Studies in Computational Intelligence 80

Joachim Diederich (Ed.)

Rule Extraction from Support Vector Machines

Springer

Joachim Diederich (Ed.)

Rule Extraction from Support Vector Machines

Studies in Computational Intelligence, Volume 80

Editor-in-chief
Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

Joachim Diederich
(Ed.)

Rule Extraction from Support Vector Machines

With 55 Figures and 51 Tables

Springer

Joachim Diederich
Honorary Professor
School of Information Technology and
Electrical Engineering
School of Medicine, Central Clinical Division
The University of Queensland
Brisbane Q 4072
Australia
joachimd@itee.uq.edu.au

ISBN 978-3-540-75389-6 e-ISBN 978-3-540-75390-2

Studies in Computational Intelligence ISSN 1860-949X

Library of Congress Control Number: 2007937227

Cover Design: WMXDesign GmbH, Heidelberg

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

spinger.com

Preface

Over a period spanning more than a decade, support vector machines (SVMs) have evolved into a leading machine learning technique. SVMs are being applied to a wide range of problems, including bioinformatics, face recognition, text classification and many more. It is fair to say that SVMs are one of the most important methods used for data mining with a wide range of software available to support their application.

A significant barrier to the widespread application of support vector machines is the absence of a capability to explain, in a human comprehensible form, either the process by which an SVM arrives at a specific decision/result, or more general, the totality of knowledge embedded in these systems. This lack of a capacity to provide an explanation is an obstacle to a more general acceptance of "back box" machine learning systems. In safety-critical or medical applications, an explanation capability is an absolute requirement.

This book provides an introduction and overview of methods used for rule extraction from support vector machines. The first part offers an introduction to the topic as well as a summary of current research issues. The second chapter surveys the field of rule extraction from SVMs, reviews areas of current research and introduces an application in the financial field.

Part II describes a range of methods currently being used to extract comprehensible rules from support vector machines. It is very fortunate that practically all authors who published break-through papers on the topic in journals and conference proceedings since 2002 are contributing to this book. One of the first papers with the title "Rule extraction from support vector machines" (if not the first paper) was published in 2002 by Núñez et al. and describes a "decompositional" method to obtain rules from SVMs. Haydemar Núñez, with coauthors, is contributing to the second part of this book. Another decompositional method was published soon afterwards by Glenn Fung who is also contributing a chapter, as is Lisa Torrey and colleagues from the University of Wisconsin in Madison. The research group led by Jude Shavlik has made significant and highly regarded contributions to both rule extraction from neural networks as well as support vector machines.

Publications that truly shaped the field and the authors are establishing the connection between SVMs and reinforcement learning in Part II of this book.

Clearly, the field of rule extraction from support vector machines has grown and is now including methods such as SVM Trees which are being introduced by Shaoning Pang and Nik Kasabov in Part II of this volume. The use of prototypes for explanatory purposes has a history in the area of rule extraction from neural networks. Prototypes are now being applied to SVMs by Marcin Blachnik and Wlodzislaw Duch in this book.

Rule extraction from neural networks is an established data mining method. This was made possible by authors who applied these emerging methods to real world data sets and compared the results with other machine learning techniques. Fortunately, researchers in the field of rule extraction from support vector machines have tackled real word problems early on and one of the first attempts by Rolf Mitsdorffer is included in Part III of this book. Part III is entirely devoted to real world applications of rule extraction from support vector machines. Mitsdorffer investigates the problem of forecasting the success of initial public offering in the US stock market by use of rule extraction and other methods.

Support vector machines are performing particularly well for high dimensional data, i.e. sample sets with input vectors that have a significant number of elements (tens of thousands and more) and relatively few nonzero values. Problems such as text, speech and image classification often include data sets of this nature. An application to speech recognition, and in particular accent classification, by Carol Pedersen has been included in the third part of this book. This chapter is also investigating novel testing and evaluation methods for rule extraction from SVMs.

As mentioned earlier, bioinformatics and in particular protein structure prediction is a core application area for support vector machines. Jieyue He and coworkers have developed rule extraction methods for these applications and the work is included in this book.

I would like to thank Professor Janusz Kacprzyk for including this volume in Springer Verlag's "Studies on Computational Intelligence" series and Thomas Dillinger and Heather King for their ongoing advice and support during the editing process of this book. This book would not have been possible without the encouragement of Wlodzislaw Duch during a visit to Singapore in 2006. Jude Shavlik offered important advice on the structure of this book. My friends and colleagues Alan Tickle and Shlomo Geva have contributed many ideas on rule extraction over the years and I am grateful for their comments.

Special thanks to Professor Paul Bailes from the School of Information Technology and Electrical Engineering at the University of Queensland for his ongoing support and the use of the excellent facilities of the University. Finally, I would like to thank the most prolific academic book author I know, my wife Susan K. Wright, for her encouragement and support.

Brisbane, Australia *Joachim Diederich*
July 2007

Contents

Part I Introduction

Rule Extraction from Support Vector Machines: An Introduction
Joachim Diederich 3
1 Explanation: The Foundations 3
1.1 Forms of Explanation 4
1.2 Analogy as a Form of Explanation 5
1.3 Explanation-Based Generalization 6
1.4 How and Why Explanations 7
1.5 Generating or Identifying the Best Explanation 8
2 Rule Extraction from Support Vector Machines: Aims and Significance 8
2.1 Provision of a "User Explanation" Capability 9
2.2 Transparency 10
2.3 Software Verification 10
2.4 Improving Generalisation 11
2.5 Data Exploration and the Induction of Scientific Theories 11
3 Translucency and Rule Quality 11
3.1 The Neural Network Case 12
3.2 Translucency and Rule Quality Applied to Rule Extraction from SVMs 13
4 An Alternative View on Rule Extraction: Information Retrieval 14
5 A Case Study 16
6 A Classification System for Rule Extraction from SVMs 26
7 Conclusions and Future Challenges 28
8 Acknowledgements 30
References 30

Rule Extraction from Support Vector Machines: An Overview of Issues and Application in Credit Scoring
David Martens, Johan Huysmans, Rudy Setiono, Jan Vanthienen, and Bart Baesens 33
1 Introduction 34
2 The Support Vector Machine 34
3 The Rationale Behind SVM Rule Extraction 36
3.1 Why Rule Extraction 36
3.2 Why SVM Rule Extraction 37
4 An Overview of SVM Rule Extraction Techniques 38
4.1 Classification Scheme for SVM Rule Extraction Techniques 38
4.2 SVM Rule Extraction Techniques 41
5 Issues Concerning SVM Rule Extraction 49
5.1 Rule Output 49
5.2 High Dimensional Data 51
5.3 Constraint Based Learning: Knowledge Fusion Problem 51
5.4 Specificness of Underlying Black Box Model 53
5.5 Regression 53
5.6 Availability of Code 53
6 Credit Scoring Application 53
6.1 Credit Scoring in Basel II 53
6.2 Classification Model 54
7 Alternatives to Rule Extraction 56
7.1 Inverse Classification 56
7.2 Self Organizing Maps 56
7.3 Incremental Approach 57
8 Conclusion 59
9 Acknowledgement 59
References 59

Part II Algorithms and Techniques

Rule Extraction for Transfer Learning
Lisa Torrey, Jude Shavlik, Trevor Walker, and Richard Maclin 67
1 Introduction 67
2 Transfer Learning and Advice Taking 67
3 SVMs in Reinforcement Learning 69
4 Extracting Rules from an RL Source Task 71
4.1 Acquiring Rules from the Q-function 72
4.2 Acquiring Rules from Observed Behavior 73
5 Case Study 76
5.1 Policy-Transfer Results 78
5.2 Skill-Transfer Results 79

6 Summary and Open Problems 81
References 81

Rule Extraction from Linear Support Vector Machines via Mathematical Programming
Glenn Fung, Sathyakama Sandilya, and R. Bharat Rao 83
1 Introduction 83
1.1 About Notation 84
2 Medical Relevance 85
3 Sparse Hyperplane Classifiers: 1-Norm Support Vector Machines 86
4 Rule Extraction from Hyperplane Classifiers 87
4.1 Volume Maximization Criteria 90
4.2 Point Coverage Maximization Criteria 91
5 Algorithm Convergence Properties 93
6 Numerical Testing 96
6.1 WDBC Dataset 98
6.2 The Lung CAD Dataset 99
7 Other Mathematical Programming Formulations 99
7.1 Conditioning Rules by Using Prior Knowledge 99
7.2 Creating a Rule that Covers an Specific Given Point or Set of Points 100
7.3 Rule Extraction and Knowledge-Based SVMS for Incremental Learning 101
8 Conclusion and Future Directions 104
References 105

Rule Extraction Based on Support and Prototype Vectors
Haydemar Núñez, Cecilio Angulo, and Andreu Català 109
1 Combining Support Vectors and Prototype Vectors to Extract Rules 110
1.1 Building an Ellipsoid and Its Associated Rule Equation 112
1.2 Generating a Set of Rules 118
1.3 Simplified Representational Language for the Model 122
1.4 Classification by Using the Set of Rules 126
2 Experiments 127
3 Eliminating Randomness from the Clustering Algorithm 130
4 Conclusions and Further Research 132
References 133

SVMT-Rule: Association Rule Mining Over SVM Classification Trees
Shaoning Pang and Nik Kasabov 135
1 Introduction 135
2 SVM Classification Tree 137
2.1 Two-Class SVM Tree 137

3 The Spanning of SVM Tree . . . 141
3.1 Depth-First Spanning Tree . . . 142
3.2 Breadth-First Spanning Tree . . . 142
3.3 The SVMT Algorithms . . . 143
3.4 Coping with Class Imbalance and Class Overlap . . . 145
4 SVMT Rules Extraction . . . 145
4.1 Logic Association Rules . . . 145
4.2 SVM Nodes Interpolation . . . 146
4.3 SVMT-Rule . . . 149
5 Experiments and Applications . . . 152
5.1 Synthetic Dataset . . . 152
5.2 Cancer Diagnosis . . . 156
5.3 Fraud Detection . . . 158
6 Discussions and Conclusions . . . 159
7 Acknowledgements . . . 160
References . . . 160

Prototype Rules from SVM
Marcin Blachnik and Włodzisław Duch . . . 163
1 Why Prototype-Based Rules? . . . 163
2 P-Rules and Their Interpretation . . . 165
2.1 Types of P-Rules . . . 166
2.2 Support Vectors as Prototypes . . . 166
2.3 Removing Linear Dependencies Among Support Vectors . . . 167
2.4 Reducing the Number of Support Vectors . . . 169
2.5 Finding Optimal Number of Support Vectors . . . 171
2.6 Problems with Interpretation . . . 174
3 Searching for Informative Prototypes . . . 174
3.1 Prototype Selection Using Context Dependent Clustering . . . 176
3.2 The Conditional Fuzzy Clustering Algorithm . . . 177
3.3 Determining the Context . . . 178
3.4 Numerical Illustration of the CFCM Approach . . . 178
4 Conclusions . . . 180
References . . . 181

Part III Applications

Prediction of First-Day Returns of Initial Public Offering in the US Stock Market Using Rule Extraction from Support Vector Machines
Rolf Mitsdorffer and Joachim Diederich . . . 185
1 Motivation . . . 185
2 Introduction . . . 186
2.1 Financial Data Mining . . . 186

2.2 IPOs as a Case Study 186
3 Overview of the Chapter 187
4 Methodology 187
4.1 Statistical Tests 188
4.2 Data 188
4.3 Machine Learning Techniques Used in This Study 191
5 Results 194
5.1 Results of Rule Extraction from SVM for Cross-Industry IPOs 194
5.2 Rule Extraction from SVM Results for Single-Industry IPOs 196
6 Discussion of Results 197
7 Conclusions 201
References 202

Accent in Speech Samples: Support Vector Machines for Classification and Rule Extraction
Carol Pedersen and Joachim Diederich 205
1 Introduction 205
1.1 Motivation and Significance 205
1.2 Overview 205
2 Accent Recognition 206
2.1 Accent 206
2.2 Automatic Speech Recognition 207
2.3 Mel Frequency Cepstrum Coefficients 208
3 Rule Extraction from Support Vector Machines for Accent 209
3.1 Support Vector Machines 209
3.2 Rule Extraction 210
3.3 Objectives 211
4 Methodology 211
4.1 Speech Data and Feature Extraction 211
4.2 Machine Learning Experiments 212
4.3 Rule Extraction and Evaluation 213
5 Results 213
5.1 Machine Learning Experiments 213
5.2 Evaluation of the Rule Extraction Results 218
6 Discussion 223
References 225

Rule Extraction from SVM for Protein Structure Prediction
Jieyue He, Hae-jin Hu, Bernard Chen, Phang C. Tai, Rob Harrison, and Yi Pan 227
1 Introduction 227
2 Rule Generation by Combing SVM and DT 229
2.1 SVM_DT 229

2.2 Protein Second Structure Prediction with SVM_DT 231
2.3 Transmembrane Segments Prediction and Understanding Using SVM_DT 235
3 Extracting Rule from SVM Based on Association Rule 238
3.1 Association Rule Based Method 238
3.2 Association Rule Mining 239
4 Rule Clustering and Super_rule Generation 244
5 Conclusions 247
6 Acknowledgements 249
References 249

Subject Index 253

Author Index 261

Part I

Introduction

Rule Extraction from Support Vector Machines: An Introduction

Joachim Diederich

American University of Sharjah, UAE and University of Queensland, Australia

Rule extraction from support vector machines (SVMs) follows in the footsteps of the earlier effort to obtain human-comprehensible rules from artificial neural networks (ANNs) in order to explain "how" a decision was made or "why" a certain result was achieved. Hence, much of the motivation for the field of rule extraction from support vector machines carries over from the now established area of rule extraction from neural networks. This introduction aims at outlining the significance of extracting rules from SVMs and it will investigate in detail what it means to *explain the decision-making process of a machine learning system* to a *human user* who may *not be an expert* on artificial intelligence or the particular application domain. It is natural to refer to both psychology and philosophy in this context because "explanation" refers to the human mind and its effort to understand the world; the traditional area of philosophical endeavours. Hence, the foundations of current efforts to simulate human explanatory reasoning are discussed as are current limitations and opportunities for rule extraction from support vector machines.

1 Explanation: The Foundations

In a series of paper and books, Paul Thagard explores what it means to explain something (most recently Thagard and Litt forthcoming). Human thinking is essentially an ongoing, inner dialogue to explain why certain events do or do not happen or why things behave in a certain way. Explanation is closely linked to problem solving because the failure to explain an event or a certain outcome may trigger a problem solving episode. People explain to themselves and others why things are not working properly and what to expect if certain actions are taken. Explanation is a continuous cognitive process almost identical to thinking because humans are constantly explaining "why" things happen and "how" things work.

1.1 Forms of Explanation

It is useful to distinguish between various types of explanation, e.g. causal explanations that are acceptable answers to "why" questions as opposed to the step-wise explanations that are acceptable responses to "how" questions. It is also possible to ask for clarifications if certain facts are known already and more detail is required. Finally, there are those types of questions that are best answered by providing an example for an event or fact that is most typical and therefore helps to explain a whole set of observations.

Thagard and Litt (forthcoming) distinguish between three major explanatory processes:

- Providing an explanation from available information
- Generating new hypotheses that provide explanations
- Evaluating competing explanations

The four major theoretical approaches are: "*deductive*, using logic or rule-based systems; *schematic*, using explanation patterns or analogies; *probabilistic*, using Bayesian networks; and *neural*, using networks of artificial neurons" (Thagard and Litt forthcoming, p. 2).

The classical explanation is deductive and requires logical reasoning. What is to be explained (the explanatory target) follows from known facts by logically applying a set of rules ("Anyone with influenza has fever, aches and cough. You have influenza. So, you have fever, aches and cough", Thagard and Litt forthcoming, p. 4). This is the modus ponens, a simple, logical argument: If X is true then Y is true. X is true. Therefore Y is true.

Sometimes the explanatory target is only probable and more than one explanation is possible. Explanation is then closer to a conditional probability. Often, it is useful or required to find the best explanation which can be a complex process because whole "explanation structures" need to be examined to determine the best (Thagard, 1978).

Logical, deductive explanation requires a set of known facts as well as a set of IF ... THEN rules. Background knowledge is then expressed in the form of propositions which are used to explain an explanatory target. Here is a simple example following (Thagard and Litt forthcoming):

1. Anyone who completes a marathon has muscle pain and feels tired.
2. Person X has muscle pain and is tired.
3. Person X has completed a marathon.

The conclusion here obviously does not necessarily hold: there are many reasons why a person may have muscle pain and may feel tired, running a marathon is just one of them and maybe not the most probable explanation. Hence it is possible to have a loser association between propositions and an explanatory target. Again following the general argument in Thagard and Litt (forthcoming), it is possible to characterize *causal schemas* as an alternative to formal deduction. Here is a simple example:

1. Explanatory pattern: Typically, running a marathon causes muscle pain and tiredness.
2. Explanatory target: Person X has muscle pain and is tired.
3. Schema instantiation: Maybe person X has completed a marathon.

The example above invites already the application of probability theory and statistics. Again in the context of our simple example: the probability of muscle pain and the feeling of tiredness after a marathon is high. The marathon explains why person X feels pain and is tired. Here explanation is more like a conditional probability and the value of the explanation depends on known probabilities and the match with the schema.

Thagard and Litt (forthcoming) outline an additional way of modelling explanation and interestingly this is the use of artificial neural networks. Thagard and Litt (forthcoming) confirm that the neural approach by itself is not a theory of explanation; it is a method that simulates the cognitive processes that are part of explanatory thinking. Thagard and coworkers have used neural networks for cognitive modelling including the generation of explanations. In the current context, it has been demonstrated again and again that in particular feedforward neural networks are lacking explanatory power and hence rule extraction has to be applied. This is our point of departure from Thagard's argument.

In summary, the classic review is that explanation is a deductive argument including background knowledge and inference rules such as modus ponens. The inference rules allow the sequential application of "if-then-else" statements in order to justify an explanatory target. Whenever no precise knowledge is available, explanatory schemas or probabilistic rules can be used. But of course other forms of explanation are possible as well.

1.2 Analogy as a Form of Explanation

Every classroom teacher knows that at times it is very difficult to introduce a new theoretical concept. Even with the best of efforts it may not be possible for the class to grasp the theoretical elements that are being introduced. A single example, however, may change all of that and leads to an "aha" experience and complete understanding of the new material. So examples do have explanatory value and can be most useful, in particular if they are typical or even prototypical. In the current context, that his rule extraction from support vector machines, it may well be an objective to identify one or more examples that explain the behaviour of the machine (see Martens et al. and Nunez et al. in this volume). If this is not possible and precise background knowledge in the form of rules is not available, an *analogy* may be used for explanation.

An analogy requires the existence of a memory system to store and search cases which may have varying degrees of similarity. For instance, a case-based system could store various types of sporting events, including those that require a great deal of endurance. There are obviously similarities between a marathon and a triathlon and it is to nobody's surprise that both can cause tiredness and muscle pain immediately after the event. Even if we don't know that a particular person has just completed a marathon, the similarities between a long run and a multi-sports event explain why a marathon runner should be just as tired as a triathlete. Here, explanation is based on obvious and implicit similarities.

Analogies and schemas go hand in hand. Instead of storing all sort of sporting events we can have one schema for endurance sports which includes the immediate consequences including muscle pain and tiredness. For any given sports of this type, special features are replaced by variables which can be instantiated whenever an explanation is required. There could be a variable "physical effects" which can be instantiated by the two known consequences of endurance sports.

In summary, the following processes can generate explanations acceptable to humans:

- Logical deduction by use of inference rules
- Probabilistic rules including conditional probabilities
- Schemas based on the similarity between cases
- The provision of one or more examples which are typical or even prototypical

In artificial intelligence, several of these processes are often combined to arrive at systems that either use or generate explanations. The best example is *explanation-based learning* or *explanation-based generalization*. This form of learning has a long history in cognitive science and is often traced back to *Gestalt* theory, a branch of psychology popular in continental Europe in the first half of the twentieth century. Gestalt theory in turn has its roots in a rational form of philosophy which assumes that a significant part of our knowledge is innate and learning occurs at the periphery of knowledge only. That is, a significant amount of background knowledge is required for successful learning. On the other side, very few examples are necessary.

1.3 Explanation-Based Generalization

Explanation-based generalization is most interesting for the discussion here because it uses logical deduction based on the presentation of a single example. Since its invention twenty years ago, it has been modified to allow probabilistic reasoning and even the use of schemas. Explanation-based generalization had a significant impact on theory formation in artificial intelligence and early efforts to realize rule extraction from neural networks are linked to the attempt to build explanation-based learning systems. Hence, explanation-based generalization had a significant impact on the current understanding of what constitutes an explanation and will be briefly summarized here.

An explanation-based generalization system requires four components (1) the target concept, (2) the training example, (3) background knowledge in the form of rules and (4) the operationality criterion which defines what is to be learned or how learning should improve performance. There is an additional requirement that the training example is a positive instance of the target concept. Also, background knowledge must be both complete and consistent. Explanation-based generalization proceeds in two stages (1) explanation and (2) generalization. During the first stage a formal proof is constructed which demonstrates that the training example is a positive instance of the goal concept. If this is not the case, learning can fail. In the second stage, on the basis of the formal proof that the example is a positive instance of the goal concept, a new rule is formed and the knowledge is added to the rule base. Most importantly, and in contrast to statistical learning systems such as neural networks and support vector machines, the generalization that is the result of the learning episode is justified: it can be formally proven that the generalization holds given the training example, the goal concept and the background knowledge.

