Platform	Visual C++ 2008
Additional Libraries	OpenCV
Computer	Genuine Intel(R) T2300 @ 2.33GHz, 0.99GB Ram
Image Resolution	320×240

Table 4 The experimental values of state change probabilities (P_i^c), state wise mean (μ_i) and variance(σ_i)

State(i)	μ_i(milliseconds)	σ_i(milliseconds)	P_i^c
0	496	24	0.0186
1	518	30	0.0977
2	557	30	0.1097
3	662	38	0.1141
4	733	78	0.6087

The experiments are conducted in a corridor near the exit which is a common surveillance setting (See Figure 8). As the first step of modeling, we record the statistics and construct the TFG. The TFG is then used to calculate the transition and steady state probabilities. The transition probability matrix and steady state probability vector are given below:

$$\begin{pmatrix} 0.9814 & 0.0150 & 0.0026 & 0.0005 & 0.0005 \\ 0.0214 & 0.9023 & 0.0595 & 0.0149 & 0.0019 \\ 0.0077 & 0.0702 & 0.8904 & 0.0285 & 0.0033 \\ 0.0091 & 0.0183 & 0.0639 & 0.8858 & 0.0228 \\ 0.0435 & 0.1739 & 0.1304 & 0.2609 & 0.3513 \end{pmatrix} \tag{23}$$

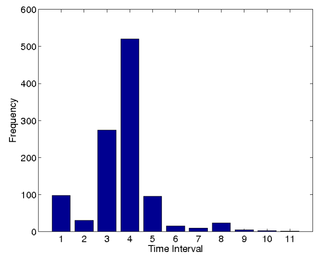
$$\begin{pmatrix} 0.4337 & 0.2474 & 0.2109 & 0.1021 & 0.0059 \end{pmatrix} \quad (24)$$

Similarly, other coefficients are calculated from the collected timing statistics. The experimental values of these coefficients are given in Table 4. The transition overhead time t_c is found to be 3.16ms.

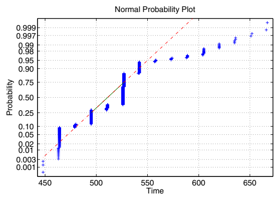
16.6.2 Hypothesis Testing: Normal Distributed Processing Time

We record the processing time statistics over a long period for a varying number of targets. To understand the behavior of the processing time within each state, we plot the histogram (Figure 9.a for state s_1 and Figure 9.c and for state s_2). To prove that the above histogram depicts a Normal distribution, we construct a *Normal Probability Plots* of the processing time data in Figure 9.b and Figure 9.d.

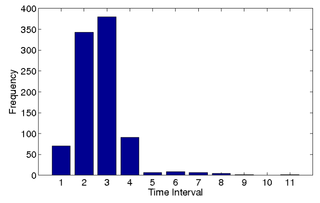
It can be observed in the normal plots that the processing time distribution is very close to Normal, except for a few cases that are likely due to the background processes running on the computer. Furthermore, we observe similar behavior in all the states. However, as anticipated, the processing time increases with state variable as shown in Figure 10. The similar behavior allows us to have a simple model for such a multi-class workload thanks to the multiple states for the Markov chain.



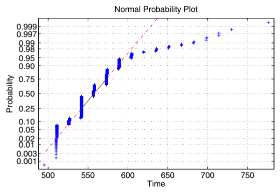
(a) Processing Time Histogram s_1



(b) Normal probability Plot for s_1



(c) Processing Time Histogram s_2



(d)Normal probability Plot for s_2

Fig. 9 Analysis of processing times.

16.6.3 Response Time

The response time is the sum of both the waiting time and the processing demand. In this experiment we assume that the size of the buffer is infinite and frames are never dropped. For each frame, we record both the waiting time and processing time. We

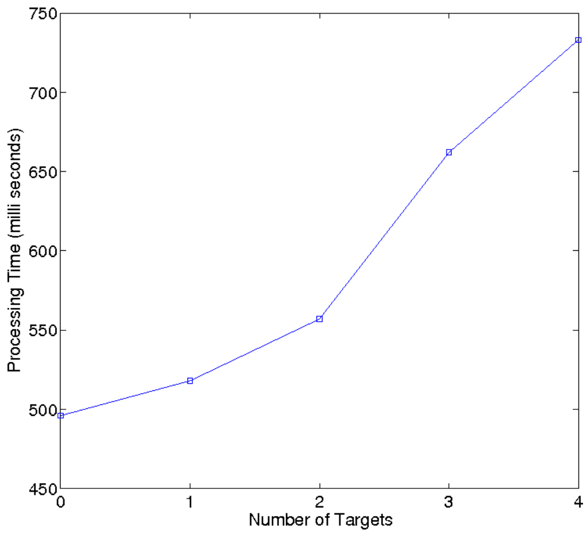


Fig. 10 The average processing time increases with number of targets in the image.

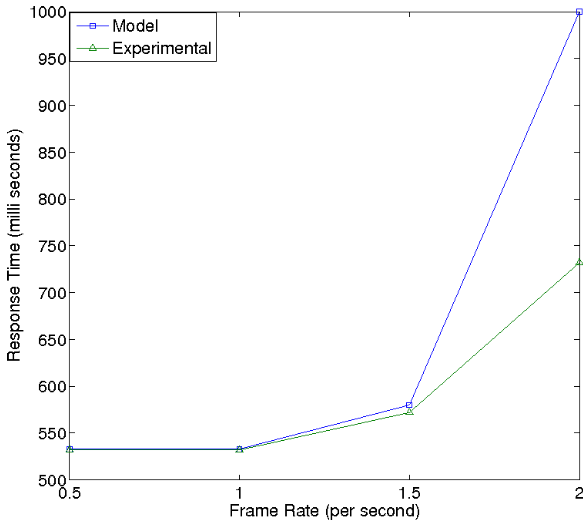


Fig. 11 The model predicted system remains close to the actual response time as long as $\mu \leq t_a$.

observe that the waiting time increases with the frame rate while the processing time remains the same. On the other hand, if we plot these timings with respect to number of targets, we observe that the waiting time and processing time both increase with number of targets.

As we observe in Figure 11, there is close agreement between the model prediction and actual measurements. For larger frame rates the model deviates because our modeling is based on the assumption that the average processing time is less than inter-frame time, which is also a necessary condition for stability.

16.6.4 Frame Drop Probability

The inter-frame time is very important for many surveillance tasks, in particular for tracking. Hence, reduced inter-frame time is always desired. In this experiment we show the effect of inter-frame time on frame drop probability. Again, it can be seen in Figure 12 that the model results are quite close to the actual results in the region where the average processing time is greater that the inter-frame time. Also, we observed that the deterministic nature of the task arrival allows us to chose a buffer size to minimize the frame drop probability. For the given system (average processing time = 593ms) and for an inter-arrival time of 600ms, even a one frame buffer brings the frame drop probability very close to zero (≈ 0.009).

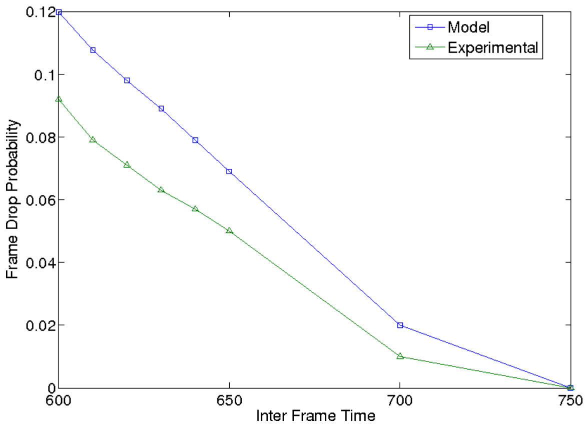


Fig. 12 The frame drop probability plotted against inter-frame time.

16.6.5 Implications

So far the current research work focuses mainly on accuracies of the algorithms with high quality images. However, the experiments reveal that understanding the workload is necessary while developing algorithms for real time surveillance systems. For instance Figure 13 shows the degradation of the tracking accuracy with increasing workload in terms of the number of targets. To cope with this situation, the designers can make use of the proposed workload model. For different states, they can have different search windows and filter coefficients.

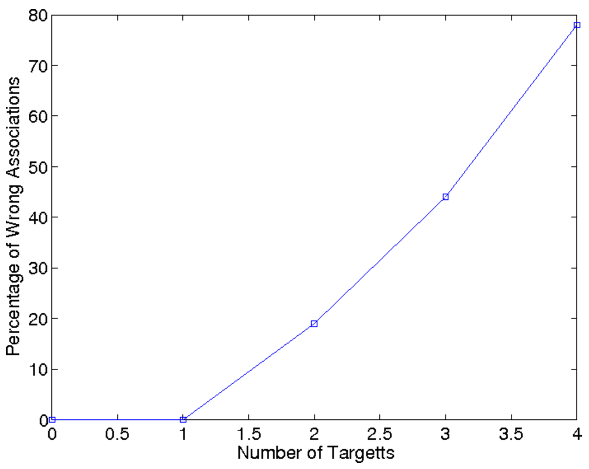


Fig. 13 The blob tracker made wrong associations while in higher states.

16.7 Conclusions and Future Work

Multimedia surveillance systems have to accomplish the tasks with high resource needs. Analytical modeling is very efficient way of studying the resource-performance trade off. The most important part of analytical modeling is workload characterization. We propose an workload model for surveillance system. The model is validated with implementation of a real surveillance system.

Based on the proposed model and the experiments, we make several observations, which are:

- The proposed workload model can help designers in determining resource requirements in a surveillance system. Designers can determine number and types of processors and the amount of memory required for a given surveillance scenario. For example, if the surveillance setup is to monitor the entry of students in a graduate lab, the processing and memory needs would be lesser in this case compared to surveillance scenario of a train station. In a graduate lab, the number of targets to process may be quite less than the crowded train station where there will be too many targets to process.
- A complex distributed surveillance system usually consists of multiple processing and memory units which process data obtained from multiple cameras. In such a situation, knowledge of workload at different units may help in designing a policy to distribute the workload so that overall waiting time can be reduced.
- Current evaluation of algorithms is usually independent of system resources availability. We observed that for the design of an algorithm for accomplishing a surveillance task, it is important to consider the amount and characteristics of the workload surveillance systems handle.

In this work we neglect the network delay and the effect of data compression. In future it would be interesting to study how the compression techniques will affect the memory demands and arrival rate. Eventually we want to evaluate the performance of more complex multimodal, multicamera, and multiprocessor system systems using our workload model.

References

1. Agarwal, A., Hennessy, J., Horowitz, M.: Cache performance of operating system and multiprogramming workloads. ACM Transactions on Computer Systems 6(4), 393–431 (1988)
2. Barford, P., Crovella, M.: Generating representative web workloads for network and server performance evaluation. In: Proceedings of ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems, Madison, Wisconsin, United States, pp. 151–160 (1998)
3. Cadez, I., Heckerman, D., Meek, C., Smyth, P., White, S.: Model-based clustering and visualization of navigation patterns on a web site. Data Mining and Knowledge Discovery 7(4), 399–424 (2003)
4. Calzarossa, M., Ferrari, D.: A sensitivity study of the clustering approach to workload modeling (extended abstract). In: Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, Austin, Texas, United States, pp. 38–39 (1985)
5. Calzarossa, M., Serazzi, G.: Construction and use of multiclass workload models. Performance Evaluation 19(4), 341–352 (1994)
6. Collins, R., Lipton, A., Kanade, T., Fujiyoshi, H., Duggins, D., Tsin, Y., Tolliver, D., Enomoto, N., Hasegawa, O.: A system for video surveillance and monitoring. Technical Report CMU-RI-TR-00-12. Robotics Institute. Carnegie Mellon University, Pittsburgh, PA (May 2000)
7. Dee, H.M., Velastin, S.: How close are we to solving the problem of automated visual surveillance? a review of real-world surveillance, scientific progress and evaluative mechanisms. Machine Vision and Applications (2007)
8. Faber, A.M., Gupta, M., Viecco, C.H.: Revisiting web server workload invariants in the context of scientific web sites. In: Proceedings of the ACM/IEEE Conference on Supercomputing, Tampa, Florida, p. 110 (2006)
9. Feitelson, D.G.: Workload modeling for performance evaluation. In: Performance Evaluation of Complex Systems: Techniques and Tools, London, UK, pp. 114–141 (2002)
10. Kelly, P.H., Katkere, A., Kuramura, D.Y., Moezzi, S., Chatterjee, S.: An architecture for multiple perspective interactive video. In: Proceedings of the third ACM International Conference on Multimedia, San Francisco, California, United States, pp. 201–212 (1995)
11. Korshunov, P., Ooi, W.T.: Critical video quality for distributed automated video surveillance. In: Proceedings of ACM International Conference on Multimedia, Hilton, Singapore, pp. 151–160 (2005)
12. Li, N., Yu, S.-Z.: Periodic hidden markov model-based workload clustering and characterization. In: IEEE International Conference on Computer and Information Technology, Sydney, Australia, pp. 378–383 (July 2008)
13. Maksyagin, A.: Modeling multimedia workloads for embedded system design: Thesis. In: ETH Zurich (2005)

14. Maxiaguine, A., Kunzli, S., Thiele, L.: Workload characterization model for tasks with variable execution demand. In: Proceedings of Design, Automation and Test in Europe Conference and Exhibition, Paris, France, vol. 2, pp. 1040–1045 (February 2004)
15. Menasce, D.A., Almeida, V.A.F., Fonseca, R., Mendes, M.A.: A methodology for workload characterization of e-commerce sites. In: Proceedings of the ACM Conference on Electronic Commerce, Denver, CO, USA, pp. 119–128 (1999)
16. Nguyen, N.T., Venkatesh, S., West, G., Bui, H.H.: Multiple camera coordination in a surveillance system. ACTA Automatica Sinica 29(3), 408–422 (2003)
17. Pan, X., Guo, Y., Men, A.: Traffic surveillance system for vehicle flow detection. In: International Conference on Computer Modeling and Simulation, Sanya, China, pp. 314–318 (2010)
18. PETS. Trec video retrieval evaluation (trecvid) (2000-2010)
19. Reed, D.A.: Queueing network models of multimicrocomputer networks. In: Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems, Minneapolis, Minnesota, United States, pp. 190–197 (1983)
20. Saini, M., Atrey, P.K., Kankanhalli, M.S.: Dynamic workload assignment in video surveillance systems. In: Proceedings of the IEEE International Conference on Multimedia and Expo, Barcelona, Spain (July 2011) (to appear)
21. Saini, M., Natraj, Y., Kankanhalli, M.: Performance modeling of multimedia surveillance systems. In: 2009 11th IEEE International Symposium on Multimedia, San Diego, California, USA, pp. 179–186 (2009)
22. Shankaran, N., Koutsoukos, X., Schmidt, D.C., Gokhale, A.: Evaluating adaptive resource management for distributed real-time embedded systems. In: Proceedings of Workshop on Reflective and Adaptive Middleware Systems, Grenoble, France (2005)
23. Sharifimehr, N., Sadaoul, S.: Markovian workload modeling for enterprise application servers. In: Proceedings of the 2nd Canadian Conference on Computer Science and Software Engineering, Montreal, Quebec, Canada, pp. 161–168 (2009)
24. Smith, K., Quelhas, P., Gatica-Perez, D.: Detecting abandoned luggage items in a public space. In: Proceedings of IEEE International Workshop on Performance Evaluation in Tracking and Surveillance, New York, pp. 75–82 (2006)
25. Song, B., Ernemann, C., Yahyapour, R.: Parallel Computer Workload Modeling with Markov Chains. In: Feitelson, D.G., Rudolph, L., Schwiegelshohn, U. (eds.) JSSPP 2004. LNCS, vol. 3277, pp. 47–62. Springer, Heidelberg (2005)
26. Sreenivasan, K., Kleinman, A.J.: On the construction of a representative synthetic workload. Commun. ACM 17(3), 127–133 (1974)
27. Venetianer, P.L., Zhang, Z., Yin, W., Lipton, A.J.: Stationary target detection using the objectvideo surveillance system. In: Proceedings of IEEE Conference on Advanced Video and Signal Based Surveillance, London, UK, pp. 242–247 (2007)
28. Wang, Q., Makaroff, D., Edwards, H.K., Thompson, R.: Workload characterization for an e-commerce web site. In: Proceedings of Conference of the Centre for Advanced Studies on Collaborative Research, Toronto, Ontario, Canada, pp. 313–327 (2003)
29. Williamson, C.L.: Network traffic measurement and modeling. In: Proceedings of ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems, Ottawa, Ontario, Canada, pp. 56–57 (1995)

Chapter 17
Rough Set and Artificial Neural Network Approach to Computational Stylistics

Urszula Stańczyk

17.1 Introduction

Computational stylistics or stylometry is a study on writing styles. Through linguistic analysis it yields observations on stylistic characteristics for authors, expressed in terms of quantifiable measures. These measures can be exploited for characterisation of writers, finding some similarities and differentiating features amongst their styles, for authorship attribution, and for recognition of documents based not on their topic, which is so common, but style. Stylistic analysis belongs with text mining, data mining, information retrieval, but also pattern recognition [4].

The fundamental stylometric notion is that of an authorial or writer invariant, such a numerical feature of texts, which allows for unique description and recognition of their authors. As the question, which elements of a text can be used as a writer invariant, remains unsettled, the task of choosing some set can be regarded as a selection of characteristic features for a classification problem [29].

Textual descriptors, which are the most universal, belong to either lexical or syntactic group [8]. Lexical markers give such characteristics as frequency of usage for single letters, words, or groups of words, average word length, average number of words in sentences, etc. Syntactic descriptors reveal the structure of sentences and paragraphs as indicated by the punctuation marks.

Theory of rough sets and artificial neural networks offer two distinctively different ways of dealing with problems of classification and recognition. The former leads to construction of algorithms consisting of decision rules, while the latter constitutes a connectionist approach, with distributed representation of knowledge and processing. Although different, both techniques work efficiently in cases of classification for data incomplete and uncertain [12, 26].

Urszula Stańczyk
Institute of Informatics, Silesian University of Technology, Akademicka 16, 44-100 Gliwice, Poland
e-mail: `urszula.stanczyk@polsl.pl`

Due to the fact that term frequencies, used in the research presented, are continuous, it is convenient to use Dominance-Based Rough Set Approach [14] rather than Classical Rough Set Approach, which deals only with abstract data [24]. A typical procedure finds relative reducts, such subsets of attributes that preserve the quality of approximation, and either minimal cover, or all rules on examples decision algorithm. Though minimal with respect to the number of rules, the first algorithm is not necessarily the best (that is resulting in the highest correct classification ratio), while the other is often too long to be efficiently implemented.