An example may be useful to explain explanation-based generalization: assume you are walking the streets and you see a car you have not seen before but you recognize as a BMW. The car has features you associate with BMWs but there are new, unexpected aspects as well. These new features immediately generate interest and start a learning process that is indeed an explanation process: the known attributes (this may be "shape of head lights", "company logo", etc.) are being used to explain that this particular car is a BMW. This is the first stage of explanation-based generalization. The process may continue with a generalization: A new class or concept is added to the background knowledge that includes the just seen car but maybe used to recognize other cars identical or similar to this one. As a matter of fact, this scenario invites a generalization as it is extremely unlikely that the just seen car is unique and no others of this type exit.

Explanation-based generalization is a very natural model for human learning. The previous paragraph describes an everyday observation and the learning process that is triggered by an observation. In addition, explanation-based generalization uses a "proof structure" (the sequence of rules that have been applied to prove that the training example is an instance of the goal concept) as the basis of an explanation. Hence, in the following chapters explanations are "rule sets" that explain "how" a certain decision was made and "why" it was made.

1.4 How and Why Explanations

As indicated earlier, it is useful to distinguish between various types of explanation, e.g. causal explanations that are acceptable answers to "why" questions in contrast to the step-wise explanations that are acceptable responses to "how" questions. Let's focus on how questions first.

A how explanation consists of a sequence of rules that map a given input to an output, in this case the input to a neural network or a support vector machine. In explanation-based generalization, a sequence of rules explains how the single training example is a positive instance of the target concept. Even though both rule extraction from neural networks and support vector machines generate rule sets, these rules are rarely applied in sequence. Very often rules are propositional in nature and include sets of inputs that result in a positive output. In this sense, rules can be independent and can even overlap. In rule extraction from neural networks, each rule includes a set of inputs that can result in a positive or negative output independently. The totality of the rule set explains how the neural network arrives at a decision.

"Why" explanations are typically used in expert systems. Here, the user may be engaged in an extensive dialogue and sometimes the system poses questions which are difficult to understand. After each question, the user has the option to ask "why" and the system will justify asking that particular question at this point in time. In this sense, "Why" explanation in expert systems does not include deep causal reasoning or the identification of a limited set of inputs that causes some output. A justification for a question is given.

1.5 Generating or Identifying the Best Explanation

Very often, more than one explanation is possible and explanations may even compete. Thagard and Litt (forthcoming) identify the evaluation of competing explanations as one of three major processes modelled by computational systems that aim to simulate human reasoning. The term "abduction" is well established in artificial intelligence and describes the inference to the best explanation as well as the generation of hypotheses (Thagard and Litt forthcoming). Thagard and Litt (forthcoming, p. 9) identify three criteria for the best explanation:

- Consilience: How much does a hypothesis explain?
- Simplicity: How many additional assumptions are required to carry out an explanation?
- Analogy: Are there hypotheses whose explanations are analogous to accepted ones?

2 Rule Extraction from Support Vector Machines: Aims and Significance

Andrews et al. (1995) describe the motivation behind rule extraction from neural networks. The five points outlined below, with the possible exception of "knowledge acquisition for expert systems", are relevant for the current effort to extraction comprehensible rules from SVMs. A brief review of Andrews et al. (1995) arguments will help to establish aims and significance for rule extraction from SVM techniques. For an introduction to support vector machines, see Martens et al. in this volume.

2.1 Provision of a "User Explanation" Capability

In symbolic artificial intelligence (AI), the term *"explanation"* refers to an *explicit structure* which can be used internally for reasoning and learning, and externally for the explanation of results to a user. Users of symbolic AI systems benefit from an explicit declarative representation of knowledge and traditionally, symbolic AI systems are deductive techniques: Reasoning (including classification) is from the *"generic"* (expressed in the form of general rules) to the *"specific"* (an instance or individual that is to be classified). Even learning, if it is based on a large amount of background knowledge, is deductive in symbolic AI systems.

The explanation capability of symbolic AI is based on intermediate steps of the reasoning process, e.g. a trace of rule firings, a proof structure, etc., which can be used to answer *"How"* questions. Gallant (1988) observes that the benefits of an explanation capability include a check on the internal logic of the system as well as enabling a novice user to gain insights into the problem at hand.

An explanation capability is considered to be one of the most important functions provided by symbolic AI systems. The ability to generate even limited explanations is essential for the user-acceptance of such systems (Davis et al., 1977). In contrast to symbolic AI systems, neural networks have no explicit declarative knowledge representation, and with exception of structured connectionist systems, neural networks do not perform deduction. Therefore neural networks have considerable difficulty in generating explanation structures and the situation is no different in support vector machines.

Traditionally, practitioners in the field of symbolic AI have experimented with various forms of user explanation, in particular *rule traces* (i.e. the sequence of rules or inference steps that are part of a problem-solving episode). However, it is obvious that explanations based on rule traces are too rigid and inflexible (Gilbert, 1989) because rules may not be equally meaningful to the user. In addition, rule traces always reflect the current structure of a knowledge base. Further, rule traces may have references to internal procedures (e.g. calculations); might include repetitions (e.g. if an inference was made more than once); and the granularity of the explanation is often inappropriate (Gilbert 1989; Andrews et al. 1995). A clear lesson from the use of rule traces in symbolic AI is that the transparency of an explanation is by no means guaranteed. For example, an explanation based on rule traces from a poorly organised rule base with perhaps hundreds of premises per rule cannot be regarded as *"transparent"*. Interestingly, it is an inherent problem of rule extraction from neural network techniques (in particular those that are learning-based), that a large number of rules with many antecedents are generated. Similar to the extended explanations based on rule traces in symbolic AI, the large rule sets extracted from neural networks offer limited or no explanation capability.

An additional example of the limitations of explanation capabilities in symbolic AI systems is described in Moore and Swartout (1989). In the field of expert systems practitioners have been linking "canned text" with rules and instead of providing the user directly with the trace of rules, the sequence of pre-prepared text elements has been used to facilitate comprehensibility. This type of user explanation is very rigid, systems always interpret questions in the same way, and there are no adequate response strategies. Although efforts have been made to take advantage of natural-language dialogues including mixed initiatives, user-models and explicitly planned explanation strategies, there is little doubt that these systems are inflexible and rigid (Andrews et al., 1995).

While the integration of an explanation capability (via rule extraction) within a trained neural network or SVM is crucial for user acceptance, such systems must avoid the problems already encountered in symbolic AI.

2.2 Transparency

The creation of a *"user explanation"* capability is the primary objective for extracting rules from neural networks and SVMs, with the provision of "transparency" of the internal states of a system a close second. Transparency means that internal states of the machine learning system are both accessible and can be interpreted unambiguously. Such transparency would allow the exploration of regions in generalisation space which may lead to erroneous or sub-optimal decisions.

Such a capability is mandatory if neural network or SVM based solutions are to be accepted into *"safety-critical"* problem domains such as air traffic control, the operation of power plants, medical surgery, etc. Rule extraction offers the potential for providing such a capability (Andrews et al. 1995).

2.3 Software Verification

If neural networks or SVMs are to be integrated in larger software systems that need to be verified, then clearly this requirement must be extended to all components, including the ANNs and SVMs. Currently, rule extraction algorithms do not allow for verification, i.e. they do not *prove* that a machine learning system behaves according to some specification. However, rule extraction algorithms provide a mechanism for either partially or completely *"decompiling"* a neural network or SVM. This is about half-way to software verification because it allows for a comparison between the extracted rules and the specification.

2.4 Improving Generalisation

If a limited or unrepresentative data set has been used in the ANN training process, it is difficult to determine *if and when generalisation fails for specific cases* even with evaluation methods such as cross-validation. By expressing learned knowledge as a set of rules, an experienced user can anticipate or predict a *generalisation failure* (Andrews et al., 1995). It may also be possible to identify regions in input space that are not represented sufficiently in the data set and need to be supplemented (Andrews et al. 1995).

2.5 Data Exploration and the Induction of Scientific Theories

This has been one of the primary objectives for rule extraction from neural networks and is essential for data mining and knowledge discovery. As Craven and Shavlik (1994) write *"a (learning) system may discover salient features in the input data whose importance was not previously recognised"*. If a neural network or SVM has learned important and possibly non-linear relationships, these relationships are encoded incomprehensibly as weight vectors, support vectors and additional parameters. Within the context of discovering new relationships, rule extraction algorithms significantly enhance the data mining capabilities of neural networks and SVMs.

3 Translucency and Rule Quality

Over the last years, a number of studies on rule extraction from support vector machines have been introduced. The research strategy in these projects is often based on this idea: develop algorithms for rule extraction based on the perception (or "view") of the underlying SVM which is either explicitly or implicitly assumed within the rule extraction technique. In the context of rule extraction from neural networks the notion of "translucency" describes the degree to which the internal representation of the ANN is accessible to the rule extraction technique (Andrews et al. 1995; Tickle et al. 1998). More broadly, a taxonomy for rule extraction from neural networks has been introduced (Andrews et al. 1995; Tickle et al. 1998) which includes five evaluation criteria: translucency, rule quality, expressive power, portability and algorithmic complexity. These evaluation criteria are now commonly used for rule extraction from SVMs.

It is important to develop new techniques for rule extraction from support vector machines, including those that are based on SVMs only and do not require any other machine learning technique. In particular support vector machines that allow the generation of structured outputs (Taskar et al. 2005) can be used to generate rule sets not unlike those extracted from neural networks. This represents a clear advancement since user explanation is realized by an SVM and not by a technique with a different representational bias. In addition, methods for the extraction of high quality rule sets from SVMs trained on high-dimensional data are required.

The following briefly describes the first two of the five evaluation criteria for rule extraction from neural networks (Andrews et al. 1995; Tickle et al. 1998) which are then discussed in the context of rule extraction from SVMs.

3.1 The Neural Network Case

Translucency describes the degree to which the internal representation of the ANN is accessible to the rule extraction technique. At one end of the translucency spectrum are those rule extraction techniques which view the underlying ANN at the maximum level of granularity, i.e. as a set of discrete hidden and output units. Craven and Shavlik (1994) categorized such techniques as "decompositional". The basic strategy of decompositional techniques is to extract rules at the level of each individual hidden and output unit within the trained ANN. In general, decompositional rule extraction techniques incorporate some form of analysis of the weight vector and associated bias (threshold) of each unit in the trained ANN. Then, by treating each unit in the ANN as an isolated entity, decompositional techniques initially generate rules in which the antecedents and consequents are expressed in terms which are local to the unit from which they are derived. A process of aggregation is then required to transform these local rules into a composite rule base for the ANN as a whole (Tickle et al. 1998).

In contrast to the decompositional approaches, the strategy of pedagogical techniques is to view the trained ANN at the minimum possible level of granularity, i.e. as a single entity or alternatively as a "black box". The focus is on finding rules that map the ANN inputs (e.g. the attribute/value pairs from the problem domain) directly to outputs (Tickle et al. 1998). In addition to these two main categories, Andrews et al. (1995) also proposed a third category which they labelled as "eclectic" to accommodate those rule extraction techniques which incorporate elements of both the decompositional and pedagogical approaches.

A number of authors have studied the algorithmic complexity of extracting rules from feedforward neural network. Here is a brief summary of results:

- Decompositional approach: The basic process of searching for subsets of rules at the level of each (hidden and output) unit is *exponential in the number of inputs to the node.*
- Heuristics are invoked to limit the depth to which the space is explored.
- Golea (1996) showed that *extracting the minimum DNF (disjunctive normal form) expression from a trained feedforward net is hard in the worst case.*

- Furthermore, Golea (1996) showed that the Craven and Shavlik (1994) algorithm is *not* polynomial in the worst case.
- This result does not apply to single-layer networks; however, extracting the best N-of-M rule from a single-layer network is again hard.

Rule extraction from neural networks early on adopted criteria for the quality of the extracted rules. The set of criteria for evaluating rule quality includes (Andrews et al. 1995):

1. Accuracy
2. Fidelity
3. Consistency, and
4. Comprehensibility of the extracted rules

A rule set is considered to be accurate if it can correctly classify a set of previously unseen examples from the problem domain (Tickle et al. 1998). Similarly a rule set is considered to display a high level of fidelity if it can mimic the behaviour of neural network from which it was extracted by capturing all of the information represented in the ANN. An extracted rule set is deemed to be consistent if, under differing training sessions, the neural network generates rule sets which produce the same classifications of unseen examples. Finally the comprehensibility of a rule set is determined by measuring the size of the rule set (in terms of the number of rules) and the number of antecedents per rule (Tickle et al. 1998).

3.2 Translucency and Rule Quality Applied to Rule Extraction from SVMs

Most current studies on rule extraction from SVMs focus on decompositional extraction; however, learning-based approaches are also available (Barakat and Diederich 2005). The idea is simple: learn what the SVM has learned. For this purpose a data set is divided into two or more parts. The first set is used to train the SVM to completion. The second set does not include targets, the inputs are presented to the SVM and the output is obtained from the SVM. Inputs and outputs combined represent a new data set that is used for a second machine learning episode by use of a machine learning system that produces rules as output.

Hence, pedagogical rule extraction from SVMs is trivial, in particular if the data set is low-dimensional. Support vector machines have been designed to process high-dimensional input data. Typical examples are text, speech and image classification. Yet most of the studies available on rule extraction from support vector machines use benchmark data sets that include a limited number of features only. As a result, SVMs are not being used in their core application area and hence rule extraction results are not very meaningful.

It is very easy to illustrate the limitations of current studies on rule extraction from SVMs by use of an example: text classification. SVMs can achieve good performance with very simple text representation formats such as the "bag-of words" (BOW) technique. BOW uses a document-term matrix such that rows are indexed by the documents and columns by the terms (e.g. words). SVMs allow the classification of texts of differing lengths; hence, document vectors may differ greatly in the number of elements.

A disadvantage of the BOW representation is that after successful classification, it may not be obvious *what* has been learned. For instance, an author may have a preference for certain topics and as a result, an SVM trained on an authorship identification problem in reality may perform topic detection. This problem has lead to various techniques to eliminate content from the BOW input, for instance by replacing content words with lexical tags (categories).

Given the fact that it is not at all obvious what contributes to classification in case of a BOW input representation, rule extraction from support vector machines is presented with a special opportunity. However, the number of features can be very large: e.g. all words that exist in a given natural language. While a combination of words constitutes meaning in a natural language, BOW and hence classification is based on words in isolation. This is a significant problem with regard to rule quality: The antecedents in a rule include individual words completely out of context. As the set of antecedents includes completely unrelated words, *human or semantic comprehensibility* is low.

4 An Alternative View on Rule Extraction: Information Retrieval

The introduction to explanation in Sect. 1 neglected one very important aspect: Explanation is frequently based on an interaction between two persons (e.g. a teacher and a student) or a machine and a human (e.g. in tutorial or help systems). Cawsey (1993) uses a very simple definition of explanation: "In general an explanation is something which makes some piece of knowledge clear to the hearer.... The explanation is complete when the hearer is satisfied with the reply and understands the piece of knowledge" (Cawsey, 1993, p. 1). Hence, explanation is based on an "information need" and essentially is a dialogue. As part of this dialogue, explanatory targets may change and may be refined.

Frequently, it is possible to retrieve more than one explanation (i.e. rule set) from a given SVM or neural network. In this case, it is often necessary to select the best explanation. There are two main reasons why it is possible to generate multiple explanations from or for a given machine learning system (1) Rule extraction methods include parameters that need to be initialized and the selection of certain options or values for variables results in different rule sets. (2) Rules can be expressed in different ways, e.g. a rule set with few rules that have many antecedents can be re-written as a rule set with many, simple rules, i.e. rules with few conditions. While it is generally acknowledged that a rule set with few rules and a limited set of antecedents provides best explanation capability, it is far from obvious that this is always the case. Indeed, the *information need* of the user has to be taken into consideration and the user may *interact* with the machine learning system by use of rule extraction.

A user may have a need for multiple explanations (rules sets) because the objective is to explore the generalization space of the underlying SVM or neural network. Several rule sets, if considered in turn, may offer best transparency of the ANN or SVM. In another scenario, a user may be interested in the single-best explanation in the form of a few simple rules. In addition, the user may be interested in exploring different parameter sets which lead to different learning results and consequently to different rule sets. It is difficult to consider rule quality criteria without reference to the information need of the user.

The concept of *information need* is central to the discipline of information retrieval. The performance of an information retrieval system, e.g. an Internet search engine, is traditionally evaluated by use of "precision" and "recall". Precision is the probability that a document predicted to be genuine truly belongs to this class. In other words, a document that has been retrieved from a database truly matched the information need of the user. Recall is the probability that a genuine document is classified into this class. Less formally, high recall is given if all documents that satisfy the information need of the user are indeed retrieved from a database.

It is obviously desirable to have high recall and precision simultaneously but this is difficult to achieve in information retrieval. A trade-off exists between large recall and precision. By adjusting a parameter, e.g. by altering the cost of misclassification, recall may be increased at the cost of decreasing precision and vice versa.

The observation that multiple explanations can be extracted from a trained SVM or neural network leads to the application of information retrieval concepts to rule extraction (see the case study below). In the context of rule extraction from either support vector machines or neural networks, high precision represents the scenario that the rule sets extracted are relevant to the user, i.e. match his or her information need. Recall refers to the question "how many relevant explanations that can be extracted from the SVM or neural network are indeed being extracted?" Is it possible to generate all possible explanations by way of rule extraction that match the information need of the user?

The ideas outlined above may lead to the application of additional performance measurements that are commonly used in information retrieval and that are based on precision and recall. The view that a single explanation is to be extracted from an SVM or neural network is a simplification. It may be desirably to extend the notion of "rule quality" to include assessments of multiple explanations in relation to the information need of a user.

5 A Case Study

To illustrate the problem (and opportunity) faced by rule extraction from SVMs when applied to text classification, support vector machines are used to classify business news articles from the Persian Gulf with regard to emotional content. A total of 914 news articles are used for this experiment.[1] In addition, experiments on authorship attribution (identifying the author of a text) and topic classification are performed.

The pre-processing includes two parts: text extraction and feature selection. Text extraction is performed by lexical analysis to strip all non-word annotations and to convert the text into a list of words or tokens. This step can be summarised as follows: (1) upper case letters are converted to lower case, (2) all words containing non-letter characters are removed including hyphenated words and words with an underscore, (3) all punctuations are replaced with space characters to be treated as token delimiters, (4) author identities are extracted, (5) the texts are converted to a "bag-of words" (BOW) representation.

In addition, all class identifiers are removed from the articles in the BOW format before they are used to generate a fixed vocabulary: author names and words used for topic and emotion identification are removed (see the section below on the clustering process used to generate targets for supervised machine learning). After the text extraction process, a fixed length vocabulary is built from the set of all extracted news articles through a feature selection process. Firstly, stopword removal and stemming[2] are performed on each extracted text. Secondly, document frequency thresholding is used to reduce further the feature vector space. Words occurring once only are removed.

After the vocabulary generation process, for each class (four authors, emotions and topics), the extracted texts are (1) labelled with the class and (2) mapped to an SVM data file in which each line represents a news article. Each row includes a label that indicates whether the article belongs to the target class or not.

Pre-processing of the text samples, including elimination of frequent words (using an edited list of the 6,500 most frequent words in English) led to the development of lists of words that are low frequent, but included some words with topical or emotional content which are common. Simple clustering techniques are used to extract topic and emotion information from texts to perform supervised learning.

[1] This is joint work with Insu Song, Aqeel Al Ajmi, Jihan Zhu, Imran Fanaswala and Mark Pedersen.

[2] Stopword removal refers to the elimination of function words such as articles (the, a) Stemming identifies the root of a word, e.g. "goes" will be converted to "go".

The following is a description of the algorithm used to identify topical and emotive information:

For all business news articles

1. Generate a ranked list of n words that are not in a stoplist (comprising an edited list of the 6,500 most spoken words in English)
2. Apply cluster analysis to the ranked word lists extracted from the documents
3. Identify words that are high-frequent in clusters

The method in step 2 is described by Chiu et al. (2001) who proposed a conceptual or model-based approach to hierarchical clustering. The method includes a two-step strategy to determine the number of clusters. The model associated with a cluster covers both numerical and categorical attributes and constitutes a mixture of Gaussian and multinomial models. The distance between two clusters is defined as a decrease in log-likelihood caused by merging of the two clusters under consideration. The process continues until a stopping criterion is met. As such, determination of the best number of clusters is automatic (Berkhin, 2002).

The emotion categories extracted by this process are "boom", "confident", "regret" and "demand". The topic categories are "asian economy", "oil price", "stock" and "gas". As in previous studies, authorship attribution succeeds at a very high level. All four attempts to identify the author of the texts are successful (Level-one-out cross-validation estimates of the performance: error <2%, precision 100% and recall 70–97%). Topic detection performance is lower and on par with the emotion classification results. Two of the topic detection learning results are relatively poor as are two of the attempts to discover the emotion express in the text.