To help with dimensionality reduction, there can be proposed several orderings of characteristic features, reflecting their frequencies of usage in construction of relative reducts and decision rules, taking also into account the cardinalities of reducts and supports of rules. For each ordering there is considered reduction of more or less important features. When conclusions from rough set-based analysis are employed for selection of features for ANN, it results in a hybrid classifier [34]. For the rule-based approach it returns some shortened versions of decision algorithms, comprising only these rules that have no conditions on rejected attributes [33].

Within the chapter in Section 17.2 there are presented the fundamental notions of computational stylistics, its objectives, short history, and a brief overview of techniques employed within the analysis. Section 17.3 describes the two approaches exploited in the conducted research, artificial neural networks and rough set theory. It is followed by an explanation of the experimental setup in Section 17.4, specifying texts for processing and employed descriptors. Then the performance of both constructed classifiers is given, after which follows an analysis of characteristic features. Observations from the analysis are employed in the feature reduction task, with the results commented and compared. The chapter concludes with some remarks upon possible future research.

17.2 Basics of Computational Stylistics

Whether we speak or write, the way in which we express ourselves is individual to a high degree. This ability not only distinguishes humans from other species, but different nationalities as well, and even individuals with the same social and educational background and lifetime experiences. In a spoken language a skilled listener notices nuances of various accents, cadences, niceties of pronunciation, pitch. In handwritten notes anyone can observe different sizes and shapes of letters, or slant, while experts analyse the varying pressure of a pen, spaces between letters and words, sentences and paragraphs.

Most of these descriptive elements of a text disappear when it is prepared using some word processor. Software adjusts placement of characters to fonts, applies automatic line spacing, corrects spelling or linguistic errors, by commands such as "copy and paste" facilitates incorporation of someone else's piece of writing into our own. Yet even then a writing style shows individuality in more or less frequent usage of words or their collocations, preference of some punctuation marks over others, specific formating, organisation of the text into headings, paragraphs, footnotes.

These observations lead to the fundamental notion of computational stylistics or stylometry that any writing style can be uniquely described and recognised by not just qualitative measures, which can be considered as subjective, but quantitative ones. Due to the ever growing corpora of available texts, such measures are also required to back up the efficiency of the textual analysis, by exploiting the computational power of contemporary computers [3].

17.2.1 Objectives of Textual Analysis

The primary objective of stylometry is to express a writing style by some quantifiable features of a text. These features must be sufficiently distinct for any author as to constitute so-called *writer* or *author invariant*, such numeric characteristics that remain relatively unchanged for all works by this writer and distinctively different in documents authored by others. In the context of electronic formats of manuscripts, writer invariants are also called "cyber fingerprints", "cyberprints", or, more commonly, "writerprints".

Descriptors used in stylometric analysis are usually divided into four categories:

- lexical - provide statistics such as a total number of characters (including all letters of an alphabet, punctuation marks, and any other special characters), a total number of words, an average number of words per sentence, an average number of words per paragraph, an average number of sentences per paragraph, a distribution of word length, frequencies of usage for individual letters, frequencies of usage for words and groups of words, etc.,
- syntactic - reflect structures of sentences and paragraphs formed by the punctuation marks,
- structural - describe the overall layout of a text, its organisation into all constituent elements such as headings, signatures, paragraphs, also font types, embedded hyperlinks or pictures,
- content-specific - refer to words and phrases of specific relevance to some domain, or of higher importance in a certain context.

While it is widely acknowledged that writer invariants do exist, the task of finding them remains problematic, even though it has been debated over years and decades. This is chiefly due to the definition of an author invariant. As it is required to express individuality of a writing style, it cannot be universal, but rather relative and subjective to the particular style under study.

Many measures used are strongly dependent on the length of the processed text and so are difficult to apply reliably. Thus the choice of textual markers for the analysis is one of the crucial decisions that can greatly influence the outcome [31]. Some unreliable descriptors that poorly reflect a style can lead to completely wrong conclusions. Even the good descriptors can result in mistakes, when they are calculated over a text that is too short to be representative. The wider the corpus of processed texts, the more reliable conclusions from the analysis.

Within textual analysis there are typically distinguished three tasks:

- author characterisation - yields conclusions about a social and educational background, gender, age, focuses on such elements that betray individuality,
- author comparison - dedicated to establishing, if they exist, some shared properties and similarities for texts by different authors,
- author attribution - answers the question of authorship for unattributed or disputed texts.

Out of these three aims, authorship attribution is considered as of the highest importance, yet without the other two it would be impossible to achieve it. Only when characterised properly and compared against others, a style can be correctly attributed to its author.

Stylometric analysis is most often used to detect cases of plagiarism, to establish authorship for anonymously published or disputed texts, or in criminal investigations within forensic linguistics area, for example to confirm an identity of a terrorist threat-maker.

17.2.2 Short Historical Overview

Contemporary computational stylistics is considered as a successor of historical textual analysis that exploited human abilities of detecting patterns and similarities, making associations, and dedicated to proving or disproving the authenticity of legal documents or literary works.

Probably the most famous example from this early era is proving the forgery of the Donation of Constantine, giving the western part of the Roman Empire to Pope Sylvester. In 1439 Lorenzo Valla compared Latin used in other documents that were dated to the IVth Century, and whose originality was beyond question, with Latin used in the Donation of Constantine and concluded it to be falsified [27].

These early attempts involved tedious and extremely time-consuming study of manuscripts, comparing them against others, and could use only some striking features of texts, which are not entirely reliable. Anyone can notice them and that means not theoretic but real possibility of imitating someone else's writing style.

The new course for stylometry was set in 1787 when Edmond Malone, an expert on Shakespeare's plays, argued the usage of quantitative over qualitative descriptors, such as meter and rhyme. In 1851 Augustus de Morgan proposed the usage of average word lengths, which was made famous by the study of T.C. Mendenhall on word-length distributions printed in 1887. Then followed the research by Morton and Yule (1938 and 1965), who used sentence lengths as descriptive features in authorship studies.

Apart from these theoretical considerations, the second part of the XXth Century brought also the rapid development of computers and their ever increasing computational powers enabled a much wider spectrum of methodologies and a variety of textual descriptors to be efficiently employed for text processing [10].

17.2.3 *Methodologies Employed*

Apart from the choice of descriptors to be used in the analysis, the decision about the methodology to be employed is another crucial point in the stylometric processing [9]. In fact, these two issues should not be considered separately. With the question of finding a writer invariant remaining open, the popular approach is to study descriptors in the context of a processing technique used, making them not only task-dependent but also methodology-dependent. Such attitude changes the perspective in which textual markers are perceived, which is all the more advantageous when taking into account the inherent properties of establishing the importance of features that most methodologies possess [29].

Modern stylometric analysis involves processing procedures that belong with computer-aided statistic-oriented computations, or techniques from artificial intelligence domain [2]. All these methodologies can be employed just by themselves giving satisfactory results, yet it is also possible to apply some hybrid solutions. These approaches, where the fusion of processing techniques is performed, can put the studied characteristic features in a new light, and result in some advantageous observations leading to improved performance.

17.2.3.1 Statistical Approaches

Statistic-oriented processing requires computations of probabilities and distributions of occurrences for letters, punctuation marks, and other special characters, words, groups of words, patterns of sentences, and distribution of probabilities for comparative analysis among works of known and unknown authorship [20, 28].

In the probabilistic model of natural language all constituent elements do not appear at random, in fact all letters occur with some probability. They depend on characters that precede and succeed them. Thus a text can be considered as a sequence of characters corresponding to a Markov chain of some order [17]. For the simplest model only the immediate predecessor is considered, which leads to the Markov chain of the 1st order. For all pairs of letters in an alphabet, and all special characters, there are calculated matrices of frequencies, reflecting transitions of a letter into all studied characters. These matrices are found for all texts by known authors and compared with the one for an unattributed text. The true author is selected as the one with the highest probability.

Jill M. Farrington has invented a method called QSUM or CUSUM, where there is calculated the cumulative sum (hence the name of the method) for two features. As the first of these features there is studied the sentence length. All deviations from the average are plotted for the whole text of known authorship. For the second feature there is selected the usage of 2- or 3-letter words, using words that start with a vowel, or a combination of both. Next, the same calculations are performed and plotted for a text of unknown authorship, and if their graphs match, the writer is identified [7].

Popular Linear Discriminant Analysis, Principal Component Analysis, or cluster analysis are examples of multivariate methods that aim to find some relationships amongst input data, which, upon some calculation, lead to dimensionality reduction by looking for linear combinations of variables that best explain data, or partitioning data into subsets described by some distance measure. When procedures applied both to already attributed and these unattributed texts return the same results, the identity of the author is revealed.

Most of statistic-based approaches involve high computational complexity computations, which are especially cumbersome when dealing with large datasets or when an exhaustive search is required.

17.2.3.2 Machine Learning Approaches

Machine learning algorithms are characterised by their relatively high efficiency while processing large datasets. They are used in feature extraction process and achieve high accuracy in classification tasks, which makes them particularly well suited for stylometric analysis.

The processing with genetic algorithms starts with a definition of some set of rules that express characteristics of studied texts. Next these rules are tested against a set of known texts and each is given a fitness score leading to selection operation. Some rules with low scores are discarded and some rules with highest scores kept. The remaining rules are slightly modified within mutation and some new rules added. The whole process is repeated until reaching the point when the evolved rules correctly attribute texts to their authors.

Applying artificial neural networks in stylometric processing begins with constructing a network with random weights associated with interconnections. Next the network is tested against the training samples for already attributed texts and weights adjusted as long as classification is incorrect. Once the network is trained it can be used for authorship recognition for unknown texts [36].

Rough set approach constitutes an example of rule-based methodology used in cases when input data is uncertain and incomplete. It perceives the Universe by granules of knowledge that lead to lower and upper approximations of sets. Within the processing, basing on the decision table there are constructed algorithms consisting of induced decision rules. In their premise parts the rules specify conditions on the considered features, to be satisfied for a decision to be applicable [24].

Both connectionist ANN classifier and rule-based approach of rough sets are described in more detail in the following Section 17.3.

17.3 Connnectionist and Rule-Based Classification

Connectionist and rule-based classifiers are commonly presented as two opposite approaches to the problems of classification and recognition. The former detects subtle relationships amongst input datasets and offers distributed processing and representation of knowledge, while the latter observes dominant patterns that are succinctly expressed by decision algorithms listing rules of "If... then..." type. Even though so different, both methodologies are efficiently employed in data analysis [25, 1], on their own, or combined together which results in hybrid solutions.

17.3.1 Artificial Neural Networks

For decades nervous systems present in biological organisms have been the subject of interest and study for mathematicians, trying to develop models which would describe these systems and their complexities. Artificial neural networks, as known today, have emerged as generalisations of these concepts [12]. The description of the mathematical model of the artificial neuron due to McCuloch and Pitts appeared in 1943, Hebb defined the unsupervised learning rule in 1949, while Rosenblatt's perceptron was implemented for the first time in 1958.

The applicability of ANN to computational tasks and their efficiency have been questioned many times. Especially in their early days the book "Perceptrons", published in 1969 by Minsky and Papert, caused dissipation of the initial enthusiasm and interest. Only when the backpropagation learning algorithm for supervised learning was documented in 1980s, ANN regained their status and proved to be a satisfactory solution to many problems.

A network can be perceived as a computing system that consists of some number of rather simple processing units - neurons, which are interconnected and work in parallel. The network possesses the inherent ability to learn and adapt, to generalise, allows for some fault tolerance, provides distributed knowledge representation.

A specification of ANN comprises definitions of a set of neurons giving their number and organisation, neuron activation states expressed by activation functions and offsets that specify when the neurons fire, interconnections between neurons that through their weights indicate how the output signal of a neuron affects other neurons, and the learning or training rule that shows the method of gathering information by the network.

17.3.1.1 Network Topology

Depending on topology, artificial neural networks are divided into feed-forward, with the data flow strictly from the input to output neurons organised into layers, and recurrent that contain feedback loops. In pattern classification tasks the former type is most popularly used.

An example of a feed-forward network constitutes Multilayer Perceptron (MLP), constructed from some number of layers and possessing unidirectional weighted connections between neurons. The detailed structure, that is the specific number of neurons and layers is to some extent task-dependent.

The number of input and output nodes of a network can be seen as its external specification. For classification purposes there are as many input neurons as characteristic features defined for the analysed objects, and typically the number of outputs reflects the number of distinguished classes.

The number of hidden layers and neurons is crucial to classification ability and accuracy. Without any hidden layers a network can only solve problems that are linearly separable. A single hidden layer results in the network classifying simplexes, while two hidden layers enable classification of any objects. When the number of neurons in hidden layers is higher than necessary, the network learns quickly but performs poorly for unknown data when generalisation is needed. On the other hand, when there are too few neurons in hidden layers, the network has trouble converging and may never learn the relationships amongst the input data.

With such vague indicators it is a commonly used approach to build a network with some initial number of cells and possibly layers, and then train and test the network. Depending on the network performance these numbers are next either decreased or increased.

17.3.1.2 Activation Function

Transfer or activation functions for neurons define when they fire, that is how they react to data accumulated through all weighted inputs. Typically there is used: linear or semi-linear function, a hard limiting threshold function, a sigmoid, or a hyperbolic tangent. Functional properties, such as linearity, continuity, differentiability, cause the functions to perform with varying efficiency in task-specific solutions.

Sigmoid is the most popularly used activation function in the classification tasks. It is non-linear, continuous, differentiable, and it is defined as:

$$y(n) = \frac{1}{1+e^{-\beta n}} \tag{1}$$

$$n = \mathbf{W} \cdot \mathbf{X} = \sum_{j=0}^{J} w_j x_j \tag{2}$$

where **W** is the weight vector, **X** the input vector, and $j = 0$ is reserved for offset t, by setting $w_0 = -t$ and $x_0 = 1$.

17.3.1.3 Learning Rule

To learn, which is the desired set of output states whenever a set of input data values is presented to an artificial neural network, it needs to be able to adjust the weights of interconnections and this is encompassed by the network learning or training rule.

There are employed supervised, unsupervised, or reinforcement learning rules. In Multilayer Perceptron usually there are applied some variants of the supervised backpropagation method. The term *supervised* means that there is a kind of external teacher whose role is to provide information about the desired answers for all training samples. The training data is given by means of pairs of the input values and expected outputs. When the expected output values are compared against these obtained by the network, the error function can be calculated as follows:

$$e(\mathbf{W}) = \frac{1}{2} \sum_{m=1}^{M} \sum_{i=1}^{I} (d_i^m - y_i^m(\mathbf{W}))^2 \tag{3}$$

It is a sum of errors for all M training facts on all output neurons, each defined by the difference between the expected outcome d_i^m and the one generated by the network $y_i^m(\mathbf{W})$. Minimisation of this function leads to adjustment of connection weights in such a way that results in the output values to be closest to expected, for each of the learning facts and for the whole learning set.

The classical backpropagation algorithm modifies the vector of weights $\mathbf{W}$ as indicated by the direction of the steepest descent of the gradient:

$$\Delta \mathbf{W} = -\eta \nabla e(\mathbf{W}) \tag{4}$$

η is the learning rate and its value is an important parameter in this algorithm. When it is too high, it can cause oscillations around the local minima of the error function. When it is too low, it results in slow convergence. This locality is considered to be a drawback of the algorithm, but its universality is its advantage and a reason for being so widely used.

17.3.2 Rough Set Theory

Rough set theory was developed by Zdzislaw Pawlak in the early 1980s, to deal with the problem of imperfect knowledge, incomplete and uncertain [26]. This problem has been studied by scientists for many years and to interpret and manipulate such knowledge many ways have been proposed, probably the most popular of which is fuzzy set theory due to Lotfi Zadeh.

In classical set theory elements are either included or not included in a set. In rough set theory using the available information about objects in the Universe the space is partitioned into granules of knowledge within which single objects cannot be discerned. This leads to lower and upper approximations of sets and construction of a decision algorithm. Constituent decision rules in their premise parts list all conditions on attributes that must be met for the decision to be applicable.

Classical Rough Set Approach (CRSA) with its indiscernibility relation works only for abstract or discrete data, which allows only for nominal classification [11]. In cases when the input datasets show ordinal properties, there can be employed Dominance-Based Rough Set Approach (DRSA) that substitutes the indiscernibility with dominance relation.

17.3.2.1 Dominance Relation and Set Approximations

In classification tasks the attributes $q \in Q$ that describe objects of the Universe U are divided into condition C and decision D ones. In decision making problems usually there are several condition attributes (called also criteria) and a single decision attribute. The values of attributes for all objects are contained in a decision table. When these values show monotonic properties the indiscernibility relation can be insufficient for efficient decision making [15, 13].

The indiscernibility principle can only say that when two objects are indiscernible with respect to the considered attributes, they should be classified to the same class. Thus in CRSA the granules of knowledge are the equivalence classes of objects that cannot be discerned.

On the other hand, the *dominance* or *Pareto principle* advocates that if, with respect to the considered attributes, for two objects x and y, x is at least as good as y, then x should be classified at least as good as y.

Let $\succeq_q$ be a weak preference relation that represents a preference on the set of objects, with respect to some criterion q. When for all $q \in P$, $P \subseteq C$, $x \succeq_q y$, then x dominates y with respect to P, which is denoted as xD_Py.