Pedagogical rule extraction from SVMs as outlined in Barakat and Diederich (2005) is applied to the trained SVMs (one authorship attribution as well as one topic detection and emotion classification problem). The procedure for rule extraction is as follows:

1. Divide data in two or more sets
2. Train SVM on a subset of data A
3. Get SVM predictions on subset B
4. Combine inputs from subset B with SVM predictions
5. Train a symbolic machine learning system on the new data set
6. Obtain rules from the symbolic machine learning system (in this case a decision tree learner and a classification and regression tree)

Since the decision tree learning system cannot efficiently deal with high-dimensional input spaces, the first 200 features are used for See 5 learning only. The following rules (Table 1) were extracted in one run from an SVM

Table 1. Rules extracted from an SVM trained on an authorship identification task

Rule 1:	interest <= 0
	percent > 0.079
	product <= 0.086
	-> Target author
Rule 2:	us <= 0
	price <= 0.027
	compani <= 0.031
	set > 0.061
	-> Target author
Rule 3:	market <= 0.075
	number > 0.1
	-> Target author
Rule 4:	percent <= 0.079
	product <= 0.086
	number <= 0.1
	set <= 0.061
	-> Other author
Rule 5:	us > 0
	number <= 0.1
	-> Other author
Rule 6:	price > 0.027
	-> Other author
Rule 7:	market > 0.075
	-> Other author
Default class:	Other author

trained on an authorship identification problem. Please note that in this case, the SVM classifications for inputs in the training set A are used.

The training error of the decision tree learning method is low: 2.3%. There are rules for the positive and negative classes and rule quality is high: seven rules and 17 antecedents in total (of the 200 possible features, only nine occur in the rules). The rules include content words (or rather word stems) only since pre-processing eliminated all function words. Word frequency distribution over function words is relevant for authorship attribution; nevertheless, even without function words SVM and decision tree learning succeeds.

The following rules were extracted from the topic-detection SVM (Table 2). Again, note that SVM classifications for inputs of the training set A are used as targets for decision tree learning.

The See 5 learning error is 5.0% and rule quality is obviously slightly reduced. Rule 6 is particularly problematic, a point which will be discussed in detail further below.

Finally, rules were extracted from an SVM trained on an emotion classification problem (Table 3). The learning result for this SVM is acceptable (leave-one-out cross-validation result: error 8.6%, precision 94.9%, recall 70%).

Table 2. Rules extracted from an SVM trained on a topic detection problem

Rule 1:	exchang > 0.089
	valu <= 0.024
	-> Target topic
Rule 2:	share > 0.083
	-> Target topic
Rule 3:	investor > 0.053
	advanc > 0.043
	-> Target topic
Rule 4:	market > 0.046
	fed <= 0.076
	-> Target topic
Rule 5:	market <= 0.046
	share <= 0.083
	exchang <= 0.089
	advanc <= 0.043
	-> Other topic
Rule 6:	oil <= 0.318
	investor <= 0.033
	share <= 0.083
	world < = 0.077
	foreign <= 0.091
	jordan <= 0.053
	exchang <= 0.089
	long <= 0.068
	export <= 0.097
	posit <= 0.053
	hous <= 0.058
	-> Other topic
Rule 7:	fed > 0.076
	-> Other topic
Rule 8:	german > 0.057
	japan <= 0.032
	-> Other topic
Rule 9:	share <= 0.083
	properti > 0.183
	-> Other topic
Rule 10:	dollar > 0.336
	-> Other topic
Rule 11:	sterl > 0.078
	-> Other topic
Default class:	Other topic

In contrast to the two cases above, SVM classifications for inputs of the test set B are used as target for decision tree learning.

The test set includes 120 cases and the See 5 learning error is 1.7%. Tenfold cross-validation reveals a test error of 13.3%, however, practically all positive examples are misclassified. Clearly, there are an insufficient number of positive examples in this data set (8 out of 120).

It is possible to use sub-sampling techniques or boosting trials to elicit more words (antecedents in rules) in order to identify relevant features or to clarify the classification task. Ten boosting trials lead to rule sets that confirm the result above on the one hand but also identify additional features (Table 4). The boosted decision tree classifier correctly learns all cases.

Table 3. Rules extracted from an SVM trained on an emotion classification problem

`Rule 1:`	`suppli > 0`
	`estat > 0.032`
	`-> Target emotion`
`Rule 2:`	`close > 0.02`
	`estat > 0.032`
	`-> Target emotion`
`Rule 3:`	`estat <= 0.032`
	`-> Other emotion`
`Rule 4:`	`estat > 0.032`
	`-> Other emotion`
`Default class:`	`Other emotion`

Table 4. Rules extracted by use of a tenfold boosting run from an SVM trained an emotion classification problem

`Rule 0/1:`	`suppli > 0`
	`estat > 0.032`
	`-> Target emotion`
`Rule 0/2:`	`close > 0.02`
	`estat > 0.032`
	`-> Target emotion`
`Rule 0/3:`	`estat <= 0.032`
	`-> Other emotion`
`Rule 0/4:`	`estat > 0.032`
	`-> Other emotion`
`Default class:`	`Other emotion`
`Rule 1/1:`	`war > 0.022`
	`-> Target emotion`
`Rule 1/2:`	`war <= 0.022`
	`-> Other emotion`
`Default class:`	`Other emotion`

Table 4. (Continued)

Rule 2/1:	al > 0.024
	high > 0.028
	chang <= 0
	-> Target emotion
Rule 2/2:	chang > 0
	-> Other emotion
Rule 2/3:	high <= 0.028
	-> Other emotion
Rule 2/4:	al <= 0.024
	-> Other emotion
Default class:	Other emotion
Rule 3/1:	oil > 0.222
	-> Target emotion
Rule 3/2:	estat > 0.046
	-> Target emotion
Rule 3/3:	oil <= 0.222
	estat <= 0.046
	-> Other emotion
Default class:	Other emotion
Rule 4/1:	uae > 0.193
	-> Target emotion
Rule 4/2:	higher > 0.154
	-> Target emotion
Rule 4/3:	higher <= 0.154
	uae <= 0.193
	-> Other emotion
Default class:	Other emotion
Rule 5/1:	hous > 0
	-> Target emotion
Rule 5/2:	hous <= 0
	-> Other emotion
Default class:	Other emotion
Rule 6/1:	compani <= 0.05
	gulf <= 0
	estat > 0.032
	-> Target emotion
Rule 6/2:	suppli > 0.058
	-> Target emotion
Rule 6/3:	suppli <= 0.058
	estat <= 0.032
	-> Other emotion

(*continued*)

Table 4. (Continued)

Rule 6/4:	compani > 0.05
	-> Other emotion
Rule 6/5:	gulf > 0
	-> Other emotion
Default class:	Other emotion
Rule 7/1:	dubai <= 0.087
	monei > 0.05
	-> Target emotion
Rule 7/2:	dubai > 0.087
	-> Target emotion
Rule 7/3:	dubai <= 0.087
	monei <= 0.05
	-> Other emotion
Default class:	Other emotion
Rule 8/1:	declin > 0.09
	-> Target emotion
Rule 8/2:	iraq > 0.028
	-> Target emotion
Rule 8/3:	iraq <= 0.028
	declin <= 0.09
	-> Other emotion
Default class:	Other emotion
Rule 9/1:	project > 0.116
	-> Target emotion
Rule 9/2:	real > 0.177
	-> Target emotion
Rule 9/3:	real <= 0.177
	project <= 0.116
	-> Other emotion
Default class:	Other emotion

It is obvious that different machine learning techniques that “learn what the SVM has learned” may produce different results (the value of extracting a range of different rule sets from an SVM has been outlined above). If a classification and regression tree is used to generate the explanation, two simple rules are generated (Table 5).

The rule sets above allow a number of interesting observations. In the case of the “authorship attribution” rule set (Table 1), it is not at all clear what contributes to the identification of this author. All documents are business news articles, hence the question is: does this author focus on a particular

Table 5. Rules extracted from an SVM by use of a classification and regression tree. The SVM has been trained an emotion classification problem

```
Rule 1: valu < 0.047
        invest < 0.275
        -> Other emotion
        invest >= 0.275
        -> Target emotion
Rule 2: valu >= 0.047
        currenc < 0.016
        -> Other emotion
        currenc >= 0.016
        -> Target emotion
```

topic (e.g. the US stock market) or do the rules capture relevant features of the author's style?

The rule sets obtained from the SVM trained on the emotion classification problem are even more intriguing: Rule quality is very high, rule extraction is consistent and the learning results are good. However, *none of the antecedents express emotion in any way at all!* Even though all SVM and decision tree learning results are acceptable (with exception of the "extreme" k-fold cross-validation), *the rules appear not to be linked to the task at hand*: classify documents into categories that express the emotion of a text. Since it is one of the objectives of rule extraction to explain "how" classification is realized by an SVM, the question must be asked to what extend the above rule sets help to provide an answer.

The experiments outlined above invite a number of objections that should be discussed in the context of evaluating the rules. First of all, the decision tree learner was trained by use of the 200 most frequent features (word stems) only, the SVM considers up to 6,484 attributes. It is possible that the vast majority of features that are not input to the decision tree learner are relevant for SVM learning (the feature ranked 200 in the corpus has an absolute frequency of 291; generally, word stems that occur twice or more in the entire data set are being considered for SVM learning). Joachims (1998) established that the majority of words with low frequency in a corpus do contribute to a text classification task. Hence, it is possible that the example rule sets above do not completely capture what the SVM has learned.

Since it is difficult to train a decision tree learner or a classification and regression tree on a high-dimensional data set, it may be argued that this is a case for a "decompositional" extraction method that does not rely on a non-SVM learning technique so obviously insufficient for high-dimensional problems. It is unlikely; however, that such an approach offers a solution to the problem. Obviously, individual words or word stems taken out of context cannot provide a human comprehensible explanation.

The four rules in Table 3 all include the antecedent “estat”. This is a word stem generated from the word “estate” which occurs 362 times in the corpus. Practically all of the occurrences are in the context of “real estate”, and since this is a corpus including business news articles from the Persian Gulf, “real estate” refers to the current construction boom in Dubai. Table 3 does not include any reference to “real” (as in “real estate” or “Dubai”). The boosting trials in Table 4, however, extract the words “real” and “Dubai”. Similarly, the word stem “suppli” originated from “supplies” or “suppliers” and is used in the context of “oil supplies”.

In a further experiment, a neural network was trained on the data used for rule extraction (Tables 3 and 4). The neural network achieved more than 90% accuracy in various configurations. For instance, a feedforward neural network with a 200 unit input layer, a first hidden layer of 7 and a second hidden layer with nine units as well as a one unit output layer achieves an accuracy of 90.5%. After completion of the training, sensitivity analysis was performed on the neural network to obtain information on the relevance of input features. Sensitivity analysis ranks all inputs to the neural network (in this case 200) according to the relevance of the feature for the classification task. In various runs with different neural network architectures, “real” and “estat” were both ranked by sensitivity analysis among the top 5 input features to the neural network. “suppli” was also ranked as a top feature by sensitivity analysis. These results confirm the relevance of the features in Table 3, the rules in Tables 3 and 4 as well as the interpretation provided above.

Finally, the classification and regression tree rules in Table 5 use the word stem “valu” which occurs 355 times in the corpus. The original word forms are “value”, “valuation”, and “devaluation” and so on. “invest” is one of the most frequent word stems (1,543 occurrences) similar to “currenc” (974 occurrences). Obviously, the original words appear in various contexts.

The rules in Tables 3–5 point only indirectly to the criteria the SVM utilises for classification. Again, it is important to emphasise that the SVM in question has been trained on an emotion classification task, yet, rule extraction does not reveal any word that is linked to affect or emotion. Sensitivity analysis after neural network training on the same data, however, extracts some words that do have emotional content such as “profit”. There is some indication that in reality, the SVM performs a topic classification task. The rule extraction process could be considered a success since a possible confusion between a topic and emotion classification task has been discovered. Yet again, in this particular case the SVM does perform emotion classification to some extent as indicated by the ROC curve in Fig. 1 with an “area under the curve” AUC of .86. The extracted rules should be used in full view of the learning result of the SVM in order to explain what the support vector machine has learned.

Is it possible to say that SVM emotion classification by use of a “bag-of-words” representation has failed because no emotion word occurs in the extracted rules? No, it is not possible to say this conclusively because the

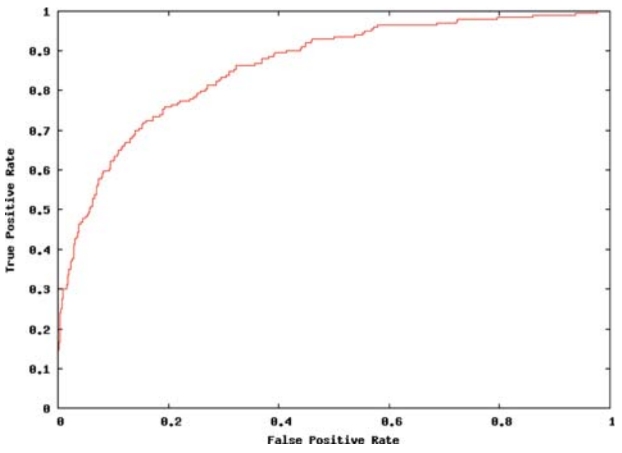


Fig. 1. The ROC curve for the emotion classification SVM. Please note that this ROC curve has been obtained from an SVM trained on a slightly larger data set (914 patterns). The SVM used for rule extraction has been trained on 734 examples

SVM has been trained on a high-dimensional data set while the decision tree learner used for extraction utilizes 200 features only. Also, the boosting runs in Table 4 generated rules including words like "war", "iraq", "declin" (for decline) and potentially several others which do have emotional content at the point in time this study was performed. Nevertheless, there is evidence that input dimensionality is crucially important for rule extraction from SVMs!

In order to enhance the explanatory value of the rules extracted from the SVMs, it may be beneficial to train an additional SVM on "bigram frequencies" and to extract rules from this SVM. "Bigrams" are sequences or combinations of words that appear in sentences. For instance, the sentence "Real estate in Dubai is expensive" includes "real estate" and "in Dubai" as bigrams. If stopwords such as "in" and "is" are eliminated, bigrams such as "real estate" and "Dubai expensive" would be generated. As part of the pre-processing of documents for SVM learning, the frequency of occurrence of bigrams would be calculated and would be utilised in the SVM data set. For instance, bigrams such as "real estate" would be attributes in the SVM data set and the normalised frequency of occurrence would be the value of the feature.

Rule extraction from SVMs would then generate rules including antecedents such as "real estate", "in Dubai" or "Dubai expensive". Due to the added context, rules with these antecedents would be much more comprehensible. Even if the SVMs trained on bigrams frequencies do not perform as well as those trained on a simple "bag-of-words" representation, the bigram rules can be compared to those extracted from an SVM trained on "bag-of-words". It is common in artificial intelligence systems to have an independent explanation system, i.e. a system separate from the core inference machine. In a similar sense, the rules extracted from the bigram SVM can be used for explanatory purposes only.

The observations above lead to a new rule quality criterion: *semantic comprehensibility*. In the cases outlined above, rule quality as originally formulated (Tickle et al. 1998) is high due to the limited number of antecedents and rules, yet *comprehensibility* for the user is low. Therefore, it is necessary to introduce a new rule quality criterion, semantic comprehensibility.

6 A Classification System for Rule Extraction from SVMs

Rule extraction from support vector machines requires evaluation criteria that emphasize data (Table 6). SVMs have demonstrated very good performance when trained on data sets with high-dimensional inputs; significant application areas are image classification (including face recognition), bioinformatics and text classification. At this point in time, many Internet search engines use support vector machines.

In light of the discussions above, the following dimensions are proposed:

1. Translucency: This dimension as originally proposed in Andrews et al. (1995) and Tickle et al. (1998) continues to be useful, even with the

Table 6. A classification system for rule extraction from SVMs

Attribute	Type	From	To
Translucency	Continuous	Decompositional	Pedagogical
Data	Continuous	Low-dimensional	High-dimensional
Expressiveness	Discrete	Boolean	First-order predicate logic
Rule quality			
Number of rules	Continuous	1	No upper limit
Number of antecedents	Continuous	1	No upper limit
Semantic comprehensibility	Discrete	Yes	No
Fidelity	Continuous	0%	100%
ROC Fidelity	Continuous	Low	High
Accuracy of rules	Continuous	0%	100%
Precision of rules	Continuous	0%	100%
Recall of rules	Continuous	0%	100%
Complexity	Continuous	Linear	Exponential
Non-SVM extraction	Discrete	Yes	No

introduction of rule extraction techniques for high-dimensional data sets. However, there are important questions with regard to the learning-based or pedagogical approach: Many of the rule-based learners have very different representational biases compared to support vector machines. In addition, many are less suitable for high-dimensional inputs, again in comparison to SVMs. These limitations have been discussed already by use of the case study.

2. Data: From low to high-dimensional input space. Many data mining practitioners would probably agree with the view that it is possible to *engineer* a neural network with similar learning performance to an SVM for low-dimensional data sets. As a matter of fact, SVMs are not necessarily the method of choice for these data sets, there are many alternatives. This includes those cases with relationships between attributes that are best expressed as a decision tree. Hence, the view here is that SVMs are the primary choice for high-dimensional inputs and rule extraction techniques should work in these cases.
3. Expressiveness of the extracted rules. Rule extraction from neural networks has previously almost exclusively been used to generate propositional rule sets (Hayward et al. 2000). While this is sufficient for many applications where rule sets can be effectively used, it is clearly desirable to provide a more general explanation capability. Hayward et al. (2000) describe an approach to representing a neural network as a PROLOG logic program, where the activation values of hidden and output units are equated with the truth value of predicates. The technique addresses several issues. Hayward et al. (2000) describe a process whereby Boolean formulae are translated into a first-order representation consisting of predicates, rules and facts. This is a field that is largely unexplored in the context of rule extraction from support vector machines; however, it is conceivable that SVMs with structured output (Taskar et al. 2005) will lead to complex rule sets and languages beyond and above propositional logic. Also, please see the chapter by Torrey et al. in this volume.
4. Rule Quality: This category includes accuracy, fidelity and comprehensibility. "Semantic comprehensibility" is given if minimal rules sets with concise rules are extracted from SVMs trained on high-dimensional data. Fidelity may be extended to "ROC fidelity", e.g. if and when the SVM and the rule set exhibit the same classification behaviour with modified cost functions.

   It is crucial to consider the sub-category "semantic comprehensibility" and the case study above is designed to outline some of the relevant issues. Given high-dimensional data sets, features in isolation have limited or no explanation capability. It has been proposed earlier to extract multiple rule sets from SVMs to explore the full feature set that contributes to a classification. In many ways, it is the user who decides which rule set has value and hence, the notion of "information need" has been adopted from information retrieval. Semantic comprehensibility is not formally defined here and is not proposed as a quantitative measure. At this stage, "semantic comprehensibility" refers to the ultimate goal to extract user-comprehensible rule sets from *any* SVM.

5. Complexity of the extraction. To date, there has been no systematic study on the algorithmic complexity of rule extraction from SVMs. The results that have been obtained for rule extraction from neural networks (some have been summarised above) are not applicable because they rely on (1) the structure of the neural network (single or multi-layer) and (2) properties of a learning algorithm such as backpropagation. Many of the current approaches for rule extraction from SVMs include heuristics and/or machine learning or statistical techniques that are interchangeable. For instance, the learning-based rule extraction from SVM technique used in the case study above uses either a decision tree learner or a classification and regression tree. Núñez et al. (2002) use clustering techniques with the aim to identify regions in decision space that can be translated to rules. Opportunities and limitations of the rule extraction from support vector machines enterprise are yet to be fully explored.
6. Non-SVM extraction: It is important to develop new techniques for rule extraction from support vector machines, including those that are solely based on SVMs and do not require any other machine learning technique. In particular, support vector machines that allow the generation of structured outputs (Taskar et al. 2005) can be used to generate rule sets not unlike those extracted from neural networks. This represents a clear advancement since user explanation is realized by an SVM and not by a technique with a different representational bias. In addition, methods for the extraction of high quality rule sets from SVMs trained on high-dimensional data are required.

By way of example, it is easily possible to apply this classification system to the simple case study provided above, the emotion classification problem.

Table 7 clearly demonstrates the limitations of the algorithm used in the case study. Rules lack expressiveness, the extraction process does not really consider the dimensionality of the data set, and while rule quality is quite good, the algorithmic complexity of the extraction process is not satisfactory.

7 Conclusions and Future Challenges

It is obviously not possible to discuss all aspects of rule extraction from support vector machines in this brief introduction. There is one area in particular that would deserve a fuller consideration. This is the use of committee

Table 7. The classification of the case study in Sect. 5

Translucency	Pedagogical
Data	Low-dimensional (SVM is trained on high-dimensional data)
Expressiveness	Boolean
Rule quality	
Number of rules	2–11
Number of antecedents	5–28
Semantic comprehensibility	No
Fidelity	Not tested
ROC Fidelity	Not tested
Accuracy of rules	90%
Precision of rules	66%
Recall of rules	90%
Complexity[3]	$\mathrm{O(mn \log n) + O(n(\log n)^2)}$
Non-SVM extraction	No

machines or ensemble learning approaches.[4] Obviously, it is more difficult to extract comprehensive rules from a set of support vector machines or an ensemble of machine learning techniques of different type. The requirements for comprehensibility are even harder to meet if more than one classifier is involved.

A first attempt has been made in a project to predict the return of stocks in the US market.[5] An SVM was trained to accept input from various machine learning techniques to predict the next day return of shares. The machine learning methods includes various types of neural networks, support vector machines as well as an implementation of Ripper. The target for the SVM-based committee machine is the next day return. Rules were then extracted which took the following form:

> IF the value of the prediction of C4.5 is equal to -1 THEN the SVM committee machine classifies the sample as -1, Otherwise, it classifies the case as $+1$.

Given the low-dimensionality of the input, SVM learning (i.e. the committee machine) did not significantly improve the overall results and the SVM tends to agree with the decision tree learner C4.5

[3] Decision tree learning complexity for C4.5 according to Witten and Frank (1999, p. 168). n is the number of samples and m the number of attributes. See 5 has been used in this study.

[4] Thank you to Alan Tickle for suggesting the importance of committee machines.

[5] This work was performed by Hanan Tayeb, Shahrazad Mohammed, Ghasaq Yousif and Shrouq Hasan as part of a final year undergraduate project, Department of Computer Science, American University of Sharjah, Spring 2007.

The objective of this introductory chapter is to outline a number of research issues, some of which are addressed in the following chapters. The ultimate goal, however, is to achieve what rule extraction from neural network undoubtedly has achieved, and this is to propose a set of techniques suitable for data mining and commercial applications. The following chapters include new algorithms for rule extraction as well as applications in a variety of domains. This includes financial applications as well as speech recognition. There is no doubt that the current research on developing new kernel methods to increase the accuracy of classification and regression must be complemented by a set of techniques that allow user explanation at a very high level.

8 Acknowledgements

Thank you to David Martens, Lisa Torrey, Shlomo Geva and Alan Tickle for comments on earlier versions of this introduction.

References

Andrews R, Diederich J, Tickle AB (1995) A Survey and Critique of Techniques For Extracting Rules From Trained Artificial Neural Networks, Knowledge Based Systems, 8, pp. 373–389

Barakat N, Diederich J (2005) Eclectic rule extraction from support vector machines. International Journal of Computational Intelligence, 2(1), 59–62, 2005

Berkhin P (2002) Survey of Clustering Data Mining Techniques. Accrue Software, San Jose, California

Cawsey A (1993) Explanation and Interaction. The Computer Generation of Explanatory Dialogues. The MIT Press Cambridge London

Chiu T, Fang D, Chen J, Wang Y, Jeris C (2001) A Robust and Scalable Clustering Algorithm for Mixed Type Attributes in Large Database Environment. Proceedings of the Seventh ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 263

Craven M, Shavlik J (1994) Using sampling and queries to extract rules from trained neural networks. In Proceedings of the 11th International Conference on Machine Learning, 1994, 37–45

Davis R, Buchanan BG, Shortliffe E (1977) Production rules as a representation for a knowledge-based consultation program'. Artificial Intelligence 8:1 15–45

Gallant S (1988) Connectionist expert systems. Communications of the ACM. 13:2 152–169

Gilbert N (1989) Explanation and dialogue. The Knowledge Engineering Review. 4:3 235–247

Golea, M (1996) “On the complexity of rule extraction from neural networks and network querying”. Proceedings of the Rule Extraction from Trained Artificial Neural Networks Workshop, Society for the Study of Artificial Intelligence and Simulation of Behavior Workshop Series (AISB’96) University of Sussex, Brighton, UK, 51–59

Hayward R, Nayak R, Diederich J (2000) Using Predicates to Explain Networks. In: ECAI-2000 Workshop: “Foundations of Connectionist-Symbolic Integration: Representation, Paradigms and Algorithms. Berlin, Germany

Joachims T (1998) Text Categorization with Support Vector Machines: Learning With Many Relevant Features. In: ECML-98, 10th European Conference on Machine Learning, Heidelberg, Germany, 137–142

Moore, JD, Swartout WR (1989) A Reactive Approach to Explanation. In: IJCAI-89 International Joint Conference on Artificial Intelligence, 1504–1510

Núñez H, Angulo C, Catala A (2002) Rule extraction from support vector machines. In: ECAI-92 Proceedings of European Symposium on Artificial Neural Networks, 107–112

Taskar B, Chatalbashev V, Koller D, Guestrin C (2005) Learning structured prediction models: A large margin approach. In: ICML 2005, Proceedings of the 22$^{\text{nd}}$ International Conference on Machine Learning

Thagard PR (1978) The best Explanation: Criteria for Theory Choice. The Journal of Philsosophy 75:2, 76–92

Thagard P, Litt A (forthcoming). Models of scientific explanation. In R. Sun (ed.), The Cambridge handbook of computational cognitive modeling. Cambridge: Cambridge University Press

Tickle A, Andrews R, Golea M, Diederich J (1998) The truth will come to light: directions and challenges in extracting the knowledge embedded within trained artificial neural network. IEEE Transactions on Neural Networks 9:6, 1057–1068

Witten IH, Frank E (1999) Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations. Paperback 162–164

Rule Extraction from Support Vector Machines: An Overview of Issues and Application in Credit Scoring

David Martens[1], Johan Huysmans[1], Rudy Setiono[2], Jan Vanthienen[1], and Bart Baesens[3,1]

[1] Department of Decision Sciences and Information Management, K.U.Leuven Naamsestraat 69, B-3000 Leuven, Belgium {David.Martens;Johan.Huysmans; Bart.Baesens;Jan.Vanthienen}@econ.kuleuven.be

[2] School of Computing, National University of Singapore, 3 Science Drive 2, Singapore 117543, Singapore rudys@comp.nus.edu.sg

[3] University of Southampton, School of Management, Highfield Southampton, SO17 1BJ, UK Bart@soton.ac.uk

Summary. Innovative storage technology and the rising popularity of the Internet have generated an ever-growing amount of data. In this vast amount of data much valuable knowledge is available, yet it is hidden. The Support Vector Machine (SVM) is a state-of-the-art classification technique that generally provides accurate models, as it is able to capture non-linearities in the data. However, this strength is also its main weakness, as the generated non-linear models are typically regarded as incomprehensible black-box models. By extracting rules that mimic the black box as closely as possible, we can provide some insight into the logics of the SVM model. This explanation capability is of crucial importance in any domain where the model needs to be validated before being implemented, such as in credit scoring (loan default prediction) and medical diagnosis. If the SVM is regarded as the current state-of-the-art, SVM rule extraction can be the state-of-the-art of the (near) future. This chapter provides an overview of recently proposed SVM rule extraction techniques, complemented with the pedagogical Artificial Neural Network (ANN) rule extraction techniques which are also suitable for SVMs. Issues related to this topic are the different rule outputs and corresponding rule expressiveness; the focus on high dimensional data as SVM models typically perform well on such data; and the requirement that the extracted rules are in line with existing domain knowledge. These issues are explained and further illustrated with a credit scoring case, where we extract a Trepan tree and a RIPPER rule set from the generated SVM model. The benefit of decision tables in a rule extraction context is also demonstrated. Finally, some interesting alternatives for SVM rule extraction are listed.

1 Introduction

Over the past decades we have witnessed a true explosion of data, which has mainly been driven by an ever growing popularity of the Internet and continuous innovations in storage technology. Information management and storage company EMC has recently calculated that 161 billion GigaByte of data has been created, with an expected 988 billion GigaByte to be created in 2010 [23]. Being able to find useful knowledge in this tremendous amount of data is humanly no longer possible, and requires advanced statistical and data mining techniques.

The Support Vector Machine (SVM) is currently the state-of-the-art in classification techniques. Benchmarking studies reveal that in general, the SVM performs best among current classification techniques [4], due to its ability to capture non-linearities. However, its strength is also its main weakness, as the generated non-linear models are typically regarded as incomprehensible black-box models. The opaqueness of SVM models can be remedied through the use of rule extraction techniques, which induce rules that mimic the black-box SVM model as closely as possible. If the SVM is regarded as the current state-of-the-art, SVM rule extraction can be the state-of-the-art of the (near) future.

This chapter is structured as follows. Before elaborating on the rationale behind SVM rule extraction (Sect. 3) as well as some of the issues (Sect. 5) and techniques (Sect. 4), an obligatory introduction to SVMs follows in the next section. We will illustrate these principles with an application in the financial domain, namely credit scoring, in Sect. 6, and finally discuss some possible alternatives for SVM rule extraction in Sect. 7.

2 The Support Vector Machine

Given a training set of N data points $\{(\mathbf{x}_i, y_i)\}_{i=1}^N$ with input data $\mathbf{x}_i \in \mathrm{I\!R}^n$ and corresponding binary class labels $y_i \in \{-1, +1\}$, the SVM classifier, according to Vapnik's original formulation satisfies the following conditions [20, 64]:

$$\begin{cases} \mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) + b \geq +1, & \text{if } \ y_i = +1 \\ \mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) + b \leq -1, & \text{if } \ y_i = -1 \end{cases} \tag{1}$$

which is equivalent to

$$y_i[\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}_i) + b] \geq 1, \qquad i = 1, \ldots, N. \tag{2}$$

The non-linear function $\boldsymbol{\varphi}(\cdot)$ maps the input space to a high (possibly infinite) dimensional feature space. In this feature space, the above inequalities basically construct a hyperplane $\mathbf{w}^T\boldsymbol{\varphi}(\mathbf{x}) + b = 0$ discriminating between the two classes. By minimizing $\mathbf{w}^T\mathbf{w}$, the margin between both classes is maximized (Fig. 1).

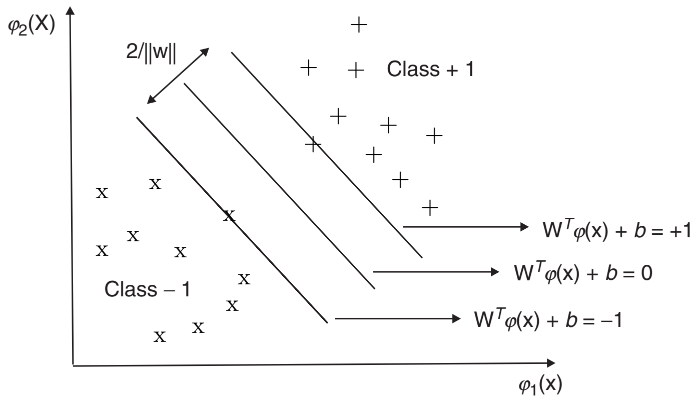


Fig. 1. Illustration of SVM optimization of the margin in the feature space

In primal weight space the classifier then takes the form

$$y(\mathbf{x}) = \text{sign}[\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}) + b], \tag{3}$$

but, on the other hand, is never evaluated in this form. One defines the convex optimization problem:

$$\min_{\mathbf{w},b,\boldsymbol{\xi}} \mathcal{J}(\mathbf{w}, b, \boldsymbol{\xi}) = \tfrac{1}{2}\mathbf{w}^T\mathbf{w} + C \textstyle\sum_{i=1}^N \xi_i \tag{4}$$

subject to

$$\begin{cases} y_i[\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_i) + b] \geq 1 - \xi_i, & i = 1, \ldots, N \\ \xi_i \geq 0, & i = 1, \ldots, N. \end{cases} \tag{5}$$

The variables ξ_i are slack variables which are needed in order to allow misclassifications in the set of inequalities (e.g. due to overlapping distributions). The first part of the objective function tries to maximize the margin between both classes in the feature space, whereas the second part minimizes the misclassification error. The positive real constant C should be considered as a tuning parameter in the algorithm.

The Lagrangian to the constraint optimization problem (4) and (5) is given by

$$\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}; \boldsymbol{\alpha}, \boldsymbol{\nu}) = \mathcal{J}(\mathbf{w}, b, \boldsymbol{\xi}) - \textstyle\sum_{i=1}^N \alpha_i \{y_i[\mathbf{w}^T \boldsymbol{\varphi}(\mathbf{x}_i) + b] - 1 + \xi_i\} - \textstyle\sum_{i=1}^N \nu_i \xi_i \tag{6}$$

The solution to the optimization problem is given by the saddle point of the Lagrangian, i.e. by minimizing $\mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}; \boldsymbol{\alpha}, \boldsymbol{\nu})$ with respect to $\mathbf{w}$, b, $\boldsymbol{\xi}$ and maximizing it with respect to $\boldsymbol{\alpha}$ and $\boldsymbol{\nu}$.

$$\max_{\boldsymbol{\alpha},\boldsymbol{\nu}} \min_{\mathbf{w},b,\boldsymbol{\xi}} \mathcal{L}(\mathbf{w}, b, \boldsymbol{\xi}; \boldsymbol{\alpha}, \boldsymbol{\nu}). \tag{7}$$

This leads to the following classifier:

$$y(\mathbf{x}) = \text{sign}[\textstyle\sum_{i=1}^N \alpha_i\, y_i\, K(\mathbf{x}_i, \mathbf{x}) + b], \tag{8}$$

whereby $K(\mathbf{x}_i, \mathbf{x}) = \boldsymbol{\varphi}(\mathbf{x}_i)^T \boldsymbol{\varphi}(\mathbf{x})$ is taken with a positive definite kernel satisfying the Mercer theorem. The Lagrange multipliers α_i are then determined by means of the following optimization problem (dual problem):

$$\max_{\alpha_i} -\frac{1}{2} \sum_{i,j=1}^{N} y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \alpha_i \alpha_j + \sum_{i=1}^{N} \alpha_i \tag{9}$$

subject to

$$\begin{cases} \sum_{i=1}^{N} \alpha_i y_i = 0 \\ 0 \leq \alpha_i \leq C, \quad i = 1, ..., N. \end{cases} \tag{10}$$

The entire classifier construction problem now simplifies to a convex quadratic programming (QP) problem in α_i. Note that one does not have to calculate $\mathbf{w}$ nor $\boldsymbol{\varphi}(\mathbf{x}_i)$ in order to determine the decision surface. Thus, no explicit construction of the non-linear mapping $\boldsymbol{\varphi}(\mathbf{x})$ is needed. Instead, the kernel function K will be used. For the kernel function $K(\cdot,\cdot)$, one typically has the following choices:

$K(\mathbf{x}, \mathbf{x}_i) = \mathbf{x}_i^T \mathbf{x}$,	(linear kernel)
$K(\mathbf{x}, \mathbf{x}_i) = (1 + \mathbf{x}_i^T \mathbf{x}/c)^d$,	(polynomial kernel of degree d)
$K(\mathbf{x}, \mathbf{x}_i) = \exp\{-\|\mathbf{x} - \mathbf{x}_i\|_2^2/\sigma^2\}$,	(RBF kernel)
$K(\mathbf{x}, \mathbf{x}_i) = \tanh(\kappa\, \mathbf{x}_i^T \mathbf{x} + \theta)$,	(MLP kernel),

where d, c, σ, κ and θ are constants.

For low-noise problems, many of the α_i will be typically equal to zero (sparseness property). The training observations corresponding to non-zero α_i are called support vectors and are located close to the decision boundary. This observation will be illustrated with Ripley's synthetic data in Sect. 5.

As (8) shows, the SVM classifier is a complex, non-linear function. Trying to comprehend the logics of the classifications made is quite difficult, if not impossible.

3 The Rationale Behind SVM Rule Extraction

SVM rule extraction is a natural variant of the well researched ANN rule extraction domain. To understand the usefulness of SVM rule extraction we need to discuss (1) why rule extraction is performed, and (2) why SVM rule extraction is performed rather than the more researched ANN rule extraction.

3.1 Why Rule Extraction

Rule extraction is performed for the following two reasons: (1) to understand the classifications made by the underlying non-linear black-box model,[1] thus

[1] As this can be an ANN, SVM or any other non-linear model, we will refer to it as the black box model.

to *open up the black box*; and (2) to *improve the performance of rule induction techniques* by removing idiosyncrasies in the data.

1. The most common motivation for using rule extraction is to obtain a set of rules that can explain the black box model. By obtaining a set of rules that mimic the predictions of the SVM, some insight is gained into the logical workings of the SVM. The extent to which the set of rules is consistent with the SVM is measured by the fidelity, and provides the percentage of test instances on which the SVM and the rule set concur with regard to the class label. If the rules and fidelity are satisfactory, the user might decide the SVM model has been sufficiently explained and use the SVM as decision support model.
2. An interesting observation is that the (generally) better performing non-linear model can be used in a pre-processing step to clean up the data [35,42]. By changing the class labels of the data by the class label of the black box, all noise is removed from the data. This can be seen from Fig. 2, which shows the synthetic Ripley's data set. Ripley's data set has two variables and thus allows for visualization of the model. The data set has binary classes, where the classes are drawn from two normal distributions with a high degree of overlap [51]. In Fig. 2a the original test data is shown, where one needs to discriminate between the blue dots and the red crosses. As can be seen, there is indeed much noise (overlap) in the data. The decision boundary of the induced SVM model, which has an accuracy of 90%, as well as the original test data are shown in Fig. 2b. If we change the class labels of the data to the class labels as predicted by the SVM model, that is all data instances above the decision boundary become blue dots, all below become red crosses, we obtain Fig. 2c. As this figure illustrates no more noise or conflict is present in the data. Finally, Fig. 2d shows that the SVM model can be used to provide class labels to artificially generated data, thereby circumventing the problem of having only few data instances. A rule extraction technique that makes advantage of this approach is Trepan, discussed in the next section. In our previous work, we have shown that performing rule induction techniques on these new SVM predicted data set can increase the performance of traditional rule induction techniques [42].

3.2 Why SVM Rule Extraction

Rule extraction from ANNs has been well researched, resulting in a wide range of different techniques (a full overview can be found in [29], an application of ANN rule extraction in credit scoring is given in [3]). The SVM is, as the ANN, a non-linear predictive data mining technique. Benchmarking studies have shown that such models exhibit good and comparable generalization behavior (out-of-sample accuracy) [4,63]. However, SVMs have some important benefits over ANNs. First of all, ANNs suffer from local minima in

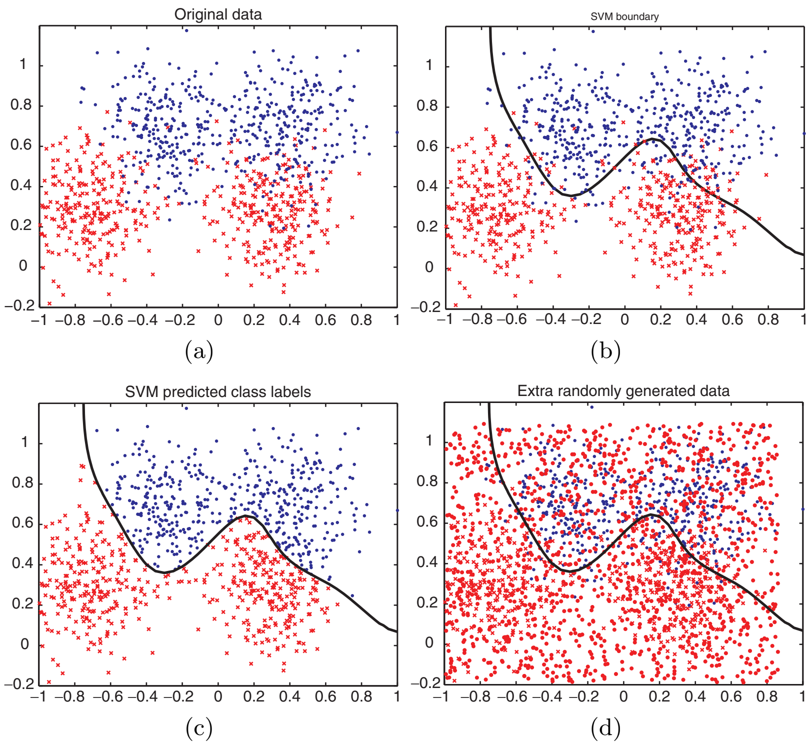


Fig. 2. (**a**) Ripley's synthetic data set, with SVM decision boundary (**b**). In (**c**) the class labels have been changed to the SVM predicted class labels, thereby removing present noise. Artificial data examples can be generated with their class labels assigned by the SVM model, as shown by the 1,500 extra generated instances in (**d**)

the weight solution space [8]. Secondly, several architectural choices (such as number of hidden layers, number of hidden nodes, activation function, etc.) need to be determined (although we need to remark that for SVMs the regularization parameter C and bandwidth σ for an RBF kernel, also need to be set. These are typically set using a gridsearch procedure [63]). Extracting rules from this state-of-the-art classification technique is the natural next step.

4 An Overview of SVM Rule Extraction Techniques

4.1 Classification Scheme for SVM Rule Extraction Techniques

Andrews et al. [2] propose a classification scheme for neural network rule extraction techniques that can easily be extended to SVMs, and is based on the following criteria:

1. Translucency of the extraction algorithm with respect to the underlying neural network;

2. Expressive power of the extracted rules or trees;
3. Specialized training regime of the neural network;
4. Quality of the extracted rules;
5. Algorithmic complexity of the extraction algorithm.

As for SVM rule extraction the training regime is not as much an issue as for ANNs, and the algorithmic complexity of a rule extraction algorithm is hard to assess, we will only elaborate on the translucency, the expressive power of the rules, and the quality of the rules as part of the rule extraction technique evaluation.

Translucency

The translucency criterion considers the technique's perception of the SVM. A decompositional approach is closely intertwined with the internal workings of the SVM and its constructed hyperplane. On the other hand, a pedagogical algorithm considers the trained model as a black box. Instead of looking at the internal structure, these algorithms directly extract rules which relate the inputs and outputs of the SVM. These techniques typically use the trained SVM model as an oracle to label or classify artificially generated training examples which are later used by a symbolic learning algorithm, as already illustrated in Fig. 2d. The idea behind these techniques is the assumption that the trained model can better represent the data than the original data set. That is, the data is cleaner, free of apparent conflicts. The difference between decompositional and pedagogical rule extraction techniques is schematically illustrated in Fig. 3. Since the model is viewed as a black box, most pedagogical algorithms lend themselves very easily to rule extraction from other machine learning algorithms. This allows us to extrapolate rule extraction techniques from the neural networks domain to our domain of interest, SVMs.