In DRSA the granules of knowledge are: a set of objects dominating x (P-dominating set), and a set of objects dominated by x (P-dominated set), defined as follows:

$$\begin{aligned} D_P^+(x) &= \{y \in U : yD_Px\} \\ D_P^-(x) &= \{y \in U : xD_Py\} \end{aligned} \tag{5}$$

A single attribute $D = \{d\}$ partitions the Universe into some finite number of classes $\boldsymbol{Cl} = \{Cl_t\}$, with $t = 1, \ldots, n$. The classes are ordered and the increasing indices indicate increasing preference, due to which the sets to be approximated are upward or downward unions of classes (dominance cones), defined respectively as:

$$\begin{aligned} Cl_t^{\geq} &= \bigcup_{s \geq t} Cl_s \\ Cl_t^{\leq} &= \bigcup_{s \leq t} Cl_s \end{aligned} \tag{6}$$

P-lower approximation of $Cl_t^{\geq}$, denoted as $\underline{P}(Cl_t^{\geq})$, is the set of objects that belong without any ambiguity to $Cl_t^{\geq}$. P-upper approximation of $Cl_t^{\geq}$, denoted as $\overline{P}(Cl_t^{\geq})$, is the set of objects that could belong to $Cl_t^{\geq}$, which is defined as follows:

$$\begin{aligned} \underline{P}(Cl_t^{\geq}) &= \{x \in U : D_P^+(x) \subseteq Cl_t^{\geq}\} \\ \overline{P}(Cl_t^{\geq}) &= \{x \in U : D_P^-(x) \cap Cl_t^{\geq} \neq \emptyset\} \end{aligned} \tag{7}$$

The boundary regions of $Cl_t^{\geq}$ and $Cl_t^{\leq}$ with respect to P, denoted as $Bn_P(Cl_t^{\geq})$ and $Bn_P(Cl_t^{\leq})$ respectively, are defined by the differences between the upper and lower approximations:

$$\begin{aligned} Bn_P(Cl_t^{\geq}) &= \overline{P}(Cl_t^{\geq}) - \underline{P}(Cl_t^{\geq}) \\ Bn_P(Cl_t^{\leq}) &= \overline{P}(Cl_t^{\leq}) - \underline{P}(Cl_t^{\leq}) \end{aligned} \tag{8}$$

17.3.2.2 Relative Reducts

Data contained in the decision table can be expressed in more succinct way by the inherent mechanism of the rough set theory such as the concept of a *relative reduct*.

Quality of approximation of $\boldsymbol{Cl}$ by criteria P, $P \subseteq C$, can be defined as:

$$\gamma_P(\boldsymbol{Cl}) = \frac{\left| \left(U - \left(\bigcup_{t \in \{2,\ldots,n\}} Bn_P(Cl_t^{\geq}) \right) \right) \right|}{|U|} \tag{9}$$

A relative reduct, RED_{Cl}, is such an irreducible subset $P \subseteq C$ for which $\gamma_P(\boldsymbol{Cl}) = \gamma_C(\boldsymbol{Cl})$, that is the quality of approximation remains the same even though only the selected subset of condition attributes is considered. A decision table can have many reducts. The intersection of reducts is called a *relative core*, $CORE_{Cl}$.

Relative reducts betray the importance of individual attributes for the classification process [22]. If some attribute does not belong to any reduct, it means that it can be completely disregarded from considerations. When an attribute belongs to the core, it can never be discarded. With the empty core and multitude of reducts the choice of one for the processing is not a trivial task, and may require some domain knowledge.

17.3.2.3 Construction of Decision Algorithms

Finding approximations of dominance cones is the starting point for induction of rules to be included in a decision algorithm [15]. There are distinguished several types of decision rules:

- certain $D_{\geq}$-rules - supported by objects belonging to $Cl_t^{\geq}$ without any ambiguity,
- possible $D_{\geq}$-rules - supported by objects belonging to $Cl_t^{\geq}$ with or without ambiguity,

- certain $D_{\leq}$-rules - supported by objects belonging to $Cl_t^{\leq}$ without any ambiguity,
- possible $D_{\leq}$-rules - supported by objects belonging to $Cl_t^{\leq}$ with or without ambiguity,
- approximate $D_{\geq\leq}$-rules - supported by objects belonging to $Cl_s \cup Cl_{s+1} \cup \ldots \cup Cl_t$ without possibility of discerning classes.

A set of decision rules is called *complete* when, according to it, no object of the decision table remains unclassified. A set of rules is *minimal* when it is complete and irreducible, that is exclusion of any rule makes the set incomplete [35].

The minimal set of rules ensures covering the training samples, yet the testing set can vary to such degree that for some samples there are no rules matching. It is especially true for the multidimensional input space. To increase the probability of higher classification accuracy, instead of providing a minimal cover when constructing decision algorithms, there are also tried approaches generating all rules on examples. However, it means more time-consuming processing and often results in an unmanageably high number of decision rules found. In such case an optimised classifier can be built, comprising a selection of rules, basing on their support, length, or some assumed weights. A rule support states how many training samples match this rule and its higher value means that it is more likely for the rule to match the testing examples. When the rules are long, it may indicate overfitting the learning data that results in poor generalisation unto unknown examples.

17.4 Experimental Setup

Stylometric analysis can be seen as a multistage and even repetitive process. The first stage is the selection of texts for training and testing samples. The second stage, the choice of textual descriptors, can be seen as connected or dependent on the decision as to the processing technique to be employed that comes next. Then follows the stage of calculating characteristics for all texts. Depending on the methodology selected, some auxiliary processing (for example such as discreatisation of values) can be required before the actual stylometric analysis is performed. Next, the results from the analysis are tested, conclusions drawn, and then, possibly, the whole procedure is executed once again. The repetitive performance enables better adjustment of some parameters, such as including more or discarding some of characteristic features, using modified versions of algorithms, or employing hybrid solutions, incorporating findings from the former analysis.

17.4.1 Input Datasets

Written works can be extremely dissimilar. They can significantly vary in length, address numerous subjects, use various registers. Over a long time span, many changes can be observed in any language. Forms change, spelling differs, some words become anachronisms while new ones are created. Comparing some manuscripts that in most aspects show no resemblance makes little sense. Without some measure of similarity of the analysed documents, the unavoidable conclusion is the one that is already known - that they differ, regardless of the processing technique applied.

The most convenient field of application for computational stylistics is offered by literature, as the corpus of texts available for analysis is wide enough to give reliable descriptors. Still the works have different lengths, yet this issue is easily dealt with by dividing the long texts into some smaller parts, of approximately the same size.

It can be argued that all stylistic features can be influenced by a text genre and it is undeniably true. Plays or poems are governed by the completely different rules than narratives, crime and mystery stories read rather unlike memoirs. While the comparison of writings in prose with these in verse would be in fact difficult, or even impossible, when the considerations are limited to one type it is possible to determine such textual markers that work as writer invariants, and enable author attribution, by referring to the most commonly used elements of language. As they are used to some degree subconsciously, and accordingly to the individual writing habits, the patterns formed are likely to appear no matter what an author writes about and thus they can be exploited to recognise their unique style [3].

17.4.1.1 Texts for Analysis

To illustrate the concepts and elements of stylometric analysis, as texts for processing there were selected works by Edith Wharton and Jane Austen (listed in Table 1), available due to Project Gutenberg (`http://www.gutenberg.org`). Their novels provide the corpora wide enough for the characteristic features, found basing on the training data, to be representative of other texts. This generalised knowledge can be employed to either discount or confirm the possibility of any of the considered writers being recognised as the author of a text of unknown origin.

As within an author characterisation stylometric task there is included determination of gender [19], it is not a coincidence that both selected authors are female. Their novels contain dialogs and narrative parts, and on one hand show some similarities while on the other are diversified enough to constitute a basis for the analysis.

In the learning and testing set there were included 200 and 90 samples respectively, corresponding to parts of comparable lengths for several works for each writer. Such approach enables discarding from the considerations the influence of a text length upon its style. Each sample is several pages long.

Table 1 Works by Edith Wharton and Jane Austen selected for the analysis.

a) Learning set

Author	Title
Edith Wharton	A Backward Glance
	The Age of Innocence
	The Glimpses of the Moon
	The Reef
Jane Austen	Emma
	Mansfield Park
	Persuasion
	Pride and Prejudice

b) Testing set

Author	Title
Edith Wharton	Certain People
	House of Mirth
	Summer
Jane Austen	Northanger Abbey
	Love and Friendship
	Sense and Sensibility

17.4.1.2 Selection of Characteristic Features and Processing Techniques

For the texts selected for stylistic analysis, structural and content-specific markers could not be used reliably as author invariants. All texts contain no specific formatting and the vocabulary is too diversified to look for words of some key meaning [6]. That leaves lexical and syntactic descriptors.

Syntactic markers reflect the structure of sentences as organised by the punctuation marks [5]. Some authors create simple sentences while other prefer complex ones, some writers use many commas while others employ both commas and colons, one author inserts a comment in brackets and another puts commas around it, some emphasis can be expressed with a question or exclamation mark. All this depends on an individual style of writing, thus to work as textual markers there were considered frequencies of usage for 8 punctuation marks: a fullstop, a comma, a question mark, an exclamation mark, a semicolon, a colon, a bracket, a hyphen.

Lexical descriptors seem to be the most natural ones to be employed. They give averages or frequencies of usage for single letters, words, or groups of words. To minimise the influence of vocabulary on the author recognition there were selected 17 entries form the list of the most popular words in English language as follows: but, and, not, in, with, on, at, of, this, as, that, what, from, by, for, to, if.

The resulting set of 8 syntactic+17 lexical = 25 descriptors has proven its efficiency in authorship attribution studies [33, 34].

For processing there were selected two techniques from the artificial intelligence domain: the connectionist approach of artificial neural networks and rule-based approach of rough set theory. As frequencies obtained are continuous values, they are not directly applicable in Classical Rough Set Approach that works on abstract or discrete data. In such case it is possible to use some discretisation strategy by defining a discretisation factor [32]. On the other hand, the data is ready for Dominance-Based Rough Set Approach since clearly there is ordering in value sets.

However, from the stylometric point of view it cannot be definitely stated how attributes are *preference* ordered as it would imply some a priori knowledge that some greater or lower frequency is preferable to others as characteristics for the considered writers.

17.4.2 Connectionist Classification

Artificial neural networks can be simulated with dedicated hardware or software. In the research described here there was employed the latter approach, implemented by California Scientific Brainmaker.

As the base topology of an artificial neural network, there was used the feed-forward Multilayer Perceptron with the sigmoid activation function, trained by the classical backpropagation algorithm. The number of inputs was equal to the number of textual descriptors used and there were two outputs to reflect the two recognised authors.

To establish the best network structure, that is with the highest classification accuracy for the task, there were conducted experiments with varying the numbers of hidden layers and neurons in them. Since initiation of interconnections weights can greatly influence the training results, there was employed the multi-starting approach with the learning procedure repeated 20 times. The results are given in Table 2, and there is specified not only the average performance for each network configuration but also minimal and maximal classification accuracies.

Table 2 Performance of ANN classifier in relation to the network structure.

Number of hidden layers											
	0			1					2		
Number of neurons in hidden layers											
							25	25	14	21	19
	0	27	25	14	13	7	25	2	13	6	6
Classification accuracy [%]											
Minimal	90,00	90,00	90,00	90,00	90,00	90,00	90,00	90,00	88,89	90,00	90,00
Median	90,00	91,11	91,11	91,11	91,11	91,11	91,11	91,11	91,11	91,11	91,11
Maximal	91,11	92,22	91,11	92,22	92,22	92,22	93,33	92,22	91,11	92,22	93,33

These results are very similar but for two hidden layers the maximal classification accuracy is the highest. With several network layers usually there is adopted the structure of an inverted pyramid: the input layer is the most numerous, then the numbers of neurons decrease as the layers get closer to the output. Thus the rightmost column specifies the structure selected and employed within further research dedicated to dimensionality reduction, with incorporating the analysis of the significance of individual characteristic features for the task, described in Sec. 17.4.5.

17.4.3 Rule-Based Classification

DRSA methodology requires indication of a preference order for all attributes, both condition and decision ones. Stylistics domain knowledge is insufficient to answer this question in some universal way, therefore another approach must be employed.

For just two classes to be recognised for a single decision attribute, there are only two possibilities of preference ordering, but for 25 condition attributes there are as many as 2^{25}, which in total gives 2^{26} different preference orderings. Trying all of them and testing to find the best would be far too time-consuming. Some detailed analysis of values for all learning samples could betray the preferences for writers, but they are not necessarily conclusive. In the presented experiments the preference order was established arbitrarily for all attributes.

When the efficiency of rule-based classifiers obtained is verified with testing samples, the results are given in three categories: correct decisions, incorrect decisions, and ambiguous decision. The last category corresponds to cases of no matching rules, but can also mean that there are several contradicting decisions. Then for the final verdict for a sample, the decision can be based on the majority of decisions by voting, or supports of constituent rules, yet such approach can lengthen the processing time even several times, especially when decision algorithms comprise high numbers of decision rules. To avoid this additional processing, all ambiguous decisions were treated as incorrect.

The details of DRSA processing are given in Table 3. The number of relative reducts was almost two thousand with cardinalities varying from 2 to 10, yet the relative core was empty. The classification with minimal cover algorithm gave unsatisfactory results of only 51%, thus all rules on examples algorithm was calculated, returning 52,108 constituent decision rules. With such high number of rules, when there are no constraints on them, decisions for all testing samples are ambiguous. The highest classification accuracy was 74% of correct decisions and it was obtained when limiting the set of rules by the values of their support.

The threshold support on rules resulting in the highest classification accuracy was for 47, 48, 49 and 51, giving the number of rules equal to 35, 30, 23 and 12

Table 3 Details of DRSA processing involving all attributes.

Relative reducts	
Number of reducts	1979
Reduct cardinalities	from 2 to 10
Relative core	empty
Union of reducts	complete set of attributes
Minimal cover algorithm	
Number or rules	55
Rule lengths	from 1 to 3
Rule supports	from 1 to 43
Classification accuracy	51%
All rules on examples algorithm	
Number or rules	52,108
Rule lengths	from 1 to 9
Rule supports	from 1 to 61
Threshold support	51
Number of rules in shortened algorithm	12
Classification accuracy	74%

respectively. This shortest decision algorithm, comprising just four rules classifying to "edith" class and eight rules with decisions "jane", is as follows, the numbers at the end of each rule giving its support.

All rules on examples algorithm (25 attributes and 52,108 rules, for threshold support≥51 limited to 12 rules involving 10 attributes, classification accuracy 74%)

```
Rule 1235. (what >= 0.002147) & (; >= 0.002863)
         & (- >= 0.016106) => edith [53]
Rule 3442. (? >= 0.003632) & (; >= 0.003148)
         & (- >= 0.018644) => edith [54]
Rule 6345. (on >= 0.004737) & (what >= 0.00203)
         & (; >= 0.003384) => edith [55]
Rule 18877. (on >= 0.007042) & (- >= 0.015649)
         => edith [52]
Rule 21807. (in <= 0.01701) & (; <= 0.002195)
         => jane [61]
Rule 33044. (and <= 0.035352) & (in <= 0.016907)
         &  (; <= 0.002459) => jane [53]
Rule 33791. (and <= 0.039362) & (in <= 0.016722)
         &  (; <= 0.002315) => jane [59]
Rule 34047. (and <= 0.037153) & (at <= 0.007326)
         &  (of <= 0.036368) & (; <= 0.00314)
         => jane [54]
Rule 34341. (at <= 0.008142) & (; <= 0.003016)
         &  ( ( <= 0.000603) => jane [60]
Rule 36278. (at <= 0.008724) & ( ( <= 0.001651)
         &  (- <= 0.010611) => jane [54]
Rule 49807. (in <= 0.017019) & (; <= 0.001761)
         => jane [54]
Rule 51590. (; <= 0.00142) & ( ( <= 0.00142)
         => jane [51]
```

Fig. 1 shows how the classification accuracies of decision algorithms change with the increasing values of threshold support limiting rules included, and how it reflects upon the number of remaining rules.

When the support required of rules is low, their high number works against their classification accuracy: there are too many ambiguous decisions. The critical point is crossed for support equal at least 32 which limits the rules to 301, and they return 51% of correct decisions. Then, along with the support, the correct recognition ratio increases till 74%, to decrease when there are too few rules left.

When the results of rule-based classifier are compared against the ones obtained for the connectionist solution, it may seem that they are much worse with 74% against 91%, yet it should be taken into account that the distance between the two numbers could be smaller if, instead of just discarding all ambiguous decisions, they were further processed to arrive at either correct or incorrect verdict.

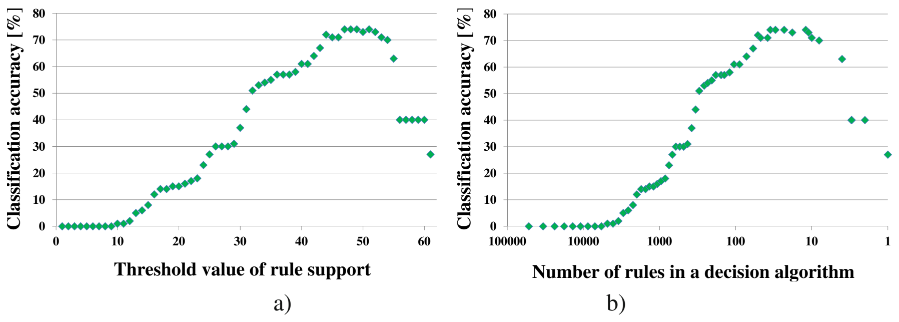


Fig. 1 Classification accuracy in relation to: a) threshold values of support imposed on the rules to be included in a decision algorithm, b) the numbers of rules included in decision algorithms with limited support (displayed in logarithmic scale).

17.4.4 Analysis of Characteristic Features

Both processing methodologies employed possess their own ways of establishing the importance of individual features for the classification task considered.

In artificial neural networks this is achieved by the training phase when interconnection weights are adjusted to arrive at the minimum of the error function. For the already trained network, by some form of sensitivity analysis it is possible to find these inputs that have relatively small influence on the network outputs, which in turn can lead to pruning. Pruning can involve the interconnections with low weights, but also some neurons in the hidden layers, or inputs. Due to the connectionist nature, these reduction techniques typically require time-consuming and complex calculations [18, 16].

In rough set approach selection of relatives reducts assumes various significance of some attributes, which influences decision rules constructed. Once they are induced, still some features can be disregarded by exploiting not the full but limited algorithm, with selection of rules based on some assumed measures [21].

To obtain some importance indicators for characteristic features, there can be also employed elements of frequency analysis. It is the study of occurrence frequencies for single letters or their groups, popularly exploited within cryptography to construct or break ciphers. Basing on the wide corpus of texts of varied register and style for a language, it is possible to compute how often, in relation to all, particular letters and words are used, which leads to ordering by their rank. Once this order is known the encrypted message can be read even when some substitution or transposition, or some more complex ciphers are applied [23].

In the considered context there were studied occurrences of attributes within the calculated relative reducts and decision rules, included in the all rules on examples algorithm. A basic analysis can involve just these frequencies of usage, yet such attitude means treating all reducts and all rules alike, regardless of their parameters.