Expressive Power

The expressive power of the extracted rules depends on the language used to express the rules. Many types of rules have been suggested in the literature. Propositional rules are simple **If**... **Then**... expressions based on conventional propositional logic.

The second rule type we will encounter are M-of-N rules and are usually expressed as follows:

$$\begin{array}{l}\textbf{If}\ \{\text{at least/exactly/at most}\}\ \text{M of the N conditions}\,(\text{C1}, \text{C2}, \ldots, \text{CN})\\ \text{are satisfied}\ \textbf{Then}\ \text{Class} = 1.\end{array} \tag{11}$$

This type of rules allows one to represent complex classification concepts more succinctly than classical propositional DNF rules.

The rule types considered above are crisp in the sense that their antecedent is either true or false. Fuzzy rules allow for more flexibility and are usually

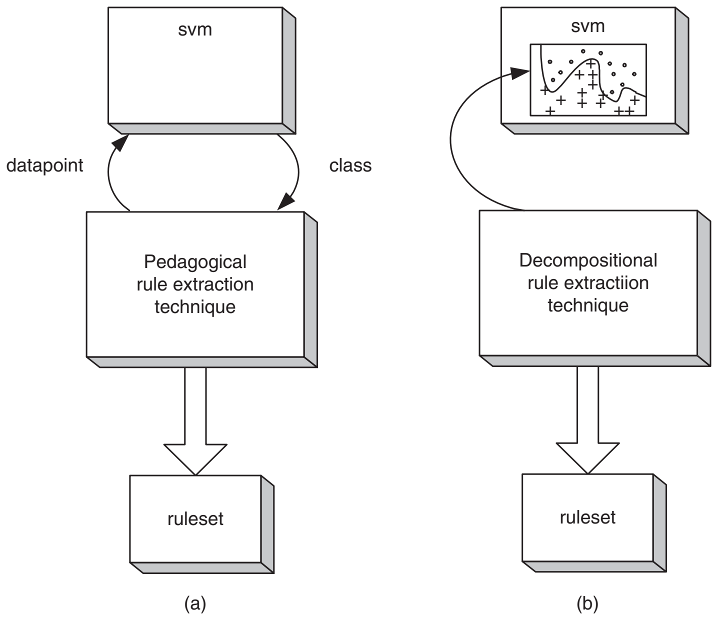

svm
datapoint
class
Pedagogical rule extraction technique
ruleset
(a)
svm
Decompositional rule extractiion technique
ruleset
(b)

Fig. 3. Pedagogical (**a**) and decompositional (**b**) rule extraction techniques

expressed in terms of linguistic concepts which are easier to interpret for humans.

Rule Extraction Technique Evaluation

In order to evaluate the rule extraction algorithms, Craven and Shavlik [18] listed five performance criteria:

1. Comprehensibility: The extent to which extracted representations are humanly comprehensible.
2. Fidelity: The extent to which the extracted representations model the black box from which they were extracted.
3. Accuracy: The ability of extracted representations to make accurate predictions on previously unseen cases.
4. Scalability: The ability of the method to scale to other models with large input spaces and large number of data.
5. Generality: The extent to which the method requires special training regimes or restrictions on the model architecture.

The latter two performance measures are often forgotten and omitted, since it is difficult to quantify them. In the context of SVM rule extraction mainly scalability becomes an important aspect, as SVMs perform well on large dimensional data. Craven and Shavlik additionally consider software availability as key to the success of rule extraction techniques.

4.2 SVM Rule Extraction Techniques

Table 1 provides an overview of SVM rule extraction techniques, and describes the translucency and rule expressiveness.[2] A chronological overview of all discussed algorithms (and some additional techniques that were not discussed in the text) is given below in Table 1. For each algorithm, we provide the following information:

Translucency (P or D): **P**edagogical or **D**ecompositional
Scope (C or R): **C**lassification or **R**egression
Summary: A very short description of the algorithm

The first set of techniques are specifically intended as SVM rule extraction techniques. Thereafter, we list some commonly used rule induction techniques that can be used as pedagogical rule extraction techniques (by changing the class to the SVM predicted class), and pedagogical ANN rule extraction techniques that can easily be used as SVM rule extraction technique. Notice that the use of such pedagogical techniques have only rarely been applied as SVM rule extraction techniques.

What follows is a short description of the proposed decompositional SVM rule extraction techniques, and some of the most commonly used rule

Table 1. Chronological overview of rule extraction algorithms

Algorithm (Year)	Ref.	Transl.	Scope	Summary
		SVM Rule extraction techniques		
SVM + Prototypes (2002)	[46]	D	C	Clustering
Barakat (2005)	[6]	D	C	Train decision tree on support vectors and their class labels
Fung (2005)	[25]	D	C	Only applicable to linear classifiers
Iter (2006)	[28]	P	C + R	Iterative growing of hypercubes
Minerva (2007)	[30]	P	C + R	Sequential covering + iterative growing
Rule induction techniques, and Pedagogical ANN rule extraction techniques, also applicable to SVMs				
CART (1984)	[11]	P	C + R	Decision tree induction
CN2 (1989)	[15]	P	C	Rule induction
C4.5 (1993)	[49]	P	C	Decision tree induction
TREPAN (1996)	[18]	P	C	Decision tree induction, M-of-N splits
BIO-RE (1999)	[56]	P	C	Creates complete truth table, only applicable to toy problems
ANN-DT (1999)	[52]	P	C + R	Decision tree induction, similar to TREPAN
DecText (2000)	[10]	P	C	Decision tree induction
STARE (2003)	[68]	P	C	Breadth-first search with sampling, prefers categorical variables over continuous variables
G-REX (2003)	[34]	P	C + R	Genetic programming: different types of rules
REX (2003)	[41]	P	C	Genetic algorithm: fuzzy rules
GEX (2004)	[40]	P	C	Genetic algorithm: propositional rules
Rabuñal (2004)	[50]	P	C	Genetic programming
BUR (2004)	[14]	P	C	Based on gradient boosting machines
Re-RX (2006)	[53]	P	C	Hierarchical rule sets: first splits are based on discrete attributes
AntMiner + (2007)	[44]	P	C	Ant-based induction of rules

[2] Partially based upon artificial neural network classification scheme by Andrews, Diederich and Tickle [2].

induction and pedagogical rule extraction techniques, which were originally proposed in the context of neural networks.

SVM+Prototypes

One of the few rule extraction methods designed specifically for support vector machines is the SVM+Prototypes method proposed in [46]. This decompositional algorithm is not only able to extract propositional (interval) classification rules from a trained SVM, but also rules from which the conditions are mathematical equations of ellipsoids. We will discuss the variant that results in propositional rules.

The SVM+Prototypes algorithm is an iterative process that proceeds as follows:

Step 1 **Train a Support Vector Machine** The SVM's decision boundary will divide the training data in two subsets $\mathcal{S}^+$ and $\mathcal{S}^-$, containing the instances for which the predicted class is respectively positive and negative. Initialize the variable i to 1.

Step 2 For each subset, use some clustering algorithm to find i clusters (new subsets) and calculate the prototype or centroid of each cluster. For each of these new subsets find the support vector that lies farthest to the prototype. Use the prototype as center and the support vector as vertex to create a hypercube in the input space.

Step 3 Do a partition test on each of the hypercubes. This partition test is performed to minimize the level of overlapping between cubes for which the predicted class is different. One possible method is to test whether all of the corners of the hypercube are predicted to be of the same class. If this is the case then we say that the partition test is positive.

Step 4 Convert the hypercubes with a negative partition test into rules. If there are hypercubes with a positive partition test and i is smaller than a user-specified threshold I_{max} then increase i, take the subsets from which these cubes were created and go back to step 2, else go to step 5.

Step 5 If i is equal to I_{max}, convert all of the current hypercubes into rules.

The example of Fig. 4 shows the principal idea behind the algorithm. During the first iteration ($i = 1$), the algorithm looks for the centroid of respectively the black and white instances. These prototypes are indicated by a star sign. It will then search the support vector in that partition that lies farthest away from the prototype and will create a cube from these two points (Step Two). In step three, the partition test will be positive for the leftmost cube as one of its vertices lies in the area for which the SVM predicts a different class. The other cube has a negative partition test and will therefore become a rule in step 4. For the first cube with a positive partition test, we will iterate the above procedure but with $i = 2$. This will result in the creation of two new rules.

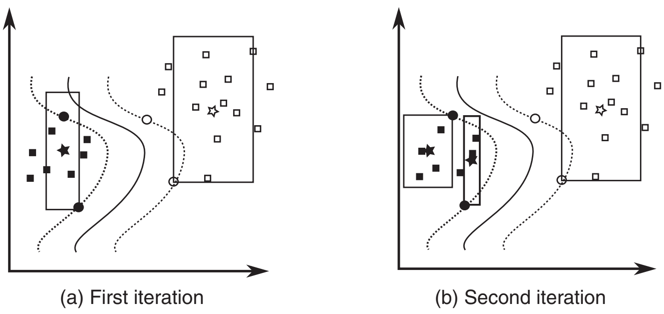


Fig. 4. Example of SVM+Prototypes algorithm

The main drawback of this algorithm is that the extracted rules are neither exclusive nor exhaustive which results in conflicting or missing rules for the classification of new data instances. Each of the extracted rules will also contain all possible input variables in its conditions, making the approach undesirable for larger input spaces as it will extract complex rules that lack interpretability. In [6], another issue with the scalability of this method is observed: a higher number of input patterns will result in more rules being extracted, which further reduces comprehensibility.

An interesting approach for this technique to avoid the time-consuming clustering might be the use of Relevance Vector Machines [58, 59]. This technique is introduced by Tipping in 2000, and similar to the SVM but based on Bayesian learning. As he mentions *unlike for the SVM, the relevance vectors are some distance from the decision boundary (in x-space), appearing more "prototypical" or even "anti-boundary" in character.* In this manner, prototypes are immediately formed and could be used in the rule extraction technique.

Fung et al.

In [25], Fung et al. present an algorithm to extract propositional classification rules from linear classifiers. The method is considered to be decompositional because it is only applicable when the underlying model provides a linear decision boundary. The resulting rules are parallel with the axes and non-overlapping, but only (asymptotically) exhaustive. Completeness can however, be ensured by retrieving rules for only one of both classes and specification of a default class.

The algorithm is iterative and extracts the rules by solving a constrained optimization problem that is computationally inexpensive to solve. While the mathematical details are relatively complex and can be found in [25], the principal idea is rather straightforward to explain. Figure 5 shows execution of the algorithm when there are two inputs and when only rules for the black squares are being extracted.

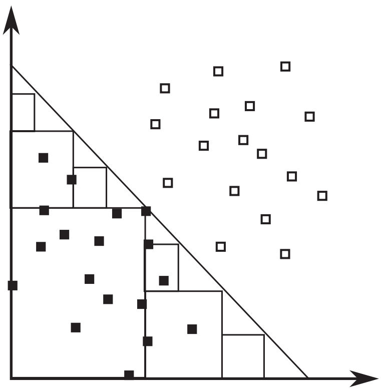


Fig. 5. Example of algorithm of Fung et al.

First, a transformation is performed such that all inputs of the black squares observations are in the interval [0,1]. Then the algorithm searches for an (hyper)cube that has one vertex on the separating hyperplane and lies completely in the region below the separating hyperplane. There are many cubes that satisfy these criteria, and therefore the authors added a criterion to find the "optimal" cube. They developed two variants of the algorithm that differ only in the way this optimality is defined: volume maximization and point coverage maximization. In the example of Fig. 5, this "optimal" cube is the large cube that has the origin as one of its vertices. This cube divides the region below the separating hyperplane in two new regions: the regions above and to the right of the cube. In general for an N-dimensional input space, one rule will create N new regions. In the next iteration, a new "optimal" cube is recursively retrieved for each of the new regions that contain training observations. The algorithm stops after a user-determined maximum number of iterations.

The proposed method faces some drawbacks. Similar to the SVM + Prototypes method discussed above, each rule condition involves all the input variables. This makes the method unsuitable for problems with a high-dimensional input space. A second limitation is the restriction to linear classifiers. This requirement considerably reduces the possible application domains.

Rule and Decision Tree Induction Techniques

Many algorithms are capable of learning rules or trees directly from a set of training examples, e.g., CN2 [15], AQ [45], RIPPER [16], AntMiner+ [44], C4.5 [49] or CART [11]. Because of their ability to learn predictive models directly from the data, these algorithms are not considered to be rule extraction techniques in the strict sense of the word. However, these algorithms can also be used to extract a human-comprehensible description from opaque models. When used for this purpose, the original target values of the training examples are modified by the predictions made by the black box model and the algorithm is then applied to this modified data set. Additionally, to ensure that the white box learner will mimic the decision boundary of the black box model even more, one can also create a large number of artificial examples and then ask the black box model to provide the class labels for these sampled points. The remainder of this section briefly covers both approaches, as they form the basic for most pedagogical rule extraction techniques.

Rule Induction Techniques

In this section, we discuss a general class of rule induction techniques: sequential covering algorithms. This series of algorithms extracts a rule set by learning one rule, removing the data points covered by that rule and reiterating the algorithm on the remainder of the data. RIPPER, Iter and Minerva are some of the techniques based on this general working.

Starting from an empty rule set, the sequential covering algorithm first looks for a rule that is highly accurate for predicting a certain class. If the accuracy of this rule is above a user-specified threshold, then the rule is added to the set of existing rules and the algorithm is repeated over the rest of the examples that were not classified correctly by this rule. If the accuracy of the rule is below this threshold the algorithm will terminate. Because the rules in the rule set can be overlapping, the rules are first sorted according to their accuracy on the training examples before they are returned to the user. New examples are classified by the prediction of the first rule that is triggered.

It is clear that in the above algorithm, the subroutine of learning one rule is of crucial importance. The rules returned by the routine must have a good accuracy but do not necessarily have to cover a large part of the input space. The exact implementation of this learning of one rule will be different for each algorithm but usually follows either a bottom-up or top-down search process. If the bottom-up approach is followed, the routine will start from a very specific rule and drop in each iteration the attribute that least influences the accuracy of the rule on the set of examples. Because each dropped condition makes the rule more general, the search process is also called specific-to-general search. The opposite approach is the top-down or general-to-specific search: the search starts from the most general hypothesis and adds in each iteration the attribute that most improves accuracy of the rule on the set of examples.

Decision Trees: C4.5 and CART

Decision trees [11, 36, 49] are widely used in predictive modeling. A decision tree is a recursive structure that contains a combination of internal and leaf nodes. Each internal node specifies a test to be carried out on a single variable and its branches indicate the possible outcomes of the test. An observation can be classified by following the path from the root towards a leaf node.

At each internal node, the corresponding test is performed and the outcome indicates the branch to follow. With each leaf node, a value or class label is associated.

In the rest of this section, we discuss briefly the most widespread algorithm for decision tree induction, namely C4.5. It uses a divide-and-conquer approach to construct a suitable tree from a set of training examples [48].

C4.5 induces decision trees based on information theoretic concepts. Let p_1 (p_0) be the proportion of examples of class 1(0) in sample S. The entropy of S is then calculated as follows:

$$\text{Entropy}(S) = -p_1 \log_2(p_1) - p_0 \log_2(p_0), \tag{12}$$

whereby $p_0 = 1 - p_1$. Entropy is used to measure how informative an attribute is in splitting the data. Basically, the entropy measures the order (or disorder) in the data with respect to the classes. It equals 1 when $p_1 = p_0 = 0.5$ (maximal disorder, minimal order) and 0 (maximal order, minimal disorder) when $p_1 = 0$ or $p_0 = 0$. In the latter case, all observations belong to the same class. Gain(S, x_j) is defined as the expected reduction in entropy due to sorting (splitting) on attribute x_j:

$$\text{Gain}(S, x_j) = \text{Entropy}(S) - \sum_{v \,\in\, \text{values}(x_j)} \frac{|S_v|}{|S|} \text{Entropy}(S_v), \tag{13}$$

where values(x_j) represents the set of all possible values of attribute x_j, S_v the subset of S where attribute x_j has value v and $|S_v|$ the number of observations in S_v. The Gain criterion was used in ID3, the forerunner of C4.5, to decide upon which attribute to split at a given node [48]. However, when this criterion is used to decide upon the node splits, the algorithm favors splits on attributes with many distinct values. In order to rectify this, C4.5 applies a normalization and uses the gainratio criterion which is defined as follows:

$$\begin{aligned} \text{Gainratio}(S, x_j) &= \frac{\text{Gain}(S, x_j)}{\text{SplitInformation}(S, x_j)} \quad \text{with} \\ \text{SplitInformation}(S, x_j) &= - \sum_{k \,\in\, \text{values}(\mathrm{x_j})} \frac{|S_k|}{|S|} \log_2 \frac{|S_k|}{|S|}. \end{aligned} \tag{14}$$

Another very popular tree induction algorithm is **CART**, short for Classification and Regression Trees [11]. It is largely similar to C4.5, but with a different splitting criterion (Gini Index) and pruning procedure. Additionally, CART can also be applied to regression problems.

The tree induction algorithm C4.5 is applied to the data where the output has been changed to the SVM predicted value, so that the tree approximates the SVM. Since the trees can be converted into rules, we can regard this technique as a rule extraction technique. This approach has been used in [5] to extract rules from SVMs. A slightly different variant is proposed in [6], where only the support vectors are used. A problem that arises with such decision tree learners however, is that the deeper a tree is expanded, the fewer data points are available to use to decide upon the splits. The next technique we will discuss tries to overcome this issue.

Trepan

Trepan [17, 18] is a popular pedagogical rule extraction algorithm. While it is limited to binary classification problems, it is able to deal with both continuous and nominal input variables. Trepan shows many similarities with the more conventional decision-tree algorithms that learn directly from the training observations, but differs in a number of respects.

First, when constructing conventional decision trees, a decreasing number of training observations is available to expand nodes deeper down the tree. Trepan overcomes this limitation by generating additional instances. More specifically, Trepan ensures that at least a certain minimum number of observations are considered before assigning a class label or selecting the best split. If fewer instances are available at a particular node, additional instances will be generated until this user-specified threshold is met. The artificial instances must satisfy the constraints associated with each node and are generated by taking into account each feature's marginal distribution. So, instead of taking uniform samples from (part of) the input space, Trepan first models the marginal distributions and subsequently creates instances according to these distributions while at the same time ensuring that the constraints to reach the node are satisfied. For discrete attributes, the marginal distributions can easily be obtained from the empirical frequency distributions. For continuous attributes, Trepan uses a kernel density based estimation method [55] that calculates the marginal distribution for attribute x as:

$$f(x) = \frac{1}{m}\sum_{i=1}^{m} \frac{1}{\sqrt{2\pi}\sigma} e^{-(\frac{x-\mu_i}{2\sigma})^2} \qquad (15)$$

with m the number of training examples, μ_i the value for this attribute for example i and σ the width of the gaussian kernel. Trepan sets the value for σ to $1/\sqrt{m}$. One shortcoming of using the marginal distributions is that dependencies between variables are not taken into account. Trepan tries to overcome this limitation by estimating new models for each node and using only the training examples that reach that particular node. These locally estimated models are able to capture some of the conditional dependencies between the different features. The disadvantage of using local models is that they are based on less data, and might therefore become less reliable. Trepan handles this trade-off by performing a statistical test to decide whether or not a local model is used for a node. If the locally estimated distribution and the estimated distribution at the parent are significantly different, then Trepan uses the local distributions, otherwise it uses the distributions of the parent.

Second, most decision tree algorithms, e.g., CART [11] and C4.5 [49], use the internal (non-leaf) nodes to partition the input space based on one simple feature. Trepan on the other hand, uses M-of-N expressions in its splits that allow multiple features to appear in one split. Note that an M-of-N split is satisfied when M of the N conditions are satisfied. 2-of-{a,¬b,c} is therefore logically equivalent to $(a \wedge \neg b) \vee (a \wedge c) \vee (\neg b \wedge c)$. To avoid testing all of the possibly large number of M-of-N combinations, Trepan uses a heuristic beam search with a beam width of two to select its splits. The search process is initialized by first selecting the best binary split at a given node based on the information gain criteria ([17] (or gain ratio according to [18]). This split and its complement are then used as basis for the beam search procedure that is halted when the beam remains unchanged during an iteration. During each iteration, the following two operators are applied to the current splits:

- M-of-N+1: the threshold remains the same but a new literal is added to the current set. For example, 2-of-{a,b} is converted into 2-of-{a,b,c}
- M+1-of-N+1: the threshold is incremented by one and a new literal is added to the current set. For example, 2-of-{a,b} is converted into 3-of-{a,b,c}

Finally, while most algorithms grow decision trees in a depth-first manner, Trepan employs the best-first principle. Expansion of a node occurs first for those nodes that have the greatest potential to increase the fidelity of the tree to the network.

Previous rule extraction studies have shown the potential benefit in performance from using Trepan [4, 42], which can be mainly attributed to it's extra data generating capabilities.

Re-RX

The final promising pedagogical rule extraction technique that we will discuss is Re-RX.

As typical data contain both discrete and continuous attributes, it would be useful to have a rule set that separates the rule conditions involving these two types of attributes to increase its interpretability. Re-RX is a recursive algorithm that has been developed to generate such rules from a neural network classifier [53]. Being pedagogical in its approach, it can be easily applied for rule extraction from SVM.

The basic idea behind the algorithm is to try to split the input space first using only the relevant discrete attributes. When there is no more discrete attribute that can be used to partition the input space further, in each of these subspaces the final partition is achieved by a hyperplane involving only the continuous attributes. If we depict the generated rule set as a decision tree, we would have a binary tree where all the node splits are determined by the value of a single discrete attributes, except for the last split in each tree branch where the condition of the split is a linear combination of the continuous attributes. The outline of the algorithm is given below.

Input: A set of data samples $\mathcal{S}$ having the discrete attributes $\mathcal{D}$ and continuous attributes $\mathcal{C}$.
Output: A set of classification rules.

```
1. Train and prune a neural network using the data set $\mathcal{S}$ and all its attributes $\mathcal{D}$ and $\mathcal{C}$.
2. Let $\mathcal{D}'$ and $\mathcal{C}'$ be the sets of discrete and continuous attributes still present in the network, respectively. And let $\mathcal{S}'$ be the set of data samples that are correctly classified by the pruned network.
3. If $\mathcal{D}' = \emptyset$, then generate a hyperplane to split the samples in $\mathcal{S}'$ according to the values of their continuous attributes $\mathcal{C}'$ and stop.
   Otherwise using only the discrete attributes $\mathcal{D}'$, generate the set of classification rules $\mathcal{R}$ for the data set $\mathcal{S}'$.
4. For each rule $\mathcal{R}_i$ generated:
   If $support(\mathcal{R}_i) > \delta_1$ and $error(\mathcal{R}_i) > \delta_2$, then
   – Let $\mathcal{S}_i$ be the set of data samples that satisfy the condition of rule $\mathcal{R}_i$ and $\mathcal{D}_i$ be the set of discrete attributes that do not appear in rule condition of $\mathcal{R}_i$.
   – If $\mathcal{D}_i = \emptyset$, then generate a hyperplane to split the samples in $\mathcal{S}_i$ according to the values of their continuous attributes $\mathcal{C}_i$ and stop.
     Otherwise, call Re-RX($\mathcal{S}_i, \mathcal{D}_i, \mathcal{C}_i$).
```

Using samples that have been correctly classified by the pruned neural network, the algorithm either (1) groups these samples into one of the two possible classes by a single hyperplane if only continuous attributes are found relevant by the network, or (2) generates a set of classification rules using only the relevant discrete attributes. In latter case, the support and accuracy of each generated rule are computed. Those rules that are found not to be satisfactory according to predetermined criteria need to be refined. The refinement of a rule is achieved by simply executing the algorithm Re-RX again on all samples that satisfied the condition of this rule.

An example of a rule generated by Re-RX for credit scoring application is shown in Table 2.

5 Issues Concerning SVM Rule Extraction

5.1 Rule Output

As we have seen in Sect. 4.1 rule expressiveness is one of the categories for classifying rule extraction techniques. While for performance criteria accuracy and fidelity it is straightforward to rank the results (the higher the percentage of

Table 2. Example rule set from Re-RX

```
Rule $r$: **if** Years Client $< 5$ and Purpose $\neq$ Private loan
  Rule $r_1$: **if** Number of applicants $\geq 2$ and Owns real estate = yes, **then**
    Rule $r_{1a}$: **if** Savings amount + 1.11 Income - 38,249.74 Insurance - 0.46 Debt $>$ -19,39,300
    **then** applicant = good.
    Rule $r_{1b}$: **else** applicant = bad.
  Rule $r_2$: **else if** Number of applicants $\geq 2$ and Owns real estate = no, **then**
    Rule $r_{2a}$: **if** Savings amount + 1.11 Income - 38,249.74 Insurance - 0.46 Debt $>$ -16,38,720
    **then** applicant = good.
    Rule $r_{2b}$: **else** applicant = bad.
  Rule $r_3$: **else if** Number of applicants = 1 and Owns real estate = yes, **then**
    Rule $r_{3a}$: **if** Savings amount + 1.11 Income - 38,249.74 Insurance - 0.46 Debt $>$ -16,98,200
    **then** applicant = good.
    Rule $r_{3b}$: **else** applicant = bad.
  Rule $r_4$: **else if** Number of applicants = 1 and Owns real estate = no, **then**
    Rule $r_{4a}$: **if** Savings amount + 1.11 Income - 38,249.74 Insurance - 0.46 Debt $>$ -12,56,900
    **then** applicant = good.
    Rule $r_{4b}$: **else** applicant = bad.
```

correctly classified test instances, the better), this is not the case for comprehensibility. Although one might argue that fewer rules is better, the question arises how one can compare a propositional rule set with an M-of-N decision tree, an oblique rule set or a fuzzy rule set. A decision tree can be converted into a set of rules, where each leaf corresponds to one rule, but is a rule set with 4 rules really just as comprehensible as a tree with 4 leaves? Don't many variants of a tree with 4 leaves exist (completely balanced, unbalanced, binary, etc.)?

The comprehensibility of the chosen output and the ranking among the possible formats is a very difficult issue that has not yet been completely tackled by existing research. This is mainly due to the subjective nature of "comprehensibility", which is not just a property of the model but also depends on many other factors, such as the analyst's experience with the model and his/her prior knowledge. Despite this influence of the observer, some representation formats are generally considered to be more easily interpretable than others. In [32], an experiment was performed to compare the impact of several representation formats on the aspect of comprehensibility. The formats under consideration were decision tables, (binary) decision trees, a textual description of propositional rules and a textual description of oblique rules. In addition to a comparison between the different representation formats, the experiment also investigated the influence of the size or complexity of each of these representations on their interpretability.

It was concluded that decision tables provide significant advantages if comprehensibility is of crucial importance. The respondents of the experiment were able to answer a list of questions faster, more accurately and more confidently with decision tables than with any of the other representation formats. A majority of the users also found decision tables the easiest representation format to work with. For the relation between complexity and comprehensibility the results were less ideal: whatever the representation format, the number of correct answers of the respondents was much lower for the more complex models. For rule extraction research, this result implies that only small models should be extracted as the larger models are deemed too complex to be comprehensible. We would promote collaboration between the data mining and cognitive science communities to create algorithms and representations that are both effective as well as comprehensible to the end-users.

5.2 High Dimensional Data

SVMs are able to deal with high dimensional data through the use of the regularization parameter C. This advantage is most visible in high dimensional problem domains such as text mining [33] and in bioinformatics [13]. A case study on text mining has been put forward in the introductory chapter by Diederich.

Rule induction techniques on the other hand, have more problems with this curse of dimensionality [57]. At this moment, we are not aware of any SVM rule extraction algorithm that can flexibly deal with high dimensional data, for which it is known that SVMs are particularly suitable.

5.3 Constraint Based Learning: Knowledge Fusion Problem

Although many powerful classification algorithms have been developed, they generally rely solely on modeling repeated patterns or correlations which occur in the data. However, it may well occur that observations, that are very evident to classify by the domain expert, do not appear frequently enough in the data in order to be appropriately modeled by a data mining algorithm. Hence, the intervention and interpretation of the domain expert still remains crucial. A data mining approach that takes into account the knowledge representing the experience of domain experts is therefore much preferred and of great focus in current data mining research. A model that is in line with existing domain knowledge is said to be justifiable [43].

Whenever comprehensibility is required, justifiability is a requirement as well. Since the aim of SVM rule extraction techniques is to provide comprehensible models, this justifiability issue becomes of great importance. The academically challenging problem of consolidating the automatically generated data mining knowledge with the knowledge reflecting experts' domain expertise, constitutes the knowledge fusion problem (see Fig. 6). The final goal of the knowledge fusion problem is to provide models that are accurate, comprehensible and justifiable, and thus acceptable for implementation. The most frequently encountered and researched aspect of knowledge fusion is the monotonicity constraint. This constraint demands that an increase in a certain input(s) cannot lead to a decrease in the output. More formally (similarly to [24]), given a data set $D = \{x^i, y^i\}_{i=1}^n$, with $x^i = (x_1^i, x_2^i, \dots, x_m^i) \in X = X_1 \times X_2 \times \dots X_m$, and a partial ordering $\leq$ defined over this input space X.

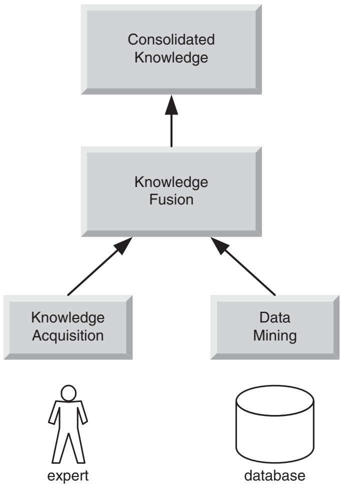


Fig. 6. The knowledge fusion process

Over the space Y of class values y^i, a linear ordering $\leq$ is defined. Then the classifier $f : x^i \mapsto f(x^i) \in Y$ is monotone if (16) holds.

$$x^i \leq x^j \Rightarrow f(x^i) \leq f(x^j),\ \forall i, j \quad (\text{or } f(x^i) \geq f(x^j), \forall i, j). \tag{16}$$

For instance, increasing income, keeping all other variables equal, should yield a decreasing probability of loan default. Therefore if client A has the same characteristics as client B, but a lower income, then it cannot be that client A is classified as a good customer and client B a bad one.

In linear mathematical models, generated by e.g., linear and logistic regression, the monotonicity constraint is fulfilled by demanding that the sign of the coefficient of each of the explanatory variables is the same as the expected sign for that variable. For instance, since the probability of loan default should be negatively correlated to the income, the coefficient of the income variable is expected to have a negative sign.

Several adaptions to existing classification techniques have been put forward to deal with monotonicity, such as for Bayesian learning [1], classification trees [7,21,24], classification rules [43] and neural networks [54,66]; e.g., in the medical diagnosis [47], house price prediction [65] and credit scoring [21, 54] domains.

Until now this justifiability constraint has not been addressed in the SVM rule extraction literature, although the application of rule induction techniques that do obtain this feature, such as AntMiner+ [43] and tree inducers proposed in [7, 24], as SVM rule extraction techniques is a first step into that direction.

5.4 Specificness of Underlying Black Box Model

Although decompositional techniques might better exploit the advantages of the underlying black box model, a danger exists that a too specific model is required. In ANN rule extraction almost all decompositional techniques require a certain architecture for the ANN, for example only one hidden node, or the need for product units.

As we've seen, the technique by Fung et al. also requires a special kind of SVM: a linear one. We believe it is important for a successful SVM rule extraction technique not to require a too specific SVM, such as for instance a LS-SVM, or a RVM. Although this is not yet a real issue with SVM rule extraction, this can certainly be observed in ANN rule extraction and should thus be kept in mind when developing new techniques.

5.5 Regression

From Table 1 it can be seen that only few rule extraction techniques focus on the regression task. Still, there is only little reason for exploring the use of rule extraction for classification only, as the SVM is just as successful for regression tasks. The same comprehensibility issues are important for regression, thereby providing the same motivation for rule extraction.

5.6 Availability of Code

A final issue in rule extraction research is the lack of executable code for most of the algorithms. In [19], it was already expressed that availability of software is of crucial importance to achieve a wide impact of rule extraction. However, only few algorithms are publicly available. This makes it difficult to gain an objective view of the algorithms' performance or to benchmark multiple algorithms on a data set. Furthermore, we are convinced that it is not only useful to make the completed programs available, but also to provide code for the subroutines used within these programs as they can often be shared. For example, the creation of artificial observations in a constrained part of the input space is a routine that is used by several methods, e.g., Trepan, ANN-DT and Iter. Other routines that can benefit from sharing and that can facilitate development of new techniques are procedures to query the underlying model or routines to optimize the returned rule set.

6 Credit Scoring Application

6.1 Credit Scoring in Basel II

The introduction of the Basel II Capital Accord has encouraged financial institutions to build internal rating systems assessing the credit risk of their various credit portfolios. One of the key outputs of an internal rating system is the probability of default (PD), which reflects the likelihood that a counterparty will default on his/her financial obligation. Since the PD modeling problem basically boils down to a discrimination problem (defaulter or not), one may rely on the myriad of classification techniques that have been suggested in the literature. However, since the credit risk models will be subject to supervisory review and evaluation, they must be easy to understand and transparent. Hence, techniques such as neural networks or support vector machines are less suitable due to their black box nature, while rules extracted from these non-linear models are indeed appropriate.

6.2 Classification Model

We have applied two rule extraction techniques with varying properties to the German credit scoring data set, publicly available from the UCI data repository [26]. The provided models illustrate some of the issues, mentioned before, such as the need to incorporate domain knowledge, the different comprehensibility accompanied by different rule outputs, and the benefits of decision tables.

First, a Trepan tree is provided in Fig. 7, while Table 3 provides the rules extracted by RIPPER on the data set with class labels predicted by the SVM. The attentive reader might also notice some intuitive terms, both in the Trepan tree, and in RIPPER rules. For RIPPER, for instance, the fourth rule is rather unintuitive: an applicant that has paid back all his/her previous loans in time (and does not fulfill any of the previous rules) is classified as a bad applicant. In the Trepan tree, the third split has similar intuitiveness problems. This monotonicity issue, as discussed in Sect. 5.3, can restrict or even prohibit the implementation of these models in practical decision support systems.

When we compare the Trepan tree, the RIPPER rule set, and the Re-RX rule example in Table 2, we clearly see the rule expressiveness issue of the different rule outputs. As decision tables seem to provide the most comprehensible decision support system (see Sect. 5.1), we have transformed the RIPPER rule set into a decision table with the use of the Prologa software (Fig. 8).[3] The reader will surely agree that the decision table provides some advantages over the rule set, e.g., where in the rule set one needs to consider the rules in order, this is not the case for the decision table.

Note that the more rules exist, the more compact the decision table will be compared to the set of rules. We mention this, as the benefit of the the decision table is expected to be bigger for the typical, larger rule sets.

[3] Software available at `http://www.econ.kuleuven.ac.be/prologa/`.

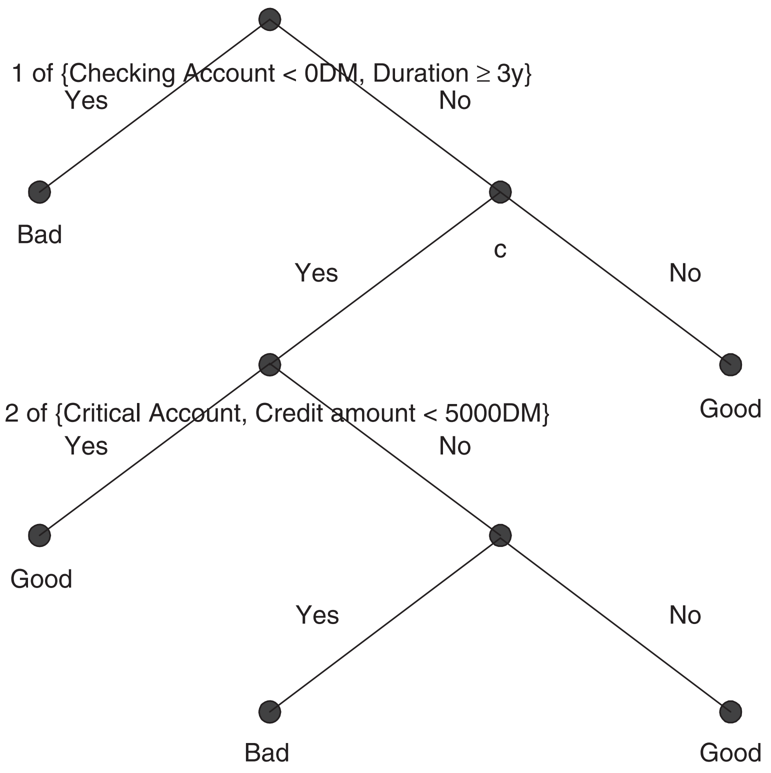

1 of {Checking Account < 0DM, Duration ≥ 3y}
Yes
No
Bad
c
Yes
No
Good
2 of {Critical Account, Credit amount < 5000DM}
Yes
No
Good
Yes
No
Bad
Good

Fig. 7. Trepan tree

Table 3. Example rule set from RIPPER

if (Checking Account $<$ 0DM) and (Housing = rent)
then Applicant = Bad

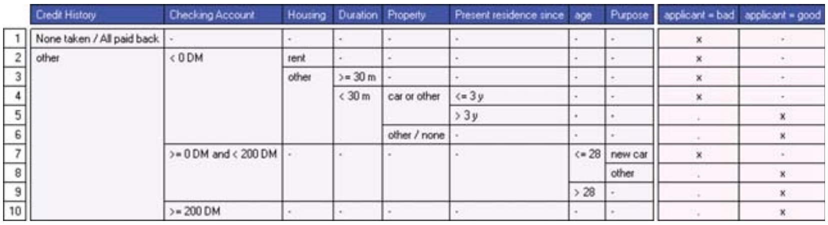
elseif (Checking Account $<$ 0DM) and (Property = car or other) and (Present residence since $\leq$ 3y)
then Applicant = Bad

elseif (Checking Account $<$ 0DM) and (Duration $\geq$ 30m)
then Applicant = Bad

elseif (Credit history = None taken/All paid back duly)
then Applicant = Bad

elseif ($0 \leq$ Checking Account $<$ 200DM) and (Age $\leq$ 28) and (Purpose = new car)
then Applicant = Bad

else Applicant = Good



	Credit History	Checking Account	Housing	Duration	Property	Present residence since	age	Purpose	applicant = bad	applicant = good
1	None taken / All paid back	-	-	-	-	-	-	-	x	-
2	other	< 0 DM	rent	-	-	-	-	-	x	-
3			other	>= 30 m	-	-	-	-	x	-
4				< 30 m	car or other	<= 3 y	-	-	x	-
5						> 3 y	-	-	.	x
6					other / none	-	-	-	.	x
7		>= 0 DM and < 200 DM	-	-	-	-	<= 28	new car	x	-
8								other	.	x
9							> 28	-	.	x
10		>= 200 DM	-	-	-	-	-	-	.	x

Fig. 8. Decision table classifying a loan applicant, based on RIPPER's rule set of Table 3

7 Alternatives to Rule Extraction

A final critical point needs to be made concerning SVM rule extraction, since other alternatives exist for obtaining comprehensible models. Although the expressiveness of rules is superior to the alternative outputs, it is possible that one of the alternatives is more suitable for certain applications. Therefore we mention some of the most interesting ones in this final section.

7.1 Inverse Classification

Sensitivity analysis is the study of how input changes influence the change in the output, and can be summarized by (17).

$$f(x + \Delta x) = f(x) + \Delta f \tag{17}$$

Inverse classification is closely related to sensitivity analysis and involves *determining the minimum required change to a data point in order to reclassify it as a member of a (different) preferred class* [39]. This problem is called the inverse classification problem, since the usual mapping is from a data point to a class, while here it is the other way around. Such information can be very helpful in a variety of domains: companies, and even countries, can determine what macro-economic variables should change so as to obtain a better bond, competitiveness or terrorism rating. Similarly, a financial institution can provide (more) specific reasons why a customer's application was rejected, by simply stating how the customer can change to the good class, e.g., by increasing income by a certain amount. A heuristic, genetic-algorithm based approach is used in [39].

The use of distance to the nearest support vector as an approximator for the distance to the decision boundary (thus distance to the other class) might be useful in this approach, and constitutes an interesting issue for future research within this domain.

7.2 Self Organizing Maps

SOMs were introduced in 1982 by Teuvo Kohonen [37] and have been used in a wide array of applications like the visualization of high-dimensional data [67], clustering of text documents [27], identification of fraudulent insurance claims

[12] and many others. An extensive overview of successful applications can be found in [22] and [38]. A SOM is a feedforward neural network consisting of two layers [57]. The neurons from the output layer are usually ordered in a low-dimensional (typically two-dimensional) grid.

Self-organising maps are often called topology-preserving maps, in the sense that similar inputs, will be close to each other in the final output grid. First, the SOM is trained on the available data with the independent variables, followed by assigning a color to each neuron based on the classification of the data instances projected on that neuron. In Fig. 9, light and dark shades indicate respectively "non corrupt" and "highly corrupt" countries, according their Corruption Perceptions Index [31]. We can observe that the lower right corner contains the countries perceived to be most corrupt (e.g., Pakistan (PAK), Nigeria (NIG), Cameroon (CMR) and Bangladesh (BGD)). At the opposite side, it can easily be noted that the North-European countries are perceived to be among the least corrupt: they are all situated in the white-colored region at the top of the map. As the values for the three consecutive years are denoted with different labels (e.g., usa, Usa and USA), one can notice that most European countries were projected on the upper-half of the map indicating a modest amount of corruption and that several countries seemed to be in transition towards a more European, less corrupt, model.