For relative reducts their cardinalities can be also taken under consideration within the analysis. Lower cardinality means that fewer attributes express the same knowledge, hence it can be concluded that attributes belonging to such reducts are more important than others which require a lot of company to form a reduct. The distribution of reduct cardinalities for individual attributes is specified in Table 4.

The three right-most columns in Table 4 give values for three quality indicators for features. QI_N reflects directly the number of reducts in which each condition attribute is included. QI_1 is calculated as a sum of quotients, where divisors are cardinalities of reducts and dividends the numbers of occurrences in reducts of such cardinalities. With such calculation each reduct is assumed to have an influence factor that is an inverse of its cardinality. QI_2 is computed as a sum of products, each reduct cardinality multiplied by the number of reducts with such cardinality in which each feature is included. Thus while QI_1 assumes greater significance of reducts with lower cardinalities, QI_2 does the same for these with higher cardinalities.

Next the focus was limited to reducts with the lowest four cardinalities, namely 2, 3, 4, and 5, and the frequencies of usage in such reducts for all attributes.

These four ways of interpreting the importance for the classification criteria lead to their ordering accordingly to the obtained values of considered indicators, as

Table 4 Analysis of attributes based on relative reducts and their cardinalities.

	Number of relative reducts with specific cardinality												
	Total	1	8	123	459	693	465	181	47	2	Quality indicators		
Attribute	1979	2	3	4	5	6	7	8	9	10	QI_N	QI_1	QI_2
of	818	0	3	18	189	290	224	75	18	1	818	135,11	5106
with	726	0	0	20	108	284	207	88	18	1	726	116,60	4649
at	712	0	1	31	135	181	213	122	27	2	712	114,13	4618
!	711	0	1	8	117	237	232	91	24	1	711	112,52	4620
if	705	0	0	19	142	319	167	39	19	0	705	117,16	4352
as	690	0	0	15	96	227	202	111	37	2	690	107,83	4557
and	682	0	0	11	101	257	186	98	28	1	682	107,82	4439
.	652	0	1	13	134	247	162	75	20	0	652	106,29	4121
in	620	0	0	39	181	235	100	54	11	0	620	107,37	3702
from	609	0	1	24	106	208	165	77	27	1	609	98,50	3901
to	525	0	1	18	99	154	168	63	22	0	525	84,62	3372
that	521	0	2	50	106	186	123	48	6	0	521	89,60	3151
what	515	0	0	5	77	168	180	66	18	1	515	80,71	3373
?	510	0	2	23	120	219	97	40	9	0	510	86,77	3092
for	503	0	0	3	64	150	166	86	32	2	503	76,77	3390
-	456	1	3	47	220	152	33	0	0	0	456	87,30	2442
on	447	1	3	35	85	162	113	41	6	1	447	76,28	2731
but	372	0	0	3	6	87	162	85	28	1	372	53,43	2640
by	330	0	0	7	45	110	91	53	23	1	330	51,36	2191
this	290	0	0	5	43	92	77	53	18	2	290	45,01	1932
;	274	0	6	86	79	67	36	0	0	0	274	55,61	1411
(	235	0	0	10	21	72	87	26	18	1	235	36,48	1566
not	102	0	0	0	3	16	32	39	11	1	102	14,04	756
:	77	0	0	2	16	21	22	14	1	1	77	12,30	499
,	35	0	0	0	2	17	10	4	2	0	35	5,38	232

given in Table 5. Within each ordering presented there could be distinguished the increasing and decreasing order, with increasing or decreasing values of the considered measures, labelled with "L" or "M" respectively. "L" label means focus on more important (with respect to the considered indicator) features while disregarding these less important. "M" signifies discarding more important features while keeping these that are less important.

A similar analysis was also performed for the calculated decision rules. In the analysis there were taken under consideration the frequencies of usage for all condition attributes in the induced rules as well as supports of these rules, distribution of which is given in Table 6. All rules can and have some influence upon the classification accuracy, but when they have higher supports they are more likely to express some patterns present not only in the learning samples, but also in testing ones. That is why the support can be treated as an indicator of importance for the rules and the attributes included in them.

To reflect these considerations, there were observed two orderings of condition attributes based on decision rules, one giving the numbers of rules including

Table 5 Ordering of characteristic features based on analysis of relative reducts and their cardinalities.

Order RDN			Order RD1			Order RD2			Order RD3		
of			of			of			-		
with	M1		if	M1		with	M1		on		
at			with			!			;	M1	
!			at			at			of	M2	L10
if		L12	!		L10	as		L15	that	M3	
as	M2		as	M2		and	M2		?		L9
and		L11	and			if		L14	at	M4	
.	M3	L10	in			.	M3	L13	from		L8
in	M4		.		L9	from	M4	L12	to	M5	
from		L9	from	M3	L8	in	M5	L11	.		L7
to	M5		that	M4		for	M6		!	M6	L6
that			-			what			in	M7	L5
what			?		L7	to		L10	with	M8	
?			to	M5		that	M7		if		
for		L8	what		L6	?		L9	as		
-	M6		for	M6		on	M8		and		
on		L7	on		L5	but		L8	(		L4
but	M7	L6	;	M7		-	M9	L7	by	M9	L3
by	M8	L5	but			by	M10	L6	what	M10	
this	M9		by		L4	this	M11	L5	this		L2
;		L4	this	M8	L3	(	M12	L4	for		
(		L3	(		L2	;		L3	but		
not		L2	not			not		L2	:		L1
:		L1	:		L1	:		L1	not		
,			,			,			,		

Table 6 Distribution of rule supports for condition attributes.

Number of	with support in specific range						Maximal	Total number
decision rules	48026	3015	731	247	77	12	support	of rules
Attribute	1-10	11-20	21-30	31-40	41-50	51-61	61	52108
in	11138	723	125	47	21	4	61	12058
;	3818	762	290	127	46	10	61	5053
at	11514	754	153	38	11	3	60	12473
(	9202	709	130	38	11	3	60	10093
and	12203	518	114	33	3	3	59	12874
on	6429	1020	286	79	34	2	55	7850
what	9994	387	49	27	7	2	55	10466
of	12724	599	152	63	13	1	54	13552
-	5324	693	221	94	31	4	54	6367
?	7409	717	150	33	11	1	54	8321
that	9313	810	171	56	15	0	49	10365
from	11140	521	96	26	6	0	48	11789
if	9114	494	76	13	1	0	47	9698
.	9865	386	66	17	3	0	45	10337
by	9423	254	57	12	2	0	45	9748
,	4200	34	5	1	1	0	44	4241
with	12078	708	137	12	2	0	42	12937
this	9386	212	24	1	1	0	42	9624
to	10027	320	53	15	1	0	42	10416
but	8546	151	17	0	1	0	41	8715
as	10133	263	39	3	1	0	41	10439
!	10832	252	35	6	0	0	39	11125
for	9644	193	21	7	0	0	37	9865
:	2472	13	4	0	0	0	29	2489
not	524	1	0	0	0	0	11	525

attributes regardless of rule support, and another based on the maximal support of rules, as shown in Table 7.

The presented orderings of condition attributes were next employed to construct modified versions of both connectionist and rule-based classifiers, with reduced numbers of characteristic features. For ANN the new structures involved also decreasing the number of neurons in hidden layers, in order to keep the inverted pyramid structure. In DRSA processing new algorithms were found by removing these decision rules from the complete set, which in their premise parts contained conditions on the discarded attributes. The performance for all classifiers was compared against results obtained previously for the whole set of attributes, and trends present observed, as described in more detail in the next section.

Table 7 Ordering of characteristic features based on the number of decision rules in which the features are included, and supports of these rules.

Order RLN			Order RS1		
of			in		
with	M1		;		
and			at	M1	
at	M2		(		
in	M3		and	M2	L13
from	M4	L13	on	M3	
!	M5	L12	what		L12
what	M6		of	M4	
as			-		
to			?		L11
that			that	M5	L10
.		L11	from	M6	L9
(	M7	L10	if	M7	L8
for	M8	L9	.	M8	
by	M9		by		L7
if			,	M9	L6
this		L8	with	M10	
but	M10	L7	this		
?	M11	L6	to		L5
on	M12	L5	but	M11	
-		L4	as		L4
;		L3	!		L3
,		L2	for		L2
:		L1	:		L1
not			not		

17.4.5 Performance for Feature Reduction

Dimensionality reduction can be attempted for a variety of reasons. It can be the only way to deal with a problem otherwise unsolvable because of unmanageable amounts of data. It can lead to implementation of a classifier that works faster, is more efficient, cheaper. It can help to increase the correct recognition ratio. But it also leads to more detailed analysis of the characteristic features, not always necessary, yet revealing more about the significance of features for the considered task.

Application of elements from DRSA methodology in dimensionality reduction for a connectionist classifier means creating a hybrid classifier. Such hybrid solutions have become quite popular. Not only do they often perform better, but offer also another perspective, from which the features can be studied. It is highly informative to observe whether the importance of features established for one technique can be transferred to another, especially so distinctively different. Connectionist and rule-based approaches are typically presented as opposites. ANN detects subtle relationships in the input datasets while DRSA tries to find these most striking.

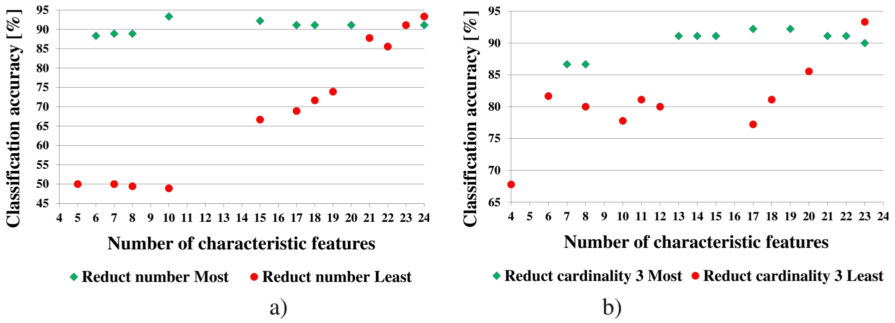


Fig. 2 ANN classification accuracy in relation to the number of features, with their reduction based on: a) numbers of reducts in which they are included regardless of their cardinalities (Order RDN), b) numbers of reducts of the lowest cardinalities in which the attributes are included (Order RD3).

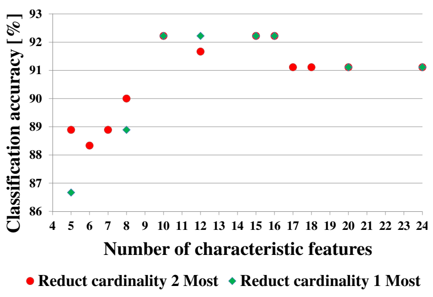
When there are considered the relative reducts and how many times each characteristic feature is used to construct them, reflected by QI_N and Order RDN, the results of reduction of attributes (presented in Fig. 2a) lead to the immediate conclusion: the features less often included in reducts are more important for ANN classifier. When they are disregarded, the performance worsens - the classification accuracy barely reaches 50% when there are fewer than ten inputs left. In contrast, when there are discarded the attributes most often exploited in construction of reducts, the network performance remains at the same or slightly increased level, as long as there are at least ten remaining inputs. Below ten, it decreases just by 2%, even when there are as few as five inputs left.

When the focus is limited to the relative reducts with the lowest cardinalities as in Order RD3, from the observation of feature reduction results given in Fig. 2b, it can be inferred that without losing the power of the classifier we can discard several of the features (12 out of 25 makes almost 50% of inputs) most often included in the reducts with low cardinalities. Rejecting the condition attributes least often present in such small reducts decreases the classification accuracy, yet not that dramatically as in the previous case depicted in Fig. 2a.

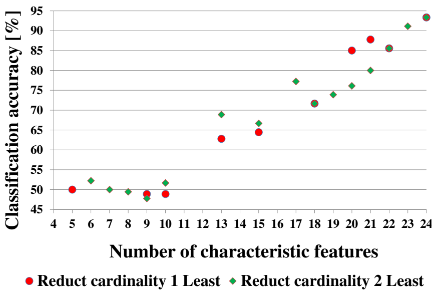
When the consideration is given to all relative reducts and their respective cardinalities, either assuming that these with lower cardinalities are more important (Order RD1) or that these with higher cardinalities are more significant (Order RD2), from the results of dimensionality reduction shown in Fig. 3, it can be induced that for both orderings the connectionist classifier performance is similar, and in both cases it is safer to remove more often exploited features.

All these tests lead to a conclusion that the importance of condition attributes, as observed within the analysis of relative reducts found and their cardinalities, should be given an inverse meaning when processing with a connectionist classifier.

Even though for this hybrid classifier the increase in the correct recognition ratio is so small (by 2%) that cannot be treated as statistically significant, the reduction of characteristic features is around 50%.



a)



b)

Fig. 3 ANN classification accuracy in relation to the number of features, with their reduction for Order RD1 and Order RD2: a) from more significant side, b) from less significant side.

Moving the analysis of features to the rough set domain, there are constructed limited decision algorithms. The rules included in them are only those that have no conditions upon the discarded attributes. When the decision rules are numerous, even when some are rejected, still the number remains high and there are many ambiguous decisions. To minimise their number the threshold support of the rules is required, limiting the algorithms even further, and making their length to be of manageable proportions.

Performance of a rule-based classifier, with feature reduction based on the number of reducts in which the condition attributes are employed (Order RDN), is specified in Table 8. There are given the numbers of attributes remaining, the numbers of rules within the limited algorithms, the threshold support values required to achieve the highest classification accuracy, the numbers of rules that meet these criteria, and these maximal classification accuracies. All details are listed for the reduction of either most (M-labelled columns) or least (L-labelled columns) significant features.

While rejecting the more significant attributes, the number of remaining decision rules falls rapidly, yet with decreasing the required threshold value of support, there are 76% correct decisions for 10 attributes discarded (40%). On the other hand, for removing the less significant features the classification accuracy is kept only at the beginning of the reduction process, when the 12 rules from the full algorithm that give the highest accuracy remain intact. Once they are rejected along with the attributes, the performance worsens.

When the reduction is based on relative reducts with lowest cardinalities (results shown in Table 9), setting aside the more important features again unsurprisingly quickly reduces the number of decision rules, and the classification accuracy steadily decreases. For the less significant features the rate of descent in the number of rules is much slower, while the correct recognition ratio can be kept at the same level with lowering of the imposed support, for reduction of up to 30% of attributes. Then it falls down, only to increase to 78% for just four features left.

Both groups of tests indicate that the condition attributes most frequently included in relative reducts stand much better chance at maintaining the power of the rule-based classifier than these features that are used seldom.

Table 8 Classification results for reduced decision algorithms. Reduction of attributes based on the number of reducts in which they are included, regardless of their cardinality (Order RDN).

	Number of attributes		Number of rules		Threshold support		Nr of rules in short alg.		Classification accuracy [%]	
Algorithm	M	L	M	L	M	L	M	L	M	L
01	24	24	38556	47867	49	51	22	12	73	74
02	20	23	12966	45653	44	51	36	12	77	74
03	18	22	6651	45170	45	51	30	12	76	74
04	17	21	5127	35626	45	47	29	29	76	71
05	15	19	2145	25404	46	38	20	48	76	65
06	10	18	470	20154	29	38	44	48	73	65
07	8	17	141	16543	4	38	94	47	51	65
08	7	15	124	11236	4	23	92	27	51	42
09	6	10	67	2589	4	11	51	47	50	47
10		8		1098		7		94		53
11		7		652		6		80		41
12		5		297		4		111		40

Table 9 Classification results for reduced decision algorithms. Reduction of attributes based on the maximal cardinalities of relative reducts in which they are included (Order RD3).

	Number of attributes		Number of rules		Threshold support		Nr of rules in short alg.		Classification accuracy [%]	
Algorithm	M	L	M	L	M	L	M	L	M	L
01	23	23	38345	47368	40	51	37	12	75	74
02	22	20	34373	30579	23	51	47	12	54	74
03	21	18	24620	19982	23	48	41	27	56	74
04	19	17	14611	16141	15	48	40	27	46	74
05	17	12	7110	3498	10	47	99	22	43	66
06	15	11	3146	2145	6	41	238	35	25	70
07	14	10	1963	1705	6	41	172	35	27	70
08	13	8	1146	956	6	41	93	32	30	66
09	8	6	47	373	1	29	47	85	0	70
10	7	4	30	171	1	6	30	147	0	78

Exploiting the analysis of decision rules, dimensionality reduction can be governed by their number for each of the considered criteria, reflected by Order RLN. The results are listed in Table 10, and their examination reveals that when the more important, that is more frequently exploited, features are set aside, after the initial slight decrease, the classification accuracy remains steadily high even when there are just 6 condition attributes left. In contrast, when reducing the less frequently used attributes, only at the beginning of reduction the classification accuracy is preserved, then it gradually decreases.

These observations may seem rather surprising and contrary to those for reduction based on reducts, yet such conclusion would be premature. To understand these

Table 10 Classification results for reduced decision algorithms. Reduction of attributes based on the number of decision rules in which they are included, regardless of their properties (Order RLN).

	Number of attributes		Number of rules		Threshold support		Nr of rules in short alg.		Classification accuracy [%]	
Algorithm	M	L	M	L	M	L	M	L	M	L
01	24	24	38556	51583	49	51	22	12	73	74
02	22	23	19733	49111	49	51	20	12	73	74
03	21	22	14084	45170	45	51	31	12	76	74
04	20	21	8874	40924	46	36	21	80	76	66
05	19	20	6325	35540	46	33	21	33	76	57
06	18	19	5076	29592	46	23	21	47	76	54
07	13	18	1195	23977	39	16	29	100	76	45
08	12	17	779	19287	22	16	70	96	75	45
09	11	14	587	9182	22	16	70	71	75	40
10	8	13	155	7246	23	16	48	71	75	40
11	7	12	122	5022	23	12	48	106	75	35
12	6	7	86	589	9	2	58	420	74	34
13		6		289		2		244		40

results it should be remembered that here there are considered all decision rules, regardless of their supports. As it happens, most rules have low supports. Out of 52,108 rules over 92% have supports in the lowest range, between 1 and 10. The rules with such low supports reflect patterns present in few of the learning samples, which are less likely to be found in the testing set than the patterns described by decision rules with higher supports.

While limiting considerations to the maximal supports of decision rules in which the attributes are included, as shown in Table 11, the results fully confirm the intuitive expectations. Disregarding the features from rules with the highest maximal supports results in the immediate decrease in the classification accuracy, while keeping them, and removing those in rules with lower supports, gives almost constant level of classification accuracy, even when reduction reaches over 70% of features.

It could be argued that application of this last approach has no merit, as the same version of the decision algorithm can be obtained by taking all rules on examples algorithm with its full set of calculated decision rules, and imposing some threshold support that gives the highest number of correct decisions. As listed in Sec. 17.4.3, this limited algorithm contains 12 rules involving 10 attributes, it is obtained for support equal at least 51, and returns 74% correct decisions. Undeniably the same 12 rules give correct recognition here, but using just a support for restricting decision rules, without reduction of features, does not explain the results for the 12th algorithm. It works on just 7 attributes, and returns 77% of correct decisions for the threshold support required equal at least 32. For all rules on examples algorithm this value of support is the previously indicated crucial point, when 301 rules ensure 51% recognition. Reduction of features based on the maximal support of rules

Table 11 Classification results for reduced decision algorithms. Reduction of attributes based on the maximal support of decision rules in which they are included (Order RS1).

	Number of attributes		Number of rules		Threshold support		Nr of rules in short alg.		Classification accuracy [%]	
Algorithm	M	L	M	L	M	L	M	L	M	L
01	23	24	35830	51583	36	51	55	12	68	74
02	21	23	20390	49111	28	51	108	12	71	74
03	20	22	13724	39741	28	51	94	12	68	74
04	18	21	8971	31165	16	51	202	12	55	74
05	15	19	2763	20610	6	51	224	12	31	74
06	14	16	1770	9852	6	51	99	12	34	74
07	13	15	1140	9161	4	51	120	12	28	74
08	12	13	716	5744	3	51	114	12	24	74
09	10	12	266	4229	1	51	266	12	15	74
10	9	11	240	3090	1	51	240	12	14	74
11	6	10	20	2148	1	51	20	12	0	74
12		7		661		32		51		77
13		5		197		17		67		60

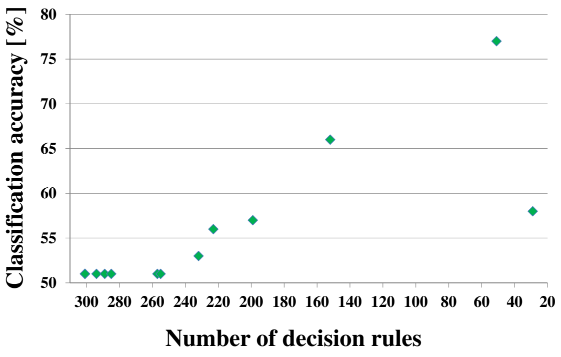


Fig. 4 Classification accuracy in relation to the numbers of rules included in the decision algorithms. Reduction of features based on the maximal support of rules in witch they are included (Order RS1), for threshold support equal at least 32.

enables restricting these 301 rules, which causes the gradual increase in the classification accuracy, as shown in Fig. 4.

Actually, it is only due to the fact that this shortest algorithm of the highest classification accuracy is so atypically short (consists of so few constituent decision rules), that it is not further reduced within the feature reduction procedure exploiting rule supports. For longer algorithms some rules can be set aside, without the change for the worse in the classifier performance.

17.5 Conclusions and Future Research

Both connectionist (ANN) and rule-based (DRSA) classifiers perform with the satisfying accuracy in an authorship attribution task of computational stylistics, for the selected lexical and syntactic markers working as effective style discriminators. The descriptors correspond to frequencies of usage for some function words and punctuation marks. Such selection of characteristic features is explained by rather subconscious habits of applying these common elements of language by authors, therefore they are less likely and more difficult to be imitated.

The significance of the selected features for the considered classification and recognition task is studied by applying elements of simple frequency analysis to the relative reducts and decision rules calculated within DRSA procedures. The analysis leads to assigning some quality indicators for each of the characteristic features, and provides a basis for their ordering. These observations are exploited to advantage within the feature reduction process for both processing methodologies, without additional computationally complex calculations, which are so common in dimensionality reduction.

When these rough set-based quality indicators are used for pruning the inputs of an artificial neural network, it turns out that in the connectionist context their meaning is inverted: the attributes more important for DRSA processing are less important for ANN. While staying within the rough set domain, the values of the quality indicators confirm the significance of condition attributes.

Within future research there will be taken into consideration yet another parameter of the induced decision rules: their length. When rules are long and contain many conditions on attributes, it may result in overfitting the data, when the rules so closely describe the learning examples that they lose their generalisation ability and cannot fit the testing or new data. Thus rule length distributions for the individual attributes could be studied in the context of their selection for both connectionist and rule-based classifier.

Also, instead of an arbitrary choice for a preference order assumed for each attribute within DRSA processing, some analysis of value sets could be executed, leading to an informed decision, which can influence not only the processing time but also, which is more important, the performance of the constructed classifier.

Furthermore, each computational stylistics task requires a new selection of descriptors to work as efficient discriminators for the studied texts. Thus for an author characterisation and comparison, which would be useful for example in automatic categorisation of texts, some new sets of textual markers should be found.

On the other hand, the presented methodology of establishing the importance of characteristic features for the classification task, which can be exploited for dimensionality reduction, should be applied in some different domain in order to confirm its efficiency when working on datasets of different type.

Acknowledgements. All texts used in the experiments performed are available in electronic form thanks to Project Gutenberg (`http://www.gutenberg.org`). 4eMka Software used in search for relative reducts and decision rules [14, 30] was downloaded in 2008 from

the website of Laboratory of Intelligent Decision Support Systems, (`http://www-idss.cs.put.poznan.pl/`), Poznan University of Technology, Poland.

References

1. Abraham, A., Falcón, R., Bello, R. (eds.): Rough Set Theory: A True Landmark in Data Analysis. SCI, vol. 174. Springer, Berlin (2009)
2. Ahonen, H., Heinonen, O., Klemettinen, M., Verkamo, A.: Applying data mining techniques in text analysis. Tech. Rep. C-1997-23, Department of Computer Science, University of Helsinki, Finland (1997)
3. Argamon, S., Burns, K., Dubnov, S. (eds.): The structure of style: Algorithmic approaches to understanding manner and meaning. Springer, Berlin (2010)
4. Argamon, S., Karlgren, J., Shanahan, J.: Stylistic analysis of text for information access. In: Proceedings of the 28th International ACM Conference on Research and Development in Information Retrieval, Brazil (2005)
5. Baayen, H., van Haltern, H., Tweedie, F.: Outside the cave of shadows: using syntactic annotation to enhance authorship attribution. Literary and Linguistic Computing 11(3), 121–132 (1996)
6. Berber Sardinha, T.: Using key words in text analysis: practical aspects (1999), `ftp://ftp.liv.ac.uk/pub/linguistics`
7. Buckland, W.: Forensic semiotics. The Semiotics Review of Books 10(3) (1999)
8. Burrows, J.: Textual analysis. In: Schreibman, S., Siemens, R., Unsworth, J. (eds.) A Companion to Digital Humanities. Blackwell, Oxford (2004)
9. Craig, H.: Stylistic analysis and authorship studies. In: Schreibman, S., Siemens, R., Unsworth, J. (eds.) A Companion to Digital Humanities. Blackwell, Oxford (2004)
10. Dale, R., Somers, H., Moisl, H. (eds.): Handbook of natural language processing. CRC Press (2000)
11. Deuntsch, I., Gediga, G.: Rough set data analysis: A road to noninvasive knowledge discovery. Mathoδos Publishers, Bangor (2000)
12. Fiesler, E., Beale, R.: Handbook of neural computation. Oxford University Press (1997)
13. Greco, S., Matarazzo, B., Slowinski, R.: Rough set theory for multicriteria decision analysis. European Journal of Operational Research 129(1), 1–47 (2001)
14. Greco, S., Matarazzo, B., Słowiński, R.: Dominance-Based Rough Set Approach as a Proper Way of Handling Graduality in Rough Set Theory. In: Peters, J.F., Skowron, A., Marek, V.W., Orłowska, E., Słowiński, R., Ziarko, W.P. (eds.) Transactions on Rough Sets VII. LNCS, vol. 4400, pp. 36–52. Springer, Heidelberg (2007)
15. Greco, S., Słowiński, R., Stefanowski, J., Żurawski, M.: Incremental versus Non-Incremental Rule Induction for Multicriteria Classification. In: Peters, J.F., Skowron, A., Dubois, D., Grzymała-Busse, J.W., Inuiguchi, M., Polkowski, L. (eds.) Transactions on Rough Sets II. LNCS, vol. 3135, pp. 33–53. Springer, Heidelberg (2004)
16. Kavzoglu, T., Mather, P.: Assessing artificial neural network pruning algorithms. In: Proceedings of the 24th Annual Conference and Exhibition of the Remote Sensing Society, Greenwich, UK, pp. 603–609 (2011)
17. Khmelev, D., Tweedie, F.: Using Markov chains for identification of writers. Literary and Linguistic Computing 16(4), 299–307 (2001)
18. Kingston, G., Maier, H., Lambert, M.: A statistical input pruning method for artificial neural networks used in environmental modelling. In: Transactions of the 2nd Biennial Meeting of the International Environmental Modelling and Software Society, Osnabrueck, Germany, pp. 87–92 (2004)

19. Koppel, M., Argamon, S., Shimoni, A.: Automatically categorizing written texts by author gender. Literary and Linguistic Computing 17(4), 401–412 (2002)
20. Lynam, T., Clarke, C., Cormack, G.: Information extraction with term frequencies. In: Proceedings of the Human Language Technology Conference, San Diego, USA, pp. 1–4 (2001)
21. Moshkov, M.J., Piliszczuk, M., Zielosko, B.: On Partial Covers, Reducts and Decision Rules with Weights. In: Peters, J.F., Skowron, A., Düntsch, I., Grzymała-Busse, J.W., Orłowska, E., Polkowski, L. (eds.) Transactions on Rough Sets VI. LNCS, vol. 4374, pp. 211–246. Springer, Heidelberg (2007)
22. Moshkov, M.J., Skowron, A., Suraj, Z.: On Covering Attribute Sets by Reducts. In: Kryszkiewicz, M., Peters, J.F., Rybiński, H., Skowron, A. (eds.) RSEISP 2007. LNCS (LNAI), vol. 4585, pp. 175–180. Springer, Heidelberg (2007)
23. Munro, R.: A queing-theory model of word frequency distributions. In: Proceedings of the 1st Australasian Language Technology Workshop, Melbourne, Australia, pp. 1–8 (2003)
24. Pawlak, Z.: Rough sets. International Journal of Computer and Information Sciences 11(5), 341–356 (1982)
25. Pawlak, Z.: Computing, artificial intelligence and information technology: Rough sets, decision algorithms and Bayes' theorem. European Journal of Operational Research 136, 181–189 (2002)
26. Pawlak, Z.: Rough sets and intelligent data analysis. Information Sciences 147, 1–12 (2002)
27. Peng, R.: Statistical aspects of literary style. Bachelor's Thesis, Yale University (1999)
28. Peng, R., Hengartner, H.: Quantitative analysis of literary styles. The American Statistician 56(3), 15–38 (2002)
29. Shen, Q.: Rough Feature Selection for Intelligent Classifiers. In: Peters, J.F., Skowron, A., Marek, V.W., Orłowska, E., Słowiński, R., Ziarko, W.P. (eds.) Transactions on Rough Sets VII. LNCS, vol. 4400, pp. 244–255. Springer, Heidelberg (2007)
30. Słowiński, R., Greco, S., Matarazzo, B.: Dominance-Based Rough Set Approach to Reasoning About Ordinal Data. In: Kryszkiewicz, M., Peters, J.F., Rybiński, H., Skowron, A. (eds.) RSEISP 2007. LNCS (LNAI), vol. 4585, pp. 5–11. Springer, Heidelberg (2007)
31. Stanczyk, U.: Dominance-Based Rough Set Approach Employed in Search of Authorial Invariants. In: Kurzynski, M., Wozniak, M. (eds.) Computer Recognition Systems 3. AISC, vol. 57, pp. 293–301. Springer, Heidelberg (2009)
32. Stańczyk, U.: Relative Reduct-Based Selection of Features for ANN Classifier. In: Cyran, K.A., Kozielski, S., Peters, J.F., Stańczyk, U., Wakulicz-Deja, A. (eds.) Man-Machine Interactions. AISC, vol. 59, pp. 335–344. Springer, Heidelberg (2009)
33. Stańczyk, U.: DRSA Decision Algorithm Analysis in Stylometric Processing of Literary Texts. In: Szczuka, M., Kryszkiewicz, M., Ramanna, S., Jensen, R., Hu, Q. (eds.) RSCTC 2010. LNCS(LNAI), vol. 6086, pp. 600–609. Springer, Heidelberg (2010)
34. Stańczyk, U.: Rough Set-Based Analysis of Characteristic Features for ANN Classifier. In: Graña Romay, M., Corchado, E., Garcia Sebastian, M.T. (eds.) HAIS 2010. LNCS(LNAI), vol. 6076, pp. 565–572. Springer, Heidelberg (2010)
35. Stefanowski, J.: On Combined Classifiers, Rule Induction and Rough Sets. In: Peters, J.F., Skowron, A., Düntsch, I., Grzymała-Busse, J.W., Orłowska, E., Polkowski, L. (eds.) Transactions on Rough Sets VI. LNCS, vol. 4374, pp. 329–350. Springer, Heidelberg (2007)
36. Waugh, S., Adams, A., Tweedie, F.: Computational stylistics using artificial neural networks. Literary and Linguistic Computing 15(2), 187–198 (2000)

Chapter 18
Application of Learning Algorithms to Image Spam Evolution

Shruti Wakade, Kathy J. Liszka, and Chien-Chung Chan

Abstract. Spam filters have become very proficient at identifying text spam, so spammers have developed different techniques to bypass filters. One such method is image spam, which first appeared in 2005 and quickly grew in popularity. KnujOn is a web site that collects and sorts spam for investigations and data analysis of email-based threats. We have been collecting image spam from KnujOn on a daily basis since April 2008, culminating in a significantly large corpus of real data. In this chapter, we have identified eight features for the detection of computer generated image spam versus ham (non-spam). We use J48 and J48 with reduced error pruning decision trees to classify the images. Finally, we perform a validation by feature analysis on thirteen months of our corpus and observe that our classification scheme is not affected by changes made to images for the purpose of avoiding OCR detection.

Keywords: spam, image spam, decision trees, classification, feature analysis.

18.1 Introduction

Email is an integral part of our lives. The consequence of this convenience is spending time every day dealing with unsolicited advertisements for Viagra, Rolex watches, low-interest loans and other products. There are many ways spammers harvest your email address even if you never open spam email and click on suspicious links. Joining a social network without setting your privacy settings ensures your data will be available to anyone. This includes your location, email, and friend lists. Newsgroups are fertile ground for collecting email addresses. Dictionary attacks are one of the oldest techniques for harvesting email addresses. Once there is an adequate list of names, botnets are created to infect machines owned by unsuspecting users to send spam.

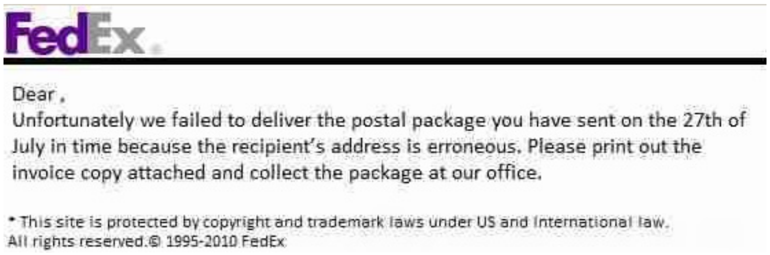
Shruti Wakade · Kathy J. Liszka · Chien-Chung Chan
Department of Computer Science, University of Akron, Akron, OH, 44325-4003, USA
e-mail: {liszka,chan}@uakron.edu

Spam filters and blacklists are fairly adept at identifying spam but spammers relentlessly develop new techniques to trick their way into our inboxes. Originally, spam was sent as text in the body of an email. Bayesian filtering and other techniques are fairly effective in fighting these types of messages. Enter image spam and the game changes.

This research uses machine learning tools to identify image spam versus image attachments that are non-spam, or more commonly known as ham. Image spam, in this context, is a picture that is computer generated as opposed to photographs. There are emails from companies, such as Home Depot, that send an advertisement in the form of a digital photo, such as solar lighting. Although we may not be interested in purchasing their products, this type of image is not the target of our research. Figure 1 (a) shows an example of this type. Figure 1(b) shows a computer generated advertisement generated for the purpose of avoiding a spam filter.

We identify a set of features in image spam and apply J48 plus pruning to create a classifier for spam detection. Our training set includes a number of trends that spammers have used to avoid detection by filters over the past several years. We add to the body of knowledge in this area by identifying why these features are effective, transcending the evolution of image spam generation techniques. The remaining chapter is organized as follows. In section 2 we discuss current research in the area of applying machine learning algorithms to this problem. Section 3 describes the image corpus and trends observed over the time span from

(a) Photographic advertisement.



FedEx

Dear ,
Unfortunately we failed to deliver the postal package you have sent on the 27th of July in time because the recipient's address is erroneous. Please print out the invoice copy attached and collect the package at our office.

* This site is protected by copyright and trademark laws under US and International law.
All rights reserved.© 1995-2010 FedEx

(b) Computer generated image spam.

Fig. 1 Image spam versus photographic advertising.

April 2008 through January 2011. Section 4 presents the features selected, preprocessing steps and the actual feature extraction. Section 5 presents the results of classifying spam and non-spam images. In section 6, we do a validation by feature analysis on thirteen months and 50,000 spam images, to show that these are, indeed, a set of features that can be used to effectively classify ham versus spam. Finally conclusions and future are given in Section 7.

18.2 Related Work

E-mail spam started as simple text-based messages. Early spam filtering solutions based on machine learning tools and natural language processing techniques have been shown to be quite effective [1, 2, 3, 4, 5, 6]. However, it is known that spammers have been reactive in devising attacks to every successful filtering method [7]. Evolved from text spam, image spam has been used to successfully pass through text-based spam filters. It follows that many learning-based methods have been introduced for image spam filtering. In the following, we briefly review some methods that are related to current work. For more comprehensive survey and review of these methods, we refer to the work by [8, 9].

One of the important issues in image-based spam filtering is how to identify and extract effective features to represent spam images to be used as training examples for learning algorithms. In [10], the Artificial Neural Networks (ANN) approach has been used for identifying image spam, where training images were first preprocessed by converting them into normalized gray scale values in the range of 0 to 1. An artificial neural network was then trained on these images using a supervised learning approach, and finally the ANN was tested for classification of new samples of spam images. A classification accuracy of 70% was reported.