7.3 Incremental Approach

An incremental approach is followed so as to find a trade-off between simple, linear techniques with excellent readability, but restricted model flexibility and

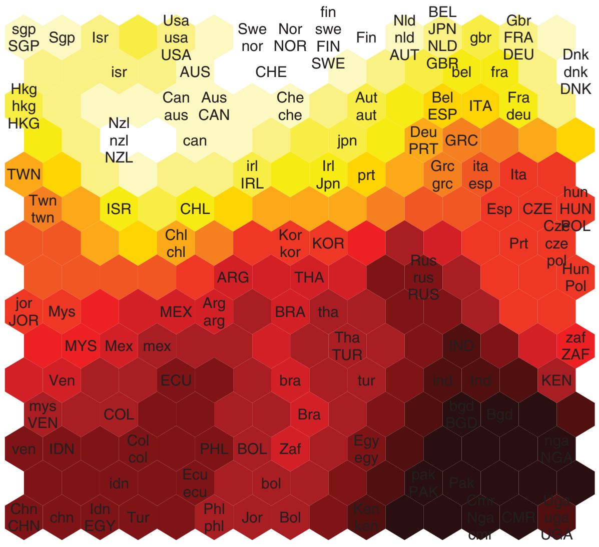


Fig. 9. Visualizing corruption index with the use of SOMs [31]

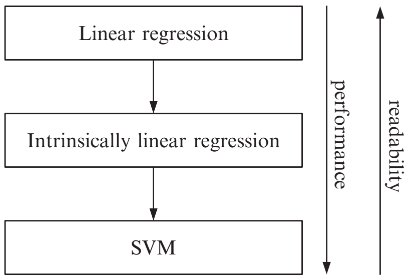


Fig. 10. From linear to non-linear models

complexity, and advanced techniques with reduced readability but extended flexibility and generalization behavior, as shown by Fig. 10.

The approach, introduced by Van Gestel et al. for credit scoring [60–62], constructs an ordinal logistic regression model in a first step, yielding latent variable z_L. In this linear model, a ratio x_i influences the latent variable z_L in a linear way. However, it seems reasonable that a change of a ratio with 5% should not always have the same influence on the score [9]. Therefore, non-linear univariate transformations of the independent variables ($x_i \mapsto f_i(x_i)$) are to be considered in the next step. This model is called intrinsically linear in the sense that after applying the non-linear transformation to the explanatory variables, a linear model is being fit [9]. A non-linear transformation of the explanatory variables is applied only when it is reasonable from both financial as well as statistical point of view. For instance, for rating insurance companies, the investment yield variable is transformed as shown by Fig. 11,[4] with cutoff values at 0% and 5%; values more than 5% do not attribute to a better rating because despite the average, it may indicate higher investment risk [62].

Finally, non-linear SVM terms are estimated on top of the existing intrinsically model by means of a partial regression, where the parameters $\boldsymbol{\beta}$ are estimated first assuming that $\boldsymbol{w} = 0$ and in a second step the $\boldsymbol{w}$ parameters are optimized with $\boldsymbol{\beta}$ fixed from the previous step. This combination of linear, intrinsically linear and SVM terms is formulated in (18).

$$\begin{aligned} z_L &= -\beta_1 x_1 - \beta_2 x_2 - \ldots - \beta_n x_n \\ z_{\mathrm{IL}} &= -\beta_1 x_1 - \ldots - \beta_m x_m - \beta_{m+1} f_{m+1}(x_{m+1}) - \ldots - \beta_n f_n(x_n) \\ z_{\mathrm{IL+SVM}} &= \overbrace{\overbrace{-\beta_1 x_1 - \ldots - \beta_m x_m}^{\textbf{linear part}} \overbrace{-\beta_{m+1} f(x_{m+1}) - \ldots - \beta_n f(x_n)}^{\textbf{nonlinear transformations}}}^{\textbf{intrinsically linear part}} \\ &\quad \underbrace{-w_1 \varphi_1(\boldsymbol{x}) - \ldots - w_p \varphi_p(\boldsymbol{x})}_{\textbf{SVM terms}} \end{aligned} \tag{18}$$

[4] A sigmoid transformation $x \mapsto f(x) = \tanh(x \times a + b)$, was used, with hyperparameters a and b estimated via a grid search.

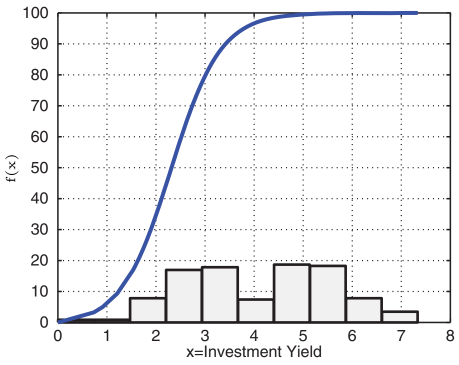


Fig. 11. Visualisation of the univariate non-linear transformations applied to the investment yield variable in the intrinsically linear model [62]

The incremental approach has been applied to provide credit ratings for countries [60], banks [61] and insurance companies [62].

8 Conclusion

In recent years, the SVM has proved its worth and has been successfully applied in a variety of domains. However, what remains as an obstacle is its opaqueness. This lack of transparency can be overcome through rule extraction. SVM rule extraction is still in its infancy, certainly compared to ANN rule extraction. As we put forward in this chapter, much can be transferred from the well researched ANN rule extraction domain, issues as well as the pedagogical rule extraction techniques. In this chapter, we have listed existing SVM rule extraction techniques and complemented this list with the often overlooked pedagogical ANN rule extraction techniques.

Many of the issues related to this field are still completely neglected or under-researched within the rule extraction domain, such as the need for intuitive rule sets, the ability to handle high dimensional data, and a ranking for rule expressiveness among the different rule outputs. We hope this chapter will contribute to further research to this very relevant topic.

9 Acknowledgement

We would like to thank the Flemish Research Council (FWO) for financial support (Grant G.0615.05).

References

1. E. Altendorf, E. Restificar, and T.G. Dietterich. Learning from sparse data by exploiting monotonicity constraints. In *Proceedings of the 21st Conference on Uncertainty in Artificial Intelligence*, Edinburgh, Scotland, 2005.

2. Robert Andrews, Joachim Diederich, and Alan B. Tickle. Survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, 8(6):373–389, 1995.
3. B. Baesens, R. Setiono, C. Mues, and J. Vanthienen. Using neural network rule extraction and decision tables for credit-risk evaluation. *Management Science*, 49(3):312–329, 2003.
4. B. Baesens, T. Van Gestel, S. Viaene, M. Stepanova, J.A.K. Suykens, and J. Vanthienen. Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the Operational Research Society*, 54(6):627–635, 2003.
5. N. Barakat and J. Diederich. Learning-based rule-extraction from support vector machines. In *14th International Conference on Computer Theory and Applications ICCTA 2004 Proceedings*, Alexandria, Egypt, 2004.
6. N. Barakat and J. Diederich. Eclectic rule-extraction from support vector machines. *International Journal of Computational Intelligence*, 2(1):59–62, 2005.
7. A. Ben-David. Monotonicity maintenance in information-theoretic machine learning algorithms. *Machine Learning*, 19(1):29–43, 1995.
8. C.M. Bishop. *Neural networks for pattern recognition.* Oxford University Press, Oxford, UK, 1996.
9. G.E.P. Box and D.R. Cox. An analysis of transformations. *Journal of the Royal Statistical Society Series B*, 26:211–243, 1964.
10. O. Boz. *Converting A Trained Neural Network To A Decision Tree. DecText - Decision Tree Extractor.* PhD thesis, Lehigh University, Department of Computer Science and Engineering, 2000.
11. L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression trees.* Wadsworth and Brooks, Monterey, CA, 1994.
12. P.L. Brockett, X. Xia, and R. Derrig. Using kohonen's self-organizing feature map to uncover automobile bodily injury claims fraud. *International Journal of Risk and Insurance*, 65:245–274, 1998.
13. M. Brown, W. Grundy, D. Lin, N. Cristianini, C. Sugnet, M. Ares Jr., and D. Haussler. Support vector machine classification of microarray gene expression data. Technical UCSC-CRL-99-09, University of California, Santa Cruz, 1999.
14. F. Chen. Learning accurate and understandable rules from SVM classifiers. Master's thesis, Simon Fraser University, 2004.
15. P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3(4):261–283, 1989.
16. W. Cohen. Fast effective rule induction. In Armand Prieditis and Stuart Russell, editors, *Proceedings of the 12th International Conference on Machine Learning*, pages 115–123, Tahoe City, CA, 1995. Morgan Kaufmann Publishers.
17. M.W. Craven. *Extracting Comprehensible Models from Trained Neural Networks.* PhD thesis, Department of Computer Sciences, University of Wisconsin-Madison, 1996.
18. M.W. Craven and J.W. Shavlik. Extracting tree-structured representations of trained networks. In D.S. Touretzky, M.C. Mozer, and M.E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 24–30. The MIT Press, 1996.
19. M.W. Craven and J.W. Shavlik. Rule extraction: Where do we go from here? Working paper, University of Wisconsin, Department of Computer Sciences, 1999.

20. N. Cristianini and J. Shawe-Taylor. *An introduction to Support Vector Machines and Other Kernel-Based Learning Methods.* Cambridge University Press, New York, NY, USA, 2000.
21. H. Daniels and M. Velikova. Derivation of monotone decision models from non-monotone data. Discussion Paper 30, Tilburg University, Center for Economic Research, 2003.
22. G. Deboeck and T. Kohonen. *Visual Explorations in Finance with selforganizing maps.* Springer-Verlag, 1998.
23. EMC. Groundbreaking study forecasts a staggering 988 billion gigabytes of digital information created in 2010. Technical report, EMC, March 6, 2007.
24. A.J. Feelders and M. Pardoel. Pruning for monotone classification trees. In *Advanced in intelligent data analysis V*, volume 2810, pages 1–12. Springer, 2003.
25. G. Fung, S. Sandilya, and R.B. Rao. Rule extraction from linear support vector machines. In *Proceedings of the 11th ACM SIGKDD international Conference on Knowledge Discovery in Data Mining*, pages 32–40, 2005.
26. S. Hettich and S. D. Bay. The uci kdd archive [http://kdd.ics.uci.edu], 1996.
27. T. Honkela, S. Kaski, K. Lagus, and T. Kohonen. WEBSOM—self-organizing maps of document collections. In *Proceedings of Workshop on Self-Organizing Maps (WSOM'97)*, pages 310–315. Helsinki University of Technology, Neural Networks Research Centre, Espoo, Finland, 1997.
28. J. Huysmans, B. Baesens, and J. Vanthienen. ITER: an algorithm for predictive regression rule extraction. In *8th International Conference on Data Warehousing and Knowledge Discovery (DaWaK 2006)*, volume 4081, pages 270–279. Springer Verlag, lncs 4081, 2006.
29. J. Huysmans, B. Baesens, and J. Vanthienen. Using rule extraction to improve the comprehensibility of predictive models. Research 0612, K.U.Leuven KBI, 2006.
30. J. Huysmans, B. Baesens, and J. Vanthienen. Minerva: sequential covering for rule extraction. 2007.
31. J. Huysmans, D. Martens, B. Baesens, J. Vanthienen, and T. van Gestel. Country corruption analysis with self organizing maps and support vector machines. In *International Workshop on Intelligence and Security Informatics (PAKDD-WISI 2006)*, volume 3917, pages 103–114. Springer Verlag, lncs 3917, 2006.
32. J. Huysmans, C. Mues, B. Baesens, and J. Vanthienen. An empirical evaluation of the comprehensibility of decision table, tree and rule based predictive models. 2007.
33. T. Joachims. *Learning to Classify Text Using Support Vector Machines: Methods, Theory and Algorithms.* Kluwer Academic Publishers, Norwell, MA, USA, 2002.
34. U. Johansson, R. König, and L. Niklasson. Rule extraction from trained neural networks using genetic programming. In *Joint 13th International Conference on Artificial Neural Networks and 10th International Conference on Neural Information Processing, ICANN/ICONIP 2003*, pages 13–16, 2003.
35. U. Johansson, R. König, and L. Niklasson. The truth is in there - rule extraction from opaque models using genetic programming. In *17th International Florida AI Research Symposium Conference FLAIRS Proceedings*, 2004.

36. R. Kohavi and J.R. Quinlan. Decision-tree discovery. In W. Klosgen and J. Zytkow, editors, *Handbook of Data Mining and Knowledge Discovery*, pages 267–276. Oxford University Press, 2002.
37. T. Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.
38. T. Kohonen. *Self-Organising Maps.* Springer-Verlag, 1995.
39. M. Mannino and M. Koushik. The cost-minimizing inverse classification problem: A genetic algorithm approach. *Decision Support Systems*, 29:283–300, 2000.
40. U. Markowska-Kaczmar and M. Chumieja. Discovering the mysteries of neural networks. *International Journal of Hybrid Intelligent Systems*, 1(3–4):153–163, 2004.
41. U. Markowska-Kaczmar and W. Trelak. Extraction of fuzzy rules from trained neural network using evolutionary algorithm. In *European Symposium on Artificial Neural Networks (ESANN)*, pages 149–154, 2003.
42. D. Martens, B. Baesens, T. Van Gestel, and J. Vanthienen. Comprehensible credit scoring models using rule extraction from support vector machines. *European Journal of Operational Research*, Forthcoming.
43. D. Martens, M. De Backer, R. Haesen, B. Baesens, C. Mues, and J. Vanthienen. Ant-based approach to the knowledge fusion problem. In *Proceedings of the Fifth International Workshop on Ant Colony Optimization and Swarm Intelligence*, Lecture Notes in Computer Science, pages 85–96. Springer, 2006.
44. D. Martens, M. De Backer, R. Haesen, M. Snoeck, J. Vanthienen, and B. Baesens. Classification with ant colony optimization. *IEEE Transaction on Evolutionary Computation*, Forthcoming.
45. R. Michalski. On the quasi-minimal solution of the general covering problem. In *Proceedings of the 5th International Symposium on Information Processing (FCIP 69)*, pages 125–128, 1969.
46. H. Núñez, C. Angulo, and A. Català. Rule extraction from support vector machines. In *European Symposium on Artificial Neural Networks (ESANN)*, pages 107–112, 2002.
47. M. Pazzani, S. Mani, and W. Shankle. Acceptance by medical experts of rules generated by machine learning. *Methods of Information in Medicine*, 40(5):380–385, 2001.
48. J. R. Quinlan. Induction of decision trees. *Machine Learning*, 1(1):81–106, 1986.
49. J.R. Quinlan. *C4.5 programs for machine learning.* Morgan Kaufmann, 1993.
50. J.R. Rabuñal, J. Dorado, A. Pazos, J. Pereira, and D. Rivero. A new approach to the extraction of ANN rules and to their generalization capacity through GP. *Neural Computation*, 16(47):1483–1523, 2004.
51. B.D. Ripley. Neural networks and related methods for classification. *Journal of the Royal Statistical Society B*, 56:409–456, 1994.
52. G.P.J. Schmitz, C. Aldrich, and F.S. Gouws. Ann-dt: An algorithm for the extraction of decision trees from artificial neural networks. *IEEE Transactions on Neural Networks*, 10(6):1392–1401, 1999.
53. R. Setiono, B. Baesens, and C. Mues. Risk management and regulatory compliance: A data mining framework based on neural network rule extraction. In *Proceedings of the International Conference on Information Systems (ICIS 2006)*, 2006.
54. J. Sill. Monotonic networks. In *Advances in Neural Information Processing Systems*, volume 10. The MIT Press, 1998.

55. D.W. Silverman. *Density Estimation for Statistics and Data Analysis*. Chapman and Hall, 1986.
56. I.A. Taha and J. Ghosh. Symbolic interpretation of artificial neural networks. *IEEE Transactions on Knowledge and Data Engineering*, 11(3):448–463, 1999.
57. P.-N. Tan, M. Steinbach, and V. Kumar. *Introduction to Data Mining*. Addison Wesley, Boston, MA, 2005.
58. M. Tipping. The relevance vector machine. In *Advances in Neural Information Processing Systems, San Mateo, CA*. Morgan Kaufmann, 2000.
59. M. Tipping. Sparse bayesian learning and the relevance vector machine. *Journal of Machine Learning Research*, 1:211–244, 2001.
60. T. Van Gestel, B. Baesens, P. Van Dijcke, J. Garcia, J.A.K. Suykens, and J. Vanthienen. A process model to develop an internal rating system: credit ratings. *Decision Support Systems*, forthcoming.
61. T. Van Gestel, B. Baesens, P. Van Dijcke, J.A.K. Suykens, J. Garcia, and T. Alderweireld. Linear and non-linear credit scoring by combining logistic regression and support vector machines. *Journal of Credit Risk*, 1(4), 2006.
62. T. Van Gestel, D. Martens, B. Baesens, D. Feremans, J; Huysmans, and J. Vanthienen. Forecasting and analyzing insurance companies ratings.
63. T. Van Gestel, J.A.K. Suykens, B. Baesens, S. Viaene, J. Vanthienen, G. Dedene, B. De Moor, and J. Vandewalle. Benchmarking least squares support vector machine classifiers. *CTEO, Technical Report 0037, K.U. Leuven, Belgium*, 2000.
64. V. N. Vapnik. *The nature of statistical learning theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.
65. M. Velikova and H. Daniels. Decision trees for monotone price models. *Computational Management Science*, 1(3–4):231–244, 2004.
66. M. Velikova, H. Daniels, and A. Feelders. Solving partially monotone problems with neural networks. In *Proceedings of the International Conference on Neural Networks*, Vienna, Austria, March 2006.
67. J. Vesanto. Som-based data visualization methods. *Intelligent Data Analysis*, 3:111–26, 1999.
68. Z.-H. Zhou, Y. Jiang, and S.-F. Chen. Extracting symbolic rules from trained neural network ensembles. *AI Communications*, 16(1):3–15, 2003.

Part II

Algorithms and Techniques

Rule Extraction for Transfer Learning

Lisa Torrey[1], Jude Shavlik[1], Trevor Walker[1], and Richard Maclin[2]

[1] Department of Computer Science, University of Wisconsin, Madison, WI 53706, USA
[2] Department of Computer Science, University of Minnesota, Duluth, MN 55812, USA

Summary. This chapter discusses transfer learning, which is one practical application of rule extraction. In transfer learning, information from one learning experience is applied to speed up learning in a related task. The chapter describes several techniques for transfer learning in SVM-basedreinforcement learning, and shows results from a case study.

1 Introduction

Typically rule extraction is done for the purposes of human interpretation. However, there are other possible applications of rule extraction. One practical application is *transfer learning*, in which knowledge learned in one task is used to aid in learning a related task. The extracted rules, which explain the learned solution to the first task, can be considered *advice* on how to approach the second task.

Transfer learning, besides being desirable in its own right, could be viewed as another way to evaluate extracted rules. That is, how well extracted knowledge transfers to a related task is a potential way of judging the value of the rule extraction algorithm. Thus transfer can be used as an alternative to traditional measures such as complexity and faithfulness to the original model. While this method is more objective and more computational than some of the traditional measures, it requires a trusted algorithm for making use of extracted knowledge.

This chapter discusses transfer learning via advice taking, in particular for reinforcement learning (RL) tasks that use support vector machines (SVMs) as function approximators. After some background information on transfer, advice, and RL with SVMs, it describes two methods for rule extraction in this context and presents a case study from our recent research.

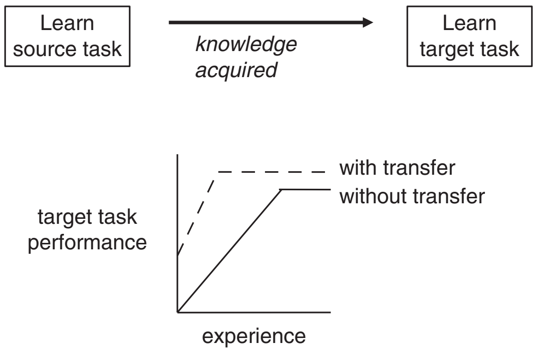


Fig. 1. With transfer from a related source task, the learning curve for the target task might improve in one or more of the ways shown above: higher initial performance, faster performance increase, and higher asymptotic performance

2 Transfer Learning and Advice Taking

Machine learning tasks are often addressed independently, under the implicit assumption that each new task has no relation to the tasks that came before. However, many machine learning domains contain several related tasks. Instead of learning each one from scratch, agents in such domains should be able to use knowledge learned in previous tasks to speed up learning in later ones. This is the goal of *transfer learning* (see Fig. 1).

For example, consider the domain of simulated soccer (e.g., [10]). Suppose an agent has learned a game of keeping the ball from its opponents by passing amongst its teammates, and the next game to learn is to score goals against opponents. Since these games have some similarities, the agent could benefit from using its knowledge from the first game while learning the second. In this case we refer to the first game as the *source task* and the second game as the *target task*.

There are several approaches to transfer learning in RL that do not involve rule extraction. Examples are Taylor et al. [15], who transfer Q-functions directly, and Soni and Singh [11], who transfer multi-step action sequences known as options. Here, however, we will focus on approaches that use rule extraction and apply those rules as advice for the target task.

Advice is a set of approximately correct instructions for a task. It may have some errors, and it usually does not provide a complete solution. Due to differences between the source and target tasks in transfer learning, extracted rules are likely to have these characteristics. In advice-taking RL algorithms, advice can be followed, refined, or ignored according to its value. In the target task, this means extracted rules can be obeyed if they lead to positive transfer, but quickly get refined or ignored if they lead to negative transfer.

For example, in the simulated soccer domain where the state features are distances and angles between players, some reasonable advice might look like:

```
IF   distance(nearestOpponent, self) ≤ 5 AND
     angle(teammate, self, minAngleOpponent(teammate)) ≥ 40
THEN prefer pass(teammate) over other actions
```

This advice tells an agent to pass to a teammate when two conditions hold: (1) an opponent is too close to the agent, and (2) the smallest angle from the teammate to the agent to an opponent is large enough (i.e. there is an open passing lane).