The Support Vector Machine (SVM) approach has been used by [11, 12, 13] to classify spam images using low-level features such as image width, height, aspect ratio, file size, compression and image area. All the features were extracted from the header of an image file. Other features used include number of colors, variance, most appearing number of different colors in the image, primary color of the image and color saturation.

Wang et al. [12] also computed the grey histogram of different gray scale values and used binary feature values to indicate type of file like JPEG or BMP or PNG. They reported accuracy over 95% in their experiment.

Text detection from embedded images is a technique used by Aradhye et al. [11]. After the text was extracted, they applied a text-based method to analyze the text and computed two other features color saturation and color heterogeneity. Color saturation was defined in the same way as in [14], and color heterogeneity was calculated by scaling the original image by maximum possible intensity such that the intensities in the RGB channels are within the range [0, 1]. It was then converted to an indexed image using minimum variance quantization such that the number of colors in the indexed image was at most k (in their case 8 and 10). They reported an accuracy of 85% of their method.

Krasser et al. [13] used the C4.5 classifiers to classify the images, and their results indicate that SVM algorithms perform better than C4.5 as it has a larger area under the Receiver Operating Characteristic (ROC) curve [15]. However, one of the advantages of decision tree based image spam filters is that it is easy to incorporate efficient Just In Time (JIT) feature extraction, namely, features can be extracted only when needed as demonstrated in [16].

He et al. [14] used low level properties along with the grey and color histograms as features. They then set a threshold for these properties and used a two-step classification strategy. In the first step, images are regarded as possible spam if the tested properties exceeded the threshold value. The possible candidates are further evaluated in the second step where the histogram similarity of test image and threshold value was compared. The advantage of this two-step classification was that the first step trapped many of the spam image files. They reported an efficiency of about 68% in JPEG images and 23% in GIF images for step 1, and 84% for JPEG and 80% for GIF were achieved in step 2.

Gao et al. [17] used a similarity measure on color histograms and gradient orientation histograms to cluster spam images using agglomerative hierarchical clustering algorithm [18]. They selected a training set from the clustered groups and built a probabilistic boosting tree (PBT) on the training set to distinguish spam images from ham images. They reported an accuracy of about 89%.

In this research we propose eight features which are computed based on the pixels in an image. J48 and RepTree of Weka [19] are used as classification algorithms to classify spam images from ham. The decision tree classifier J48 is a Java implementation of Quinlan's C4.5 algorithm [20], and RepTree is an implementation of J48 with reduced error pruning in Weka.

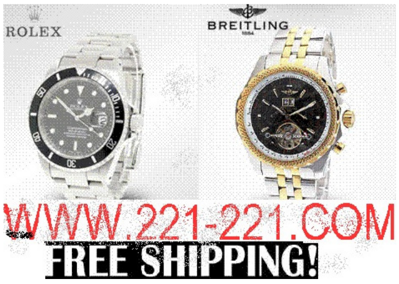
18.3 Spam Images Evolution and Datasets

18.3.1 Types and Trends of Image Spam

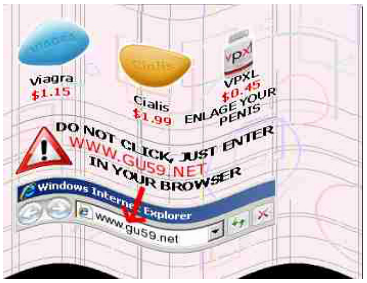
In general, spam refers to "unsolicited bulk email" [21, 22]. Though email spam is the most widely known form of spam it also appears in many other electronic media such as chats, internet telephony, social networks, and web spamming. We focus on email spam, although spam images are beginning to appear in these other contexts.

Spam filters typically look for certain key words like Viagra, cash, or money that are commonly found in spam. Spammers counteract with techniques to obfuscate spam filters. Some examples taken from the wild are described in [14] including common tricks such as adding random words before HTML, using white text on white background, using strange characters like V1@gr@, adding bogus HTML tags with a lot of random text, or adding spaces in words like "l o w I n t e r e s t R a t e".

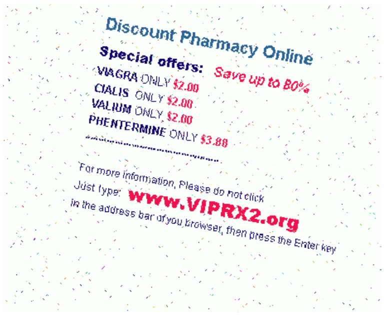
In image spam, spammers actually embed the spam message in a computer generated image instead of directly placing it as text content to evade spam filters. Optical character recognition (OCR) filters were quickly applied to extract words from the images and then determine if an image had spam content. This is an expensive process, but it worked in the beginning. Image spam techniques evolved as spammers devised new ways to evade OCR filters. Some of these include rotating the image, inducing a wave effect, adding noise, and slicing the image. Figure 2 shows several examples of this evolution.



(a) Adding noise to the image.



(b) Producing a wave effect.

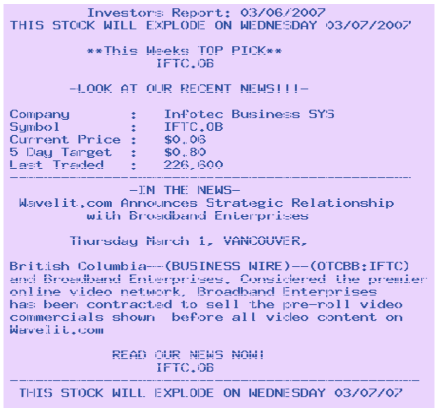

Discount Pharmacy Online
Special offers: Save up to 80%
VIAGRA ONLY $2.00
CIALIS ONLY $2.00
VALIUM ONLY $2.00
PHENTERMINE ONLY $3.88
For more information, Please do not click
Just type: www.VIPRX2.org
in the address bar of you browser, then press the Enter key

(c) Rotating the image and adding noise.

Fig. 2 Examples of the image spam evolution.

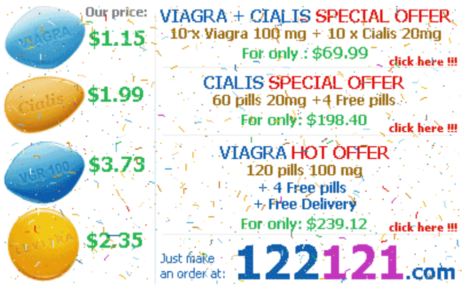
Text only images are easy enough for OCR filters to catch. However, randomization can be used to thwart signature based anti-spam algorithms. Random color stripes, random colored pixels, and shades of colors are ways to mask the content and confuse filters. Wild, colorful backgrounds and uneven text fonts also make it difficult for OCRs to extract text in the images. Figure 3 shows several examples of these types of images collected from our corpus. Trickier yet, some images are split into multiple parts, some containing the message and others containing an animation. The frames in the image rotate fast enough to display only the final result to the user.

Standard images are usually professional, neat images with none of the above tricks employed, giving them a genuine look. The entire message is contained in the image and hence scanners cannot detect it. In fact, many of the images that come in as spam today have a very professional look like the one in Figure 4.



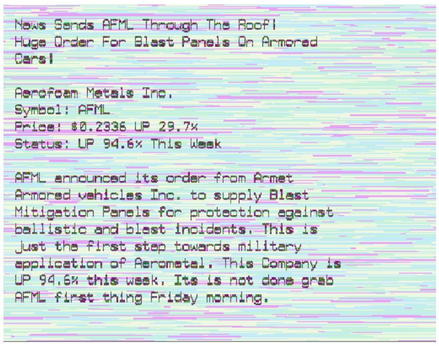
```
         Investors Report: 03/06/2007
THIS STOCK WILL EXPLODE ON WEDNESDAY 03/07/2007

         **This Weeks TOP PICK**
                IFTC.OB

       -LOOK AT OUR RECENT NEWS!!!-

Company        :   Infotec Business SYS
Symbol         :   IFTC.OB
Current Price  :   $0.06
5 Day Target   :   $0.80
Last Traded    :   226,600
-----------------------------------------------
             -IN THE NEWS-
 Wavelit.com Announces Strategic Relationship
         with Broadband Enterprises

       Thursday March 1, VANCOUVER,

British Columbia--(BUSINESS WIRE)--(OTCBB:IFTC)
and Broadband Enterprises, Considered the premier
online video network, Broadband Enterprises
has been contracted to sell the pre-roll video
commercials shown  before all video content on
Wavelit.com

            READ OUR NEWS NOW!
                IFTC.OB
-----------------------------------------------
 THIS STOCK WILL EXPLODE ON WEDNESDAY 03/07/07
```

(a) Text only image spam.



Our price: $1.15 VIAGRA + CIALIS SPECIAL OFFER
10 x Viagra 100 mg + 10 x Cialis 20mg
For only : $69.99 click here !!!
$1.99 CIALIS SPECIAL OFFER
60 pills 20mg +4 Free pills
For only: $198.40 click here !!!
$3.73 VIAGRA HOT OFFER
120 pills 100 mg
+ 4 Free pills
+ Free Delivery
For only: $239.12 click here !!!
$2.35 Just make an order at: 122121.com

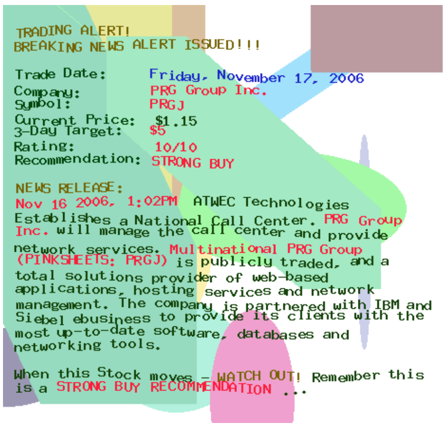
(b) Adding random colored pixels.



```
News Sends AFML Through The Roof!
Huge Order For Blast Panels On Armored
Cars!

Aerofoam Metals Inc.
Symbol: AFML
Price: $0.2336 UP 29.7%
Status: UP 94.6% This Week

AFML announced its order from Armet
Armored vehicles Inc. to supply Blast
Mitigation Panels for protection against
ballistic and blast incidents. This is
just the first step towards military
application of Aerometal. This Company is
UP 94.6% this week. Its is not done grab
AFML first thing Friday morning.
```

(c) Adding color streaks.

Fig. 3 More evolution of image spam.



TRADING ALERT!
BREAKING NEWS ALERT ISSUED!!!

Trade Date: Friday, November 17, 2006
Company: PRG Group Inc.
Symbol: PRGJ
Current Price: $1.15
3-Day Target: $5
Rating: 10/10
Recommendation: STRONG BUY

NEWS RELEASE:
Nov 16 2006, 1:02PM ATWEC Technologies
Establishes a National Call Center. PRG Group
Inc. will manage the call center and provide
network services. Multinational PRG Group
(PINKSHEETS: PRGJ) is publicly traded, and a
total solutions provider of web-based
applications, hosting services and network
management. The company is partnered with IBM and
Siebel ebusiness to provide its clients with the
most up-to-date software, databases and
networking tools.

When this Stock moves - WATCH OUT! Remember this
is a STRONG BUY RECOMMENDATION ...

(d) Adding a wild background.

Fig. 3 (*continued*)

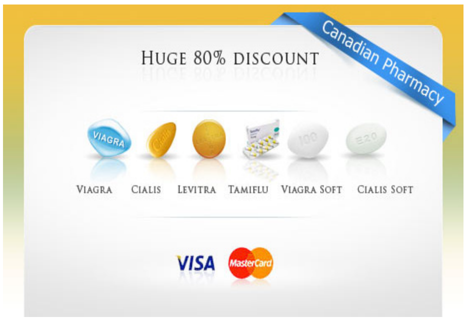


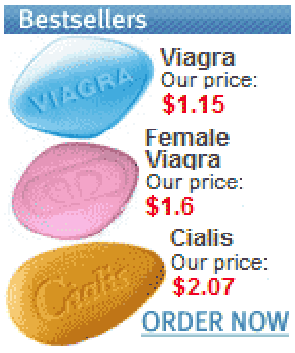
Fig. 4 A standard image.

18.3.2 The Corpus

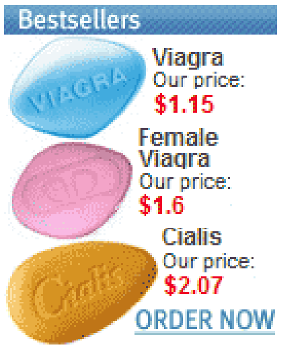
A corpus refers to a collection, in our case ham and spam images that we have used in this research. We have amassed a sizable set of spam images from the wild via a partnership with KnujOn ("no junk" spelled backwards), an anti-spam company [23]. Their mission is to fight against Internet threats, and specifically those delivered by email. They work with Internet governance bodies to help investigate abusive registrars and track cyber criminals. Part of the company's business is to allow users to upload their spam email and then process it, extracting hyperlinks and other information to help track the source of the message. In our case, images are automatically stripped out of the emails on a daily basis and forwarded to us in an effort to build a corpus for the purpose of image spam research. To date, we have over 215,000 unique images from the time span April 2008 through the February 2011.

For this research, we use 16,186 unique spam images collected from March 2009 through September 2010. These were verified to be unique by using MD5 (Message-Digest Algorithm 5), a cryptographic hash function with a 128-bit hash value [24]. It is very unlikely that two different files have same MD5 checksum, thus the MD5 hash acts like a digital fingerprint of a file. In case of spam images many spam images look similar but have different checksums. For example, Figure 5 shows two images that appear to be identical but have different checksums. In addition, 5440 ham images were collected from several different sources including personal photographs, Flickr [25], and some images from Wikipedia [26] and National Geographic Channel [27] which are covered under the Creative Commons license [28].

Spam images come in various formats. We have encountered .bmp, .gif, .jpeg and .png formats while collecting images for this research. The majority of images are JPEG. We convert the other images to the JPEG format for convenience in using the tools. It supports a 24-bit color map and has a small file size. JPEG is standard image format in many photography devices.



(a) MD5 = 996484e6cc7340ee2067ee96074ce324



(b) MD5 = 400fea9508fb5d759d9d698ef293c937

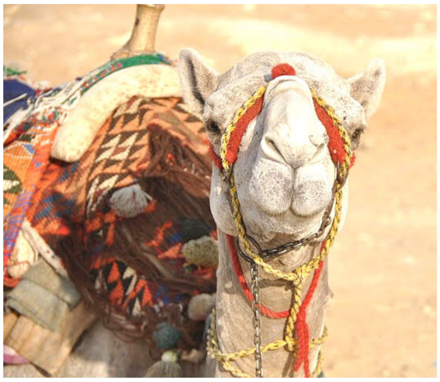
Fig. 5 Similar spam images but with different MD5 checksums.

18.4 Learning from Spam Images

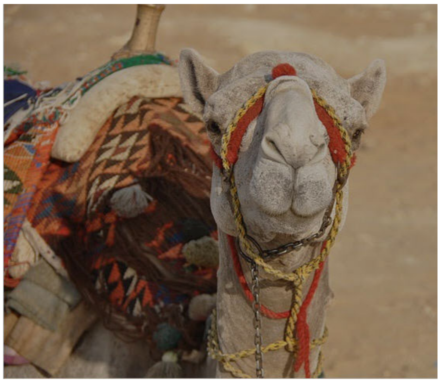
18.4.1 Spam Image Representation

When we look at an image we can easily (visually) identify whether it contains many different colors or is blurred but we cannot see other attributes clearly. In this research we have looked at some of these visual features to distinguish spam images from non-spam. The following features have been explored:

a) *Luminance*- This refers to the brightness of an image. Some images are brighter than other, as illustrated in Figure 6. From our observation, spam images are not very bright, as they are not camera quality shots.



(a) High luminance.



(b) Low luminance.

Fig. 6 Examples of different luminance.

We compare the luminance of different images using the *log-average luminance* of an image. This is calculated by finding the geometric mean of the luminance values of all pixels. In a gray scale image, the luminance value is the pixel value. In a color image, the luminance value is found by a weighted sum, described in [30]. We then take an average of the luminance values for all the pixels in an image.

$$\textbf{luminance} = \textbf{0.27 red} + \textbf{0.67 green} + \textbf{0.06 blue}$$

b) *Numbers of colors*- A JPEG image has a 24 bit color map, i.e., each pixel is 24 bits. This means that an image can potentially have 2^{24} or 16777216 different colors. We divide this range into 1677 bins with 1000 consecutive colors falling into one bin. We then identify the number of colors in the range for each bin. In this way, we reduce it to 1677 maximum colors per image.

c) *Color saturation*- This can be described as the pureness of a color. For example, this will determine "how red" is the color red in an image. If the pixel has a value of (R, G, B) = (255, 0, 0) the pixel has a high saturation of the color red. As defined in [11, 31] color saturation is the ratio of the total number of pixels in an image for which the difference:

$$\mathbf{max(R,G,B) - min(R,G,B)}$$

is greater than some threshold, T, to the number of pixels. We set the threshold value, T = 50, as recommended by Frankel et al. [31]. For every pixel in an image we calculate the maximum and minimum among the R, G, B values and then take the difference. We use a counter to count how many pixels have the difference greater than T (T=50). Finally we divide the counter by the number of pixels in the image to determine the saturation value.

d) *White pixel concentration*- Spam images generally have a solid background which is mostly pastel or white in color. For example, Figure 7 shows a ham image with subtle shades of different colors but no solid background. Compare this with the spam image in Figure 8 which has a more white background color. We calculate how many pixels in a given image have their (R, G, B) component values to be above 250 as any value above this range is a pastel shade like gray, pale white, etc. Next we take the ratio of number of white pixels to total number of pixels in an image. If an image has more white pixels than any other color then we label it to be a background color for the image.

e) *Standard deviation of colors*- This is the standard deviation of each (R, G, B) component, which tells us how much variation is there in each color component.

f) *Hue*- Hue can be described as the dominant wavelength in a color model which describes a given color. For example, if we are looking at a ripe Macintosh apple, the hue value is red, indicating what color the apple looks to us. Java's Color class provides a method to determine the hue value of a given pixel. We compute the hue values for each pixel and then compute mean of these values to represent the hue of the image.

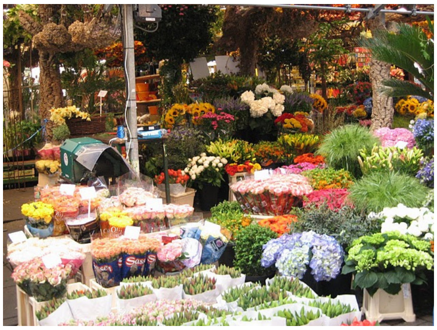


Fig. 7 Ham image.

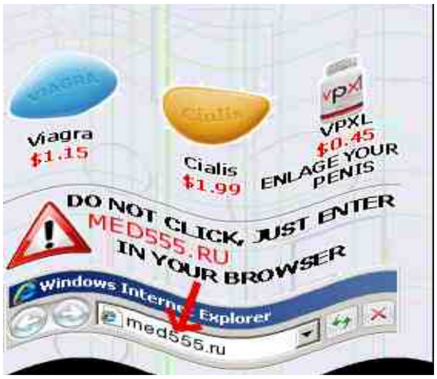


Fig. 8 Spam image.

There was some necessary cleaning performed on the images. We converted all images to JPEG format using ImageMagick [32], an open source utility that can converts images in various formats. We rescale exceedingly large ham images such that they did not exceed a size of 150KB. Rescaling does not affect the features we are working with. The image features were computed using Java's *imageIO* package to retrieve pixel values of an image. Then we extract the (R, G, B) values for each of these for further processing.

18.5 Experiments

Of many algorithms available for decision trees, C4.5 [20] is the most popular one. In our experiments, we use J48 and RepTree from the Weka data mining tools. J48 is an open source Java implementation of the C4.5 algorithm, and RepTree is another fast decision tree learner in Weka. It is an implementation of J48 with reduced-error pruning strategy.

18.5.1 Experiment with J48

In the first attempt we tried to use J48 to classify the data. We used a set of 16186 unique spam images and 5440 ham images. We selected 90% for training and 10% randomly for testing from these sets. Since most of the values in the feature set are floating point, the decision tree generated by J48 is very wide and has many leaf nodes. Discretization alone did not help much in our experiments. So, though the accuracy was about 98%, the tree seems to overfit the data, the result is biased, and it is harder to comprehend due to the large number of tree nodes.

18.5.2 Experiment with RepTree

We decided to see if any other classifier could be used which had a comparable accuracy to J48's results and generated a more comprehensible tree. RepTree is another classifier in Weka which is a fast decision tree learner. It builds a decision/regression tree using information gain/variance and prunes it using reduced-error pruning. It also lets you limit the depth of tree and sorts numeric attributes only once there by reducing number of branches of the tree. The accuracies were almost equal to using J48 with the advantage of a smaller tree.

However, the question is what value to set the depth parameter, and how to determine a reasonable value for the parameter. We picked our value by applying a simple hill-climbing strategy. We started with a depth of 1 and then kept increasing the depth with intervals of 1. We observed the accuracy at each interval change, and the point at which the accuracy stopped increasing and either became constant or started declining is selected for the depth of the tree. Table 1 below lists the depth value for each of different datasets with mixed ratios of spam and ham images we have tried in our experiments.

Table 1 Depth value parameter applied to the RepTree tool.

Ratio of Spam to Ham	No. of Spam	No. of Ham	Depth
1:1	5440	5440	5
3:1	16186	5440	7
1:9	604	5440	4
9:1	9000	1000	6

In real time there are no statistics on how many spam images are encountered for each ham image because ham images cannot be harvested in the same manner we harvest spam images. People do not donate their private correspondence containing photographs to the Knujon site. We either collected ham from our own photographs or used Flickr, or Google to get the images. The problem here was in what ratios do we choose ham images and spam images to form a realistic training set. We started with 16186 spam images and 5440 ham images which are approximately in a ratio of 3:1, i.e., three spam images to one ham image. We ran the experiments with RepTree and computed the classification accuracy.

We also tried different ratio mixes, 1:1, 1:9, and 9:1 and computed their accuracies. Table 2 lists the accuracies of each of these combinations and number of images in each set. From Table 2, we can see that we get fairly similar accuracies for different ratio mixes. We did not choose 1:9 and 9:1 as these are extreme cases. We chose 3:1 as it had better accuracy than 1:1 ratio.

Table 2 Accuracy of classification for different ratios of ham and spam images.

Ratio of Spam to Ham	No. of Spam	No. of Ham	Accuracy
1:1	5440	5440	97.94
3:1	16186	5440	98.28
1:9	604	5440	99.23
9:1	9000	1000	98.05

Based on the 3:1 ratio, a training set of 16186 spam images and 5440 ham images was constructed. The RepTree generated contains 63 leaf nodes. Figure 9 shows the tree generated by Weka. The important features from the tree are average luminance, number of colors and white pixel concentration.

Experiments were performed using the RepTree classifier with 10-fold cross-validation and independent testing file. The experiment was repeated 10 times, and we obtained an average classification accuracy of 98.28% in training.

To evaluate performance of the RepTree classifier on unseen images, we used a sample of 53300 unique images collected from the year 2010. We also downloaded 717 images for January 2011 to test how the classifier would work on recent spam images. The same tree was applied to 13 testing files created from samples collected over 13 months with sizes and types shown in Table 3.

Table 3 Sample distribution of different types of spam images in 2010.

Month	No. of Images	No. of JPEG	No. of GIF
Jan-10	2341	1945	396
Feb-10	2333	1689	644
Mar-10	12991	10626	2365
Apr-10	11471	9077	2394
May-10	8502	5801	2701
Jun-10	15734	14292	1442
Jul-10	5263	4620	643
Aug-10	33698	32631	1067
Sep-10	8777	8054	723
Oct-10	1713	1223	490
Nov-10	2303	2248	55
Dec-10	795	670	125
Jan-11	717	259	458
Total	106638	93135	13503
Unique Images	53300		

```
avgLuminance< 192.36
|   whiteConc< 0.11
|   |   stdDevRed< 84.51
|   |   |   avgLuminance< 142.59
|   |   |   |   avgSaturation< 0.81 : N (2824/60) [1411/42]
|   |   |   |   avgSaturation>= 0.81
|   |   |   |   |   avgLuminance< 93.95 : Y (25/11) [9/4]
|   |   |   |   |   avgLuminance>= 93.95 : N (126/9) [50/4]
|   |   |   avgLuminance>= 142.59
|   |   |   |   numColor< 260.5
|   |   |   |   |   avgSaturation< 0.16 : N (177/2) [78/2]
|   |   |   |   |   avgSaturation>= 0.16
|   |   |   |   |   |   numColor< 173.5 : Y (12/4) [3/0]
|   |   |   |   |   |   numColor>= 173.5 : N (223/23) [104/9]
|   |   |   |   numColor>= 260.5
|   |   |   |   |   avgHue< 0.44 : N (44/16) [22/10]
|   |   |   |   |   avgHue>= 0.44 : Y (15/0) [8/3]
|   |   stdDevRed>= 84.51
|   |   |   avgSaturation< 0.72
|   |   |   |   numColor< 263.5 : N (100/8) [82/7]
|   |   |   |   numColor>= 263.5
|   |   |   |   |   whiteConc< 0.01 : N (29/2) [15/2]
|   |   |   |   |   whiteConc>= 0.01 : Y (29/7) [18/8]
|   |   |   avgSaturation>= 0.72
|   |   |   |   numColor< 294.5
|   |   |   |   |   avgHue< 0.27 : N (3/1) [3/0]
|   |   |   |   |   avgHue>= 0.27 : Y (29/0) [15/0]
|   |   |   |   numColor>= 294.5 : N (3/1) [0/0]
|   whiteConc>= 0.11
|   |   numColor< 256.5
|   |   |   avgLuminance< 174.3
|   |   |   |   stdDevGreen< 107.79
|   |   |   |   |   numColor< 220.5 : Y (6/2) [2/0]
|   |   |   |   |   numColor>= 220.5 : N (145/12) [77/3]
|   |   |   |   stdDevGreen>= 107.79
|   |   |   |   |   avgSaturation< 0.01 : Y (5/0) [5/0]
|   |   |   |   |   avgSaturation>= 0.01
|   |   |   |   |   |   avgLuminance< 156.92 : N (4/1) [3/0]
|   |   |   |   |   |   avgLuminance>= 156.92 : Y (2/0) [1/0]
|   |   |   avgLuminance>= 174.3
|   |   |   |   stdDevRed< 75.59
|   |   |   |   |   stdDevGreen< 81.19 : N (12/2) [1/0]
|   |   |   |   |   stdDevGreen>= 81.19 : Y (2/0) [2/0]
|   |   |   |   stdDevRed>= 75.59 : Y (22/1) [5/0]
|   |   numColor>= 256.5
|   |   |   avgLuminance< 158.41
|   |   |   |   stdDevGreen< 96.66
|   |   |   |   |   whiteConc< 0.11 : Y (7/0) [2/0]
|   |   |   |   |   whiteConc>= 0.11 : N (25/8) [12/5]
|   |   |   |   stdDevGreen>= 96.66 : Y (32/0) [17/2]
|   |   |   avgLuminance>= 158.41 : Y (68/1) [33/2]
avgLuminance>= 192.36
|   numColor< 279.5
|   |   whiteConc< 0.04 : Y (46/19) [19/7]
|   |   whiteConc>= 0.04
|   |   |   avgHue< 0.14
|   |   |   |   avgLuminance< 224.89
|   |   |   |   |   avgSaturation< 0.08 : Y (10/2) [11/2]
|   |   |   |   |   avgSaturation>= 0.08 : N (4/0) [6/2]
|   |   |   |   avgLuminance>= 224.89 : Y (100/2) [70/3]
|   |   |   avgHue>= 0.14 : Y (652/4) [324/4]
|   numColor>= 279.5 : Y (9636/0) [4801/0]
```

Fig. 9 RepTree classifier generated by Weka

The test files were generated for each month with equal number of spam and ham images. We had 1688 ham images which were not used for training. So, for each month, we created a test file consisting of 1688 ham images and 1688 spam images to maintain a 1:1 ratio. For the month December 2010, we had lesser than 1688 spam images in the corpus, in such case we picked equal number of ham images randomly from 1688 images. Similarly, if number of spam samples were more than 1688 we chose 1688 images randomly from them. Table 4 lists the performance of classification for each of the months. The average accuracy of the classifier for unseen samples is 89%. The False Positive Rate (FPR) is defined as FP / (FP + TN), and the False Negative Rate is defined as FN / (FN + TP) where False Positive (FP) is the number of testing negative instances that are wrongly classified as positive by the classifier, True Negative (TN) is the number of negative instances that are classified as negative, False Negative (FN) is the number of positive instances classified as negative, and True Positive (TP) is the number of positive instances classified as positive. The Recall is defined as TP / (TP + FN), the Precision is defined as TP / (TP + FP), and the F-Measure is defined as 2*TP / (2*TP+FN+FP). In Table 4, the average FPR is 5.22%, and the average FNR is around 18.51%. The recall rate fluctuated with the accuracy; however, the average precision is staying above 92%. The average Recall, also called True Positive Rate (TPR), is 81.5%.

Table 4 Performance of classification for unseen samples.

	Accuracy	Recall	Precision	F-Measure	FPR	FNR
10-Jan	0.9252	0.9055	0.9427	0.9237	0.0551	0.0945
10-Feb	0.9057	0.8618	0.9448	0.9014	0.0503	0.1382
10-Mar	0.9390	0.9313	0.9458	0.9385	0.0533	0.0687
10-Apr	0.8990	0.8513	0.9411	0.8939	0.0533	0.1487
10-May	0.8320	0.7174	0.9308	0.8103	0.0533	0.2826
10-Jun	0.7933	0.6439	0.9184	0.7570	0.0572	0.3561
10-Jul	0.8451	0.7469	0.9295	0.8282	0.0567	0.2531
10-Aug	0.9648	0.9828	0.9485	0.9654	0.0533	0.0172
10-Sep	0.9428	0.9390	0.9463	0.9426	0.0533	0.0610
10-Oct	0.8182	0.6913	0.9263	0.7918	0.0550	0.3087
10-Nov	0.9476	0.9359	0.9582	0.9469	0.0408	0.0641
10-Dec	0.8241	0.6912	0.9415	0.7972	0.0429	0.3088
11-Jan	0.8210	0.6960	0.9281	0.7955	0.0540	0.3040

Table 5 shows the computing time for extraction features used in this study in seconds based on 21,626 images consisting of 16,186 spam images with average size of 21.32 Kbytes and 5440 ham with average size of 131.39 Kbytes. The time shown for computing the standard deviation feature is a combined time for all three colors. The table shows that it takes about 0.486 seconds to extract features for each image on the average.

Note that J48 and RepTree use built-in filter-based feature selection algorithms [19]. There are other approaches based on wrappers [33, 36], rough sets [34, 35] or fuzzy sets [37]. One of our future works is to study how different approaches to feature reduction can further lead to improvement of computing time for feature extraction.

Table 5 Computing time for feature extraction.

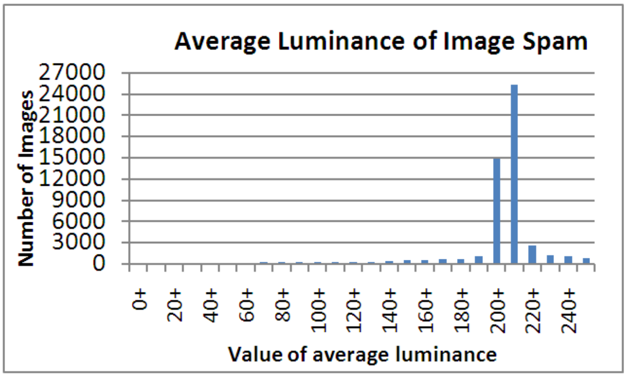
Features	Spam(sec)	ham(sec)	ham + Spam(sec)	Avg.(ham + spam)
Luminance	938.163	509.577	1447.740	0.067
Saturation	1146.220	496.305	1642.525	0.076
Hue	1416.485	573.444	1989.929	0.092
Number of colors	1555.486	615.987	2171.473	0.100
White Pixel Concentration	1022.250	461.654	1483.904	0.069
Standard Deviation	1283.551	483.472	1767.023	0.082
Total time for all features	7362.155	3140.439	10502.594	0.486

18.6 Validation by Feature Analysis

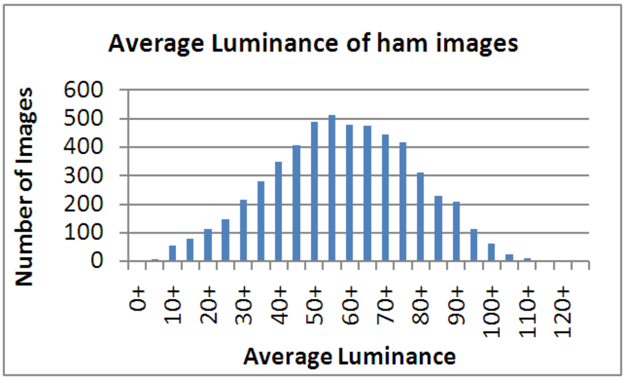
We had 53300 unique images in the test set. After feature extraction we look for a pattern in the values of the features for image spam. We use eight features, each with values that lie in a range. For example, the average value range for luminance is between 0–255, hue is between 0–1, and the number of colors is between 0– 1677. We divide these ranges into equal intervals and count the number of images with feature values in that range. Figure 10 shows graphs with the distribution of spam and ham images in these ranges.

The graphs show that most of the feature values for spam images lie in some specific narrower ranges. For example, most spam images have an average luminance value in the range of 200–220. However, for ham images these values are spread over a wider range and none of the values fall in the range of 200–220. Hence, this is a good way to determine if an image could be a spam. The decision tree also chooses luminance as the root of the tree and has a cutoff value of 192.36 for average luminance. Next, if we look at average saturation of color values we see that ham images are spread over a wider range than spam images which have saturation values mostly in the range of 0–0.2. Ham images have saturation values mostly between 0–0.8.

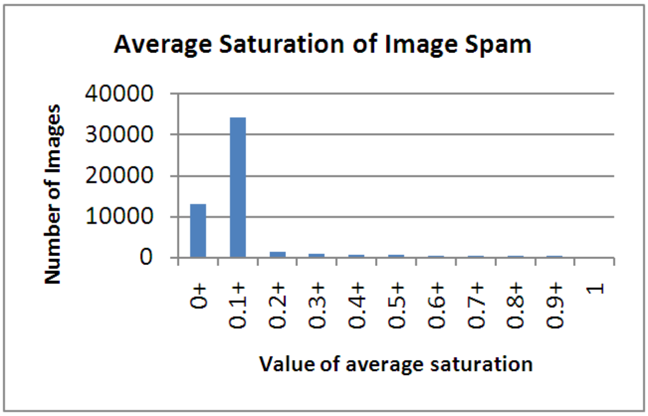
Similarly, the average hue for ham images is spread out into different ranges when compared to spam images. The number of colors and white/pastel pixel concentration are not very helpful measures to identify spam from ham as both of these spam and ham images have values in similar ranges. Standard deviation of color components is more spread out, again in the case of ham images rather than spam images. These are expected values as photographs have different shades of colors than a spam image. Pictures are taken at different times of the day, so the luminance values are spread across different ranges. Also, the luminance value of ham images is less than spam images because generally spam images are brighter so that they attract the user's attention. The text may be skewed and there may be random noise in the image but the image is bright in appearance so that a human user can read it easily but an OCR can't.



(a) Average luminance of spam.

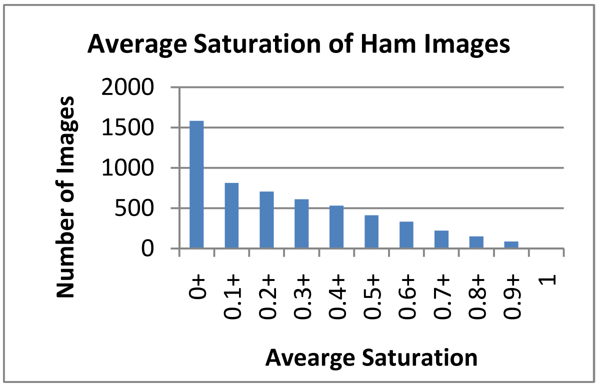


(b) Average luminance of ham.

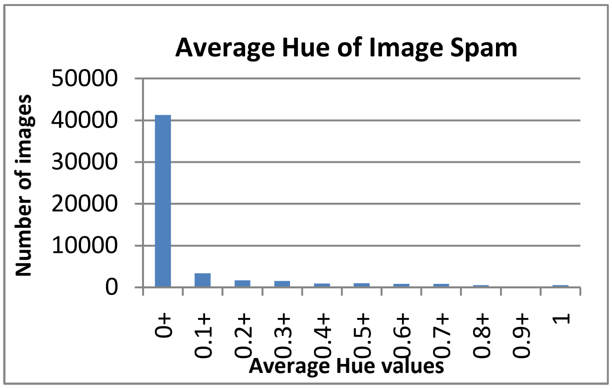


(c) Average color saturation of spam.

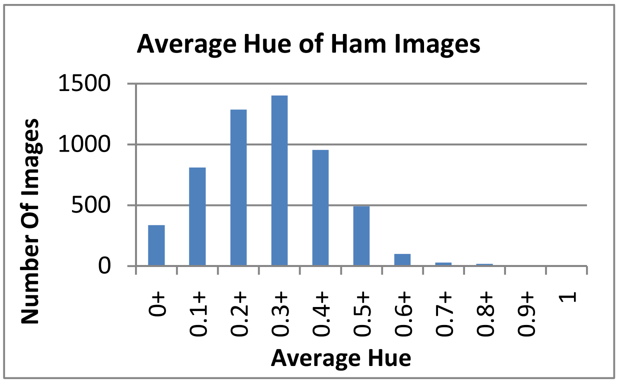
Fig. 10 Feature set.



(d) Average color saturation of ham.

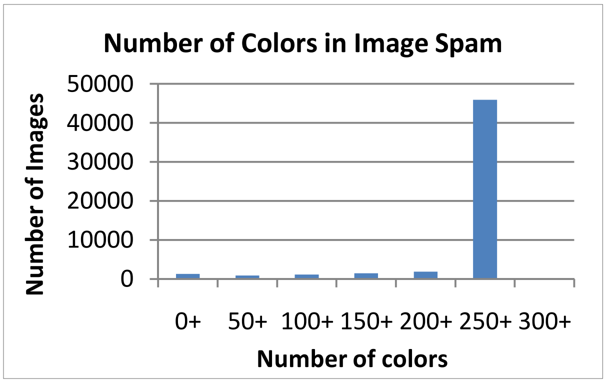


(e) Average hue of spam.

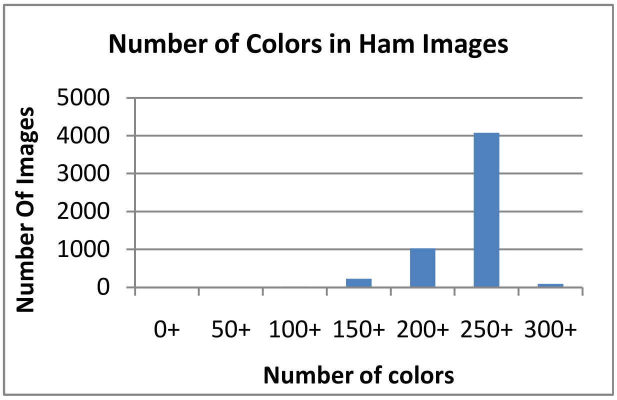


(f) Average hue of ham.

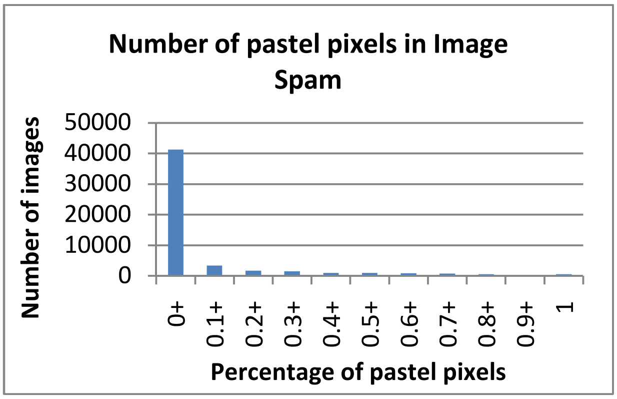
Fig. 10 (*continued*)



(g) Average number of colors in spam.

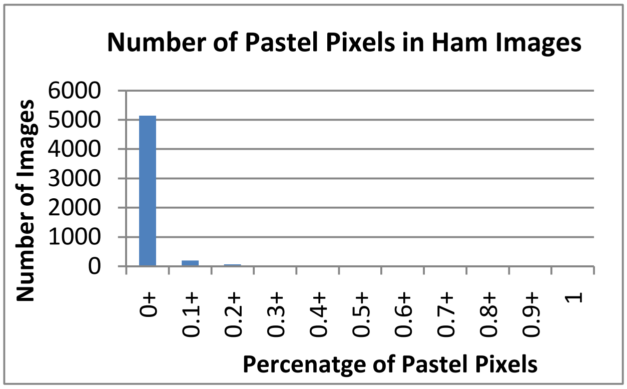


(h) Average number of colors in ham.

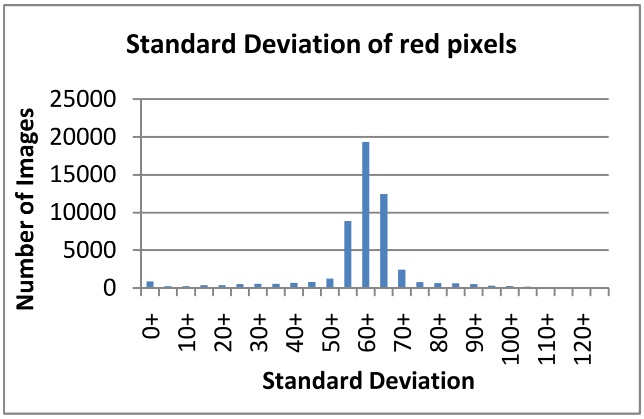


(i) Average number of pastel pixels in spam.

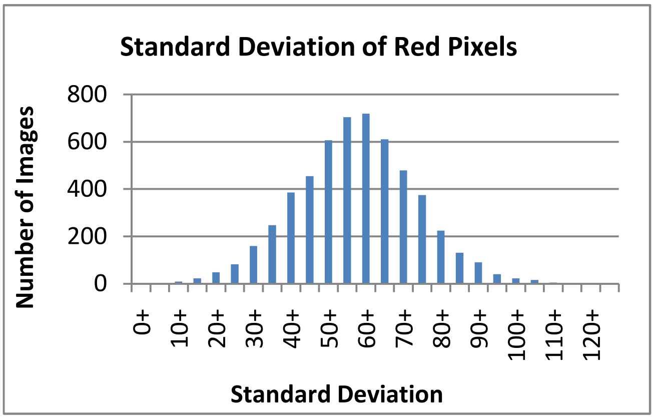
Fig. 10 (*continued*)



(j) Average number of pastel pixels in ham.

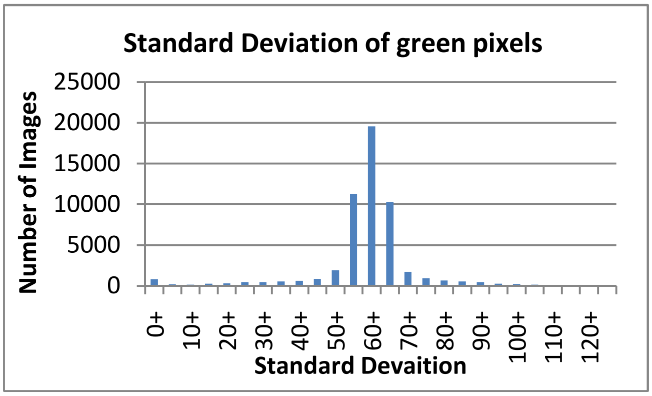


(k) Standard deviation of red pixels in spam.

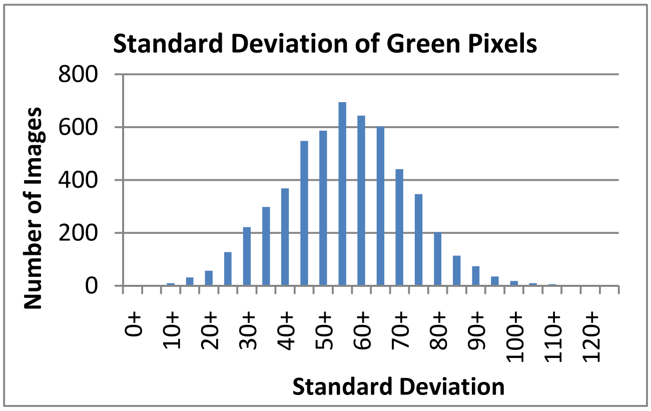


(l) Standard deviation of red pixels in ham.

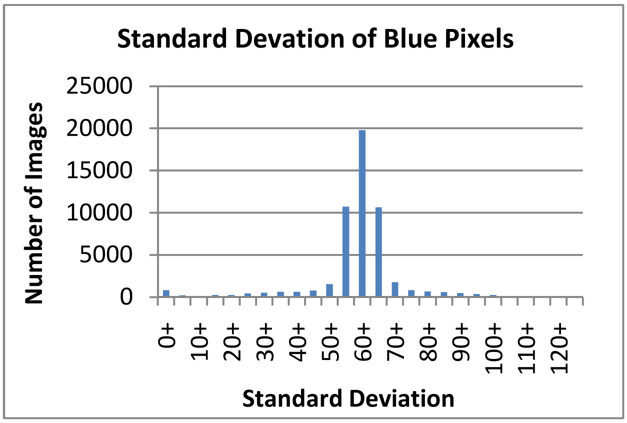
Fig. 10 (*continued*)



(m) Standard deviation of green pixels in spam.

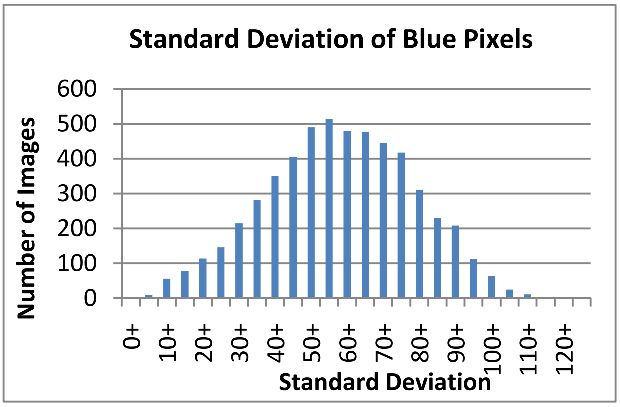


(n) Standard deviation of green pixels in ham.



(o) Standard deviation of blue pixels in spam.

Fig. 10 (*continued*)



(p) Standard deviation of blue pixels in ham.

Fig. 10 (*continued*)

Since the values of features are in clearly distinct ranges it is easy for the classifier to classify spam from ham images. The graphs also suggest that, even though spam images might contain rotated content, random noise, random pixels, or skewed text to obfuscate the filters, there are some features that do not change in spite of these tricks. For example, the luminance value is similar for many different spam images as shown in the graphs.

18.7 Conclusions

Our experiment has provided us with insightful observations about how spam images have evolved in a year. Many spam images are almost photo quality images and have multiple colors. This makes the classification process trickier as it gets harder to distinguish these images from photographs. Newer techniques are used in generating image spam like scraping off header information, images that do not load when viewed as thumbnails but will open with a picture editor, malware injection and so forth. We also observed that images seem to follow trends in a particular month, for example a dominant trend in a month could be the advertisement of vines, exercise equipment, pharmaceuticals, chocolates etc., or it could be chain letters, dating websites, money making schemes and many others.

The pros of using the described approach for image spam classification are that the features are simple and easily computable, and the generated decision tree classifiers achieve a good accuracy for unseen samples from a recent time period. Irrespective of image file size, resizing or converting image formats does not affect values of the proposed features. In addition, the proposed features may complement or enhance image spam filters based on low-level features, which used alone may not be sufficient as current spam images have comparable quality to photo images. However computing features from image is not computationally inexpensive. One way to minimize the computational cost is to use the Just In Time (JIT) feature extraction approach proposed in [16].

Finally, if the classifier is updated frequently with new incoming spam images it might provide better classification performance over a period of time. In Table 4, we can see that for November 2010 samples we have an accuracy of almost 94%. This indicates that spam images recur with modifications like change in few pixels, rotation, or noise but the properties like luminance, saturation, hue are not modified frequently. Also, in each month we get an accuracy of at least 79%. Some months had very few spam samples and this could be a reason we see a slight dip in the accuracy. A challenge in the experiment was the ham images, since they are not available so easily. We can download images using Flickr API; however, processing these is a time consuming process, and hence our ham corpus is more limited than our spam corpus.

References

1. Goodman, J., Cormack, G.V., Heckerman, D.: Spam and the ongoing battle for the inbox. Communication of the ACM 50(2), 25–33 (2007)
2. Siponen, M., Stucke, C.: Effective anti-spam strategies in companies: an international study. In: Proceedings of HICSS 2006 (2006)
3. Hershkop, S.: Behavior-based email analysis with application to spam detection. Doctoral Dissertation, Columbia University New York, NY, USA (2006) ISBN: 0-542-46250-8
4. Yeh, C.-Y., Wu, C.-H., Doong, S.-H.: Effective spam classification based on meta-heuristics. In: Proceedings of IEEE International Conference on Systems, Man and Cybernetics, SMC 2005, vol. 4, pp. 3872–3877 (2005)
5. Zhang, L., Zhu, J., Yao, T.: An evaluation of statistical spam filtering techniques. ACM Trans. Asian Lang. Inform. Process (TALIP) 3(4), 243–269 (2004)
6. Lai, C.-C., Tsai, M.-C.: An empirical performance comparison of machine learning methods for spam e-mail categorization. Hybrid. Intell. Syst., 44–48 (2004)
7. Fawcett, T.: "In vivo" spam filtering: a challenge problem for data mining. In: KDD Explor., vol. 5(2), pp. 140–148 (2003)
8. Blanzieri, E., Bryl, A.: A survey of learning-based techniques of email spam filtering. Artificial Intelligence Review 29(1), 63–92 (2009)
9. Guzella, T.S., Caminhas, W.M.: A review of machine learning approaches to Spam filtering. Expert Systems with Applications 36(7), 10206–10222 (2009)
10. Hope, P., Bowling, J.R., Liszka, K.J.: Artificial Neural Networks as a Tool for Identifying Image Spam. In: The 2009 International Conference on Security and Management (SAM 2009), pp. 447–451 (July 2009)
11. Aradhye, H.B., Myers, G.K., Herson, J.A.: Image analysis for efficient categorization of image-based spam e-mail. In: Proceedings of the Eighth International Conference on Document Analysis and Recognition, August 29, September 1, vol. 2, pp. 914–918 (2005)
12. Wang, C., Zhang, F., Li, F., Liu, Q.: Image spam classification based on low-level image features. In: 2010 International Conference on Communications, Circuits and Systems (ICCCAS), July 28-30, pp. 290–293 (2010)

13. Krasser, S., Tang, Y., Gould, J., Alperovitch, D., Judge, P.: Identifying Image Spam based on Header and File Properties using C4.5 Decision Trees and Support Vector Machine Learning. In: Proc. of the IEEE SMC Information Assurance and Security Workshop, IAW 2007, June 20-22, pp. 255–261 (2007)
14. He, P., Wen, X., Zheng, W.: Simple Method for Filtering Image Spam. In: Eighth IEEE/ACIS International Conference on Computer and Information Science, ICIS 2009, June 1-3, pp. 910–913 (2009)
15. Egan, J.P.: Signal detection theory and ROC analysis. Series in Cognition and Perception. Academic Press, New York (1975)
16. Dredze, M., Gevaryahu, R., Elias-Bachrach, A.: Learning Fast Classifiers for Image Spam. In: Fourth Conference on Email and Anti-Spam, CEAS (2007)
17. Gao, Y., Yang, M., Zhao, X., Pardo, B., Wu, Y., Pappas, T.N., Choudhary, A.: Image spam hunter. In: IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2008, March 31 -April 4, pp. 1765–1768 (2008)
18. Jardine, N., Sibson, R.: Mathematical Taxonomy. Wiley, New York (1971)
19. Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., Witten, I.H.: The WEKA Data Mining Software: An Update; SIGKDD Explorations 11(1) (2009)
20. Quinlan, J.R.: C4.5: Programs for Machine Learning. Morgan Kaufmann Publishers (1993)
21. Androutsopoulos, I., Koutsias, J., Chandrinos, K.V., Spyropoulos, C.D.: An experimental comparison of naive bayesian and keyword-based anti-spam filtering with personal e-mail messages. In: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2000, pp. 160–167. ACM Press, New York (2000) ISBN 1-58113-226-3
22. SPAMHAUS, The definition of spam (2005), `http://www.spamhaus.org/definition.html`
23. Knujon, `http://www.knujon.com/`
24. Rivest, R.: The MD5 Message-Digest Algorithm. RFC Editor (1992)
25. Flickr, `http://www.flickr.com/`
26. Wikipedia, `http://en.wikipedia.org`
27. National Geographic Channel, `http://photography.nationalgeographic.com/photography/?source=NavPhoHome`
28. Creative Commons license, `http://creativecommons.org/`
29. Types of Bitmaps, `http://msdn.microsoft.com/en-us/library/ms536393%28VS.85%29.aspx`
30. Luminance of Images, `http://www.cacs.louisiana.edu/~cice/lal/index.html`
31. Frankel, C., Swain, M.J., Athitsos, V.: WebSeer: An Image Search Engine for the World Wide Web. Computer Science Department. The University of Chicago, Chicago (1996)
32. Image Magick, `http://www.imagemagick.org/script/index.php`
33. Blum, A., Langley, P.: Selection of relevant features and examples in machine learning. Artificial Intelligence 97(1-2), 245–271 (1997)

34. Pawlak, Z.: Rough Sets. International Journal of Computer and Information Sciences 11, 341–356 (1982)
35. Kohavi, R.: Useful Feature Subsets and Rough Set Reducts. In: Proc. Third International Workshop on Rough Set and Soft Computing, pp. 310–317 (1994)
36. Kohavi, R., John, G.H.: Wrappers for feature subset selection. Artificial Intelligence 97(1-2), 273–324 (1997)
37. Pal, S.K., Chakraborty, B.: Fuzzy set theoretic measure for automatic feature evaluation. IEEE Trans. Syst., Man, Cybern. SMC-16, 754–760 (1986)

Author Index