There is a substantial body of work on advice taking in RL. Examples are Clouse and Utgoff [2], who allow a human observer to step in and advise the learner to take a specific action; Driessens and Dzeroski [4], who use human guidance to create a partial initial Q-function; and Kuhlmann et al. [5], who propose giving advice that increases Q-values by a fixed amount. As most advice-taking approaches do, these studies assume that advice comes directly from a human interacting with the learner. Here, however, we will focus on advice that is automatically extracted from a source task.

3 SVMs in Reinforcement Learning

In reinforcement learning [14], an agent navigates through an environment trying to earn rewards. The environment's state is typically described by a set of features. After each action the agent takes, it receives a reward and observes the next state.

In Q-learning [19], one common form of RL, the agent builds a Q-function to estimate the long-term value of taking an action from a state. An agent's *policy* is typically to take the action with the highest Q-value in the current state, except for occasional exploratory actions that are needed to discover better policies. After taking the action and receiving a reward, the agent updates its Q-value estimates for the current state.

The Q-function can be approximated with SVM regression models [3], so that each action's Q-value is estimated by a weighted linear sum of the state features. In this case, after taking a sequence of actions and receiving a corresponding sequence of rewards, the RL agent learns a linear SVM with weights that minimize:

$$\text{ModelSize} + C \times \text{DataMisfit}$$

Here *ModelSize* is the sum of the absolute values of the feature weights, and *DataMisfit* is the disagreement between the learned function's outputs and the correct outputs (estimated based on the rewards received). The numeric parameter C specifies the relative importance of minimizing disagreement with the data versus finding a simple model.

The agent learns many intermediate SVM models as its performance improves. Eventually it reaches an asymptote, so there is a final model that represents the learned task. This is the model from which rules are extracted to perform transfer learning.

Advice can be included in this RL algorithm with *Knowledge-Based Support Vector Regression*, abbreviated KBKR [6–8]. This algorithm adds another term to the optimization problem, so that it minimizes:

$$\text{ModelSize} + C \times \text{DataMisfit} + \mu \times \text{AdviceMisfit}$$

Here *AdviceMisfit* is the disagreement between the learned function's outputs and the advice constraints. The numeric parameter *mu* specifies the relative importance of minimizing disagreement with the advice versus minimizing the original quantity. Over time *mu* decays and C increases so that the advice has less impact as the learner gains experience and no longer requires guidance.

Advice therefore becomes a soft constraint on the task solution. Depending on whether the advice agrees with the training examples, the learner can fully follow the rule, only follow it approximately (which is like refining it), or ignore it altogether.

The details behind this intuitive idea are as follows. Let A be a matrix in which each row contains the feature values for a training example. Let y be the vector of Q-value estimates (for a single action) for this set of examples. We wish to model the Q-function as a weighted sum:

$$Aw + b\overrightarrow{e} = y, \tag{1}$$

where w is a vector of weights, b is a scalar offset, and $\overrightarrow{e}$ denotes a vector of ones (we omit this for simplicity from now on).

To learn a good Q-function, we want to find w and b to satisfy this equation. However, an exact solution may not exist. Also, it is preferable to have non-zero weights for only a few important features in order to keep the model simple and avoid overfitting the training examples. Therefore we include a vector of *slack* variables s to allow inaccuracies on some examples, and a penalty parameter C for trading off these inaccuracies with the complexity of the solution. The linear equation then becomes a linear minimization problem:

$$\begin{aligned} \min_{(w,b,s)} \quad & ||w||_1 + \nu|b| + C||s||_1 \\ s.t. \quad & |Aw + b - y| \leq s, \end{aligned} \tag{2}$$

where $|\cdot|$ denotes an absolute value, $||\cdot||_1$ denotes a sum of absolute values from a vector, and ν is a penalty on the offset term that discourages constant models. Solving this problem means finding w, b, and s such that the penalties are minimized but the model's value for $A_i w + b$ is within s_i of y_i. The RL agent therefore finds a compromise between accuracy and simplicity.

Advice generated from the source task can be expressed in the form:

$$Bx \leq d \Longrightarrow Q_p(x) - Q_n(x) \geq \beta, \tag{3}$$

where B is a matrix and d is a vector [7]. This can be read as: if the current state satisfies the set of linear inequalities $Bx \leq d$, the Q-value of the preferred action p should exceed that of the non-preferred action n by at least β. For example, consider the advice rule from Sect. 2:

```
IF   distance(nearestOpponent, self) ≤ 5 AND
     angle(teammate, self, minAngleOpponent(teammate)) ≥ 40
THEN prefer pass(teammate) over other actions
```

If we assume that the two features mentioned in this rule make up the entire feature vector x, then we would express this advice with:

$$B = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix},$$

$$d = \begin{pmatrix} 5 \\ -40 \end{pmatrix}, \tag{4}$$

where Q_p represents the Q-value of *pass*, β is a constant small fraction of the Q-value range, and there is one equation for each other action n. Note that the $\geq$ inequality in the second constraint is converted to a $\leq$ inequality by multiplying both sides of the equation by -1.

Just as our method allows some inaccuracy on the training examples, it allows advice to be satisfied only partially. To do so, following Mangasarian et al. [8], we introduce slack variables z and ζ and penalty parameters μ_1 and μ_2 for trading off the impact of the advice on the solution with the impact of the training examples. The resulting linear program, which finds Q-functions for all of the actions simultaneously, is:

$$\begin{aligned} &\min_{(w_a, b_a, s_a, z_i, \zeta_i \geq 0, u_i \geq 0)} \\ &\sum_{a=1}^{m} (||w_a||_1 + \nu |b_a| + C||s_a||_1) + \sum_{i=1}^{k} (\mu_1 ||z_i||_1 + \mu_2 \zeta_i) \\ &\quad \text{s.t.} \quad \text{for each action } a \in \{1, \ldots, m\} : \\ &\qquad |A_a w_a + b_a - y_a| \leq s_a \\ &\qquad \text{for each advice item } i \in \{1, \ldots, k\} : \\ &\qquad |w_p - w_n + B_i^T u_i| \leq z_i \\ &\qquad - d^T u_i + \zeta_i \geq \beta_i - b_p + b_n . \end{aligned} \tag{5}$$

By solving this optimization problem, the RL agents learn a policy that satisfies the advice as long as it does not disagree too much with the training examples.

4 Extracting Rules from an RL Source Task

The final SVM model representing the learned RL task can be separated into several models (one per action). Each SVM calculates the Q-value of one action as a weighted sum of the state features. We discuss two methods for extracting transfer advice from these models. In both approaches, the resulting rules are of the form:

```
IF   condition
THEN prefer one action over the others
```

where the *condition* is a conjunction of linear constraints on the state features.

As is common in transfer learning, our methods assume the existence of a *mapping* showing how features and actions correspond between the source and target tasks. Transfer is a reasonable endeavor only if there is substantial correspondence. However, there may be some features and actions in one task that do not have parallels in the other, and the transfer methods we discuss handle this problem in different ways.

4.1 Acquiring Rules from the Q-function

One approach for generating rules from RL source-task models first appeared in Torrey et al. [18]. It is called *policy transfer*, because it builds a set of rules to describe the entire policy represented by the source-task models. To reference the neural-network rule-extraction literature, this is best described as a *decompositional* strategy [1], in which the internal mechanics of the model affect the extracted rules.

Recall that the *policy* of an RL agent determines which action it will choose – generally the action with the highest Q-value. The policy can therefore be expressed as a set of rules, one for each action, saying to prefer that action when its SVM regression model assigns it the highest Q-value. Alternatively, it can be expressed as a set of rules, two for each pair of actions, saying to prefer one action over the other when its SVM regression model assigns it the higher Q-value of the pair. Table 1 gives a simple example of the construction of this pairwise ruleset.

The policy-transfer rules effectively tell the target-task learner to pretend it is performing the source task (via the mapping) and choose actions accordingly. They do not attempt to constrain the actual Q-values of target-task

Table 1. An example of constructing policy-transfer rules

SOURCE TASK MODEL:	ADVICE FORMAT:
$Q_a = w_{a1} * f_1 + w_{a2} * f_2$	IF $Q'_a - Q'_b \geq \Delta$
$Q_b = w_{b1} * f_1$	THEN prefer a' to b'
$Q_c = w_{c2} * f_2$	(and so on for each pair of actions)
USER-PROVIDED MAPPING:	FULL ADVICE EXPRESSION:
$(a, b, c) \longrightarrow (a', b', c')$	IF $(w_{a1} - w_{b1}) * f'_1 + w_{a2} * f'_2 \geq \Delta$
$(f_1, f_2) \longrightarrow (f'_1, f'_2)$	THEN prefer a' to b'
	(and so on for each pair of actions)
TRANSLATED MODEL:	
$Q'_a = w_{a1} * f'_1 + w_{a2} * f'_2$	
$Q'_b = w_{b1} * f'_1$	
$Q'_c = w_{c2} * f'_2$	

The actions in the old task are a, b, and c, and the corresponding actions in the new task are a', b', and c'. The learned models for the old task are linear Q-value expressions with weights w and features f, and these are translated into rules that use the corresponding new task features f'

Table 2. Our policy-transfer algorithm

```
GIVEN
  A learned source-task model AND
  A mapping of features and actions from source to target
DO
  for each $a \in Actions(source)$:
    for each $b \neq a \in Actions(source)$ generate advice:
      IF   $Q'_a - Q'_b \geq \Delta$
      THEN prefer $a'$ TO $b'$ in target task
```

actions, but only the relative ordering of the Q-values; this can be important if the Q-value ranges of the tasks differ. We set the parameter Δ to approximately 1% of the target task's Q-value range.

One complication that might arise for this method is if a source-task feature has no corresponding feature in the target task – for example, if the f_1 in Table 1 has no logical f'_1 mapping. In this case, the algorithm gives f'_1 a constant value. By default it uses the average value that the f_1 feature takes in the source-task data, although the user may also tell it to use the minimum or maximum value if that seems more appropriate.

Table 2 summarizes our policy-transfer algorithm in pseudocode.

We present experiments with policy transfer as part of a case study in Sect. 5. The rules constructed by this method are long, complex and not well suited to human interpretation. They capture very fine details of the source-task models, which may not actually be desirable for transfer learning because general principles are more likely to transfer to new tasks than specific details. The approach we discuss next was designed to address these shortcomings.

4.2 Acquiring Rules from Observed Behavior

A second approach for generating rules from RL source-task models is presented in Torrey et al. [17]. It is called *skill transfer*, because it learns rules that represent important source-task skills. In the neural-network rule-extraction literature, this falls under the category of *pedagogical* strategies [1], in which the model is treated as a black box and the rules mimic its outputs.

Skill transfer is intended to capture general knowledge rather than fine detail. Instead of transferring an entire policy, this method transfers only the skills that the source and target tasks have in common. A *skill* is associated with one action, and describes the circumstances under which that action should be taken. Our skill transfer algorithm learns skills by observing behavior in the source task and applying inductive logic programming [9].

Inductive Logic Programming Inductive logic programming (ILP) is a method for learning first-order rules to explain examples. For example, recall the rule from Sect. 2:

IF distance(nearestOpponent, self) $<$ 5 AND
angle(teammate, self, minAngleOpponent(teammate)) $>$ 40
THEN prefer pass(teammate) over other actions

This rule might describe the *pass* skill in soccer: pass to a teammate if an opponent is too close and the passing lane is open. If the symbol *teammate* refers to a specific teammate object, then the rule is called *propositional*; if it is a variable that can refer to any teammate object, then the rule can be called *first-order*. The first-order version is more powerful and more general than the propositional version, since it is more likely to capture the essential elements of the passing skill.

A common approach for ILP is to start by selecting a *seed* example, such as one state in which the soccer player performed the *pass* action. It then calculates a set of facts that are true for this example, which might include *distance(nearestOpponent, self) <5* and *angle(teammate, self, minAngleOpponent(teammate)) >40* along with many other less relevant facts. The clause that contains all these constraints is called the *bottom clause* [12]. A rule is then learned by searching for some subset of the bottom clause that is more general, covering many positive examples but few negative examples. The search algorithm could be any of those familiar to students of general artificial intelligence, such as heuristic search and randomized search. Figure 2 gives an example of a top-down heuristic search.

We use the Prolog-based Aleph software package [12] to perform ILP. The metric we use to score rules and select the best is F(β), a generalization of the more familiar F(1) metric, with $\beta^2 = 0.1$ to put more weight on rule precision

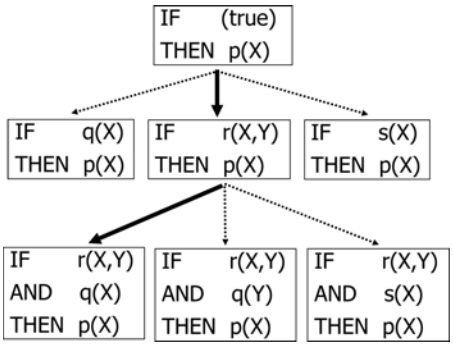


Fig. 2. A top-down heuristic ILP search that adds one constraint at a time. Here the lower-case letters are predicates and the upper-case letters are variables. The rule begins with no constraints, and at each step, the search adds the new constraint that the heuristic function scores highest, indicated by the solid arrow above. New variables that were not in the head $p(X)$ can be introduced into the body of the rule

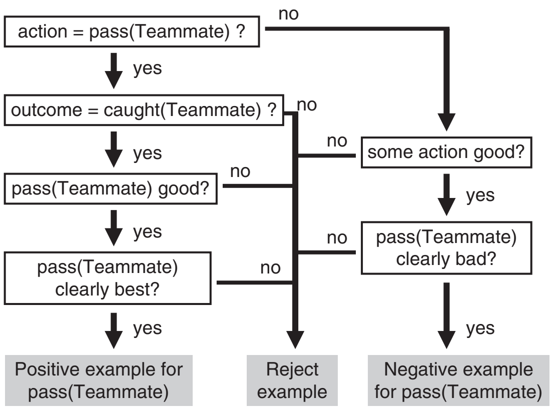

action = pass(Teammate) ?
no
yes
outcome = caught(Teammate) ?
no
yes
pass(Teammate) good?
no
yes
pass(Teammate) clearly best?
no
yes
some action good?
no
yes
pass(Teammate) clearly bad?
no
yes
Positive example for pass(Teammate)
Reject example
Negative example for pass(Teammate)

Fig. 3. Example showing how we select training examples for the skill pass (Teammate)

than rule recall. The search therefore concentrates on finding rules that cover a reasonable amount of data with high accuracy.

Skill Transfer with ILP Skill transfer with ILP is accomplished by applying the standard ILP procedure using states from source-task games as the examples. One challenge for using ILP on reinforcement learning data is to decide which states are positive examples and which are negative examples. This is an important process, and it is not immediately obvious how the choices should be made. Figure 3 illustrates the procedure that we use in Torrey et al. [17] with an example from the simulated soccer domain.

In a positive example, several conditions must be met: the skill is performed, the desired outcome occurs, and the expected Q-value is above the 10th percentile in the training set and is at least 1.05 times the predicted Q-values of all other actions. (The desired outcome of *pass(teammate)*, for example, is that the ball is caught by the *teammate*.) The purpose of these conditions is to remove ambiguous states in which several actions may be good or no actions seem good.

There are two potential types of negative examples. These conditions describe one type: some other action is performed, that action's Q-value is above the 10th percentile in the training set, and the Q-value of the skill being learned is at most 0.95 times that Q-value and below the 50th percentile in the training set. This again rules out ambiguous states. The second type of negative example includes states in which the skill being learned was taken but the desired outcome did not occur (for example, *pass(teammate)* was taken but the ball was caught by an opponent).

Note that the source and target-task features remain propositional in skill transfer, since we use a linear SVM model that requires a fixed-length

Table 3. Our skill-transfer algorithm

```
GIVEN                                 DO
    Games from source task                For each skill to transfer:
    List of skills to be transferred          Collect training examples
    User advice (optional)                    Learn rules with Aleph
                                              Select rule with highest $F(\beta)$ score
                                              Propositionalize rules for target task
```

feature vector. After learning first-order rules, our skill-transfer algorithm propositionalizes them for use in the target task.

As in policy transfer, there may be features that exist in the source but not the target. Here there is a simple solution, however: we restrict the search space of ILP so that learned rules may only contain features that both tasks share. This forces the algorithm to consider only skill definitions that are relevant in the target task.

As in policy transfer, a rule learned by the skill-transfer method describes the conditions under which one action should be preferred over the other actions shared between the source and target task. However, these rules are much simpler and easier to interpret. These are desirable qualities for extracted rules in general, and because they imply more general rules, for transfer learning as well.

Table 3 summarizes our skill transfer algorithm in pseudocode. We present experiments with skill transfer as part of the case study in Sect. 5.

Because the skill-transfer rules are more accessible to human understanding, they open up more possibilities for further human contribution. The user could add constraints to the learned rules reflecting known differences in the source and target tasks, or even provide simple new rules for skills required in the target task that were not in the source. We call this *user advice*, and include an example in the case study. User advice provides a natural and powerful way for a human to guide transfer.

5 Case Study

To illustrate the transfer learning techniques in this chapter, we present a case study in the simulated soccer domain, which is a motivating domain for transfer. The results are reproduced from Torrey et al. [16].

The RoboCup project [10] has the overall goal of producing robotic soccer teams that compete on the human level, but it also has a software simulator for research purposes. Stone and Sutton [13] introduced RoboCup as an RL domain that is challenging because of its large, continuous state space and non-deterministic action effects.

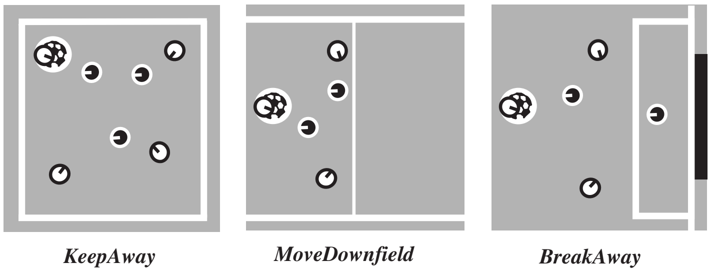


Fig. 4. Snapshots of RoboCup soccer tasks

Since the full game of soccer is quite complex, researchers have developed several smaller games in the RoboCup domain (see Fig. 4). These are inherently multi-agent games, but a standard simplification is to have only one agent (the one in possession of the soccer ball) learning at a time using a model built with combined data from all the agents.

One RoboCup task is M-on-N KeepAway [13], in which the objective of the M reinforcement learners called *keepers* is to keep the ball away from N hand-coded players called *takers*. The keeper with the ball may choose either to hold it or to pass it to a teammate. Keepers without the ball follow a hand-coded strategy to receive passes. The game ends when an opponent takes the ball or when the ball goes out of bounds. The learners receive a +1 reward for each time step their team keeps the ball.

The KeepAway state representation was designed by Stone and Sutton [13] and consists of distances and angles between players. The keepers are ordered by their distance to the learner *k0*, as are the takers.

A second RoboCup task is M-on-N MoveDownfield, where the objective of the M reinforcement learners called *attackers* is to move across a line on the opposing team's side of the field while maintaining possession of the ball. The attacker with the ball may choose to pass to a teammate or to move ahead, away, left, or right with respect to the opponent's goal. Attackers without the ball follow a hand-coded strategy to receive passes. The game ends when they cross the line, when an opponent takes the ball, when the ball goes out of bounds, or after a time limit of 25 seconds. The learners receive symmetrical positive and negative rewards for horizontal movement forward and backward.

The MoveDownfield state representation was introduced in Torrey et al. [17] and consists of distances and angles between players and the goal. The attackers are ordered by their distance to the learner *a0*, as are the defenders.

A third RoboCup task is M-on-N BreakAway, where the objective of the M attackers is to score a goal against $N-1$ hand-coded *defenders* and a hand-coded *goalie*. The attacker with the ball may choose to pass to a teammate, to move ahead, away, left, or right with respect to the opponent's goal, or to shoot at the left, right, or center part of the goal. Attackers without the ball follow a hand-coded strategy to receive passes. The game ends when they score a goal, when an opponent takes the ball, when the ball goes out of bounds, or after a time limit of 10 seconds. The learners receive a +1 reward if they score a goal, and zero reward otherwise.

The BreakAway state representation was introduced in Torrey et al. [18] and consists of distances and angles between players and the goal. The attackers are ordered by their distance to the learner *a0*, as are the non-goalie defenders.

These three RoboCup games have substantial differences in features, actions, and rewards. The goal, goalie, and shoot actions exist in BreakAway but not in the other two tasks. The move actions do not exist in KeepAway but do in the other two tasks. Rewards in KeepAway and MoveDownfield occur for incremental progress, but in BreakAway the reward is more sparse. These differences mean the solutions to the tasks may be quite different. However, some knowledge should clearly be transferable between them, since they share many features and some actions, such as the *pass* action. Furthermore, since these are difficult RL tasks, speeding up learning through transfer is desirable.

5.1 Policy-Transfer Results

Figure 5 displays results from several policy-transfer experiments. The target task in each experiment is 3-on-2 BreakAway, and the three source tasks are 2-on-1 BreakAway, 3-on-2 MoveDownfield, and 3-on-2 KeepAway. One curve is the average of 25 runs of standard reinforcement learning. The other curves are RL with transfer via model reuse from various source tasks. Each transfer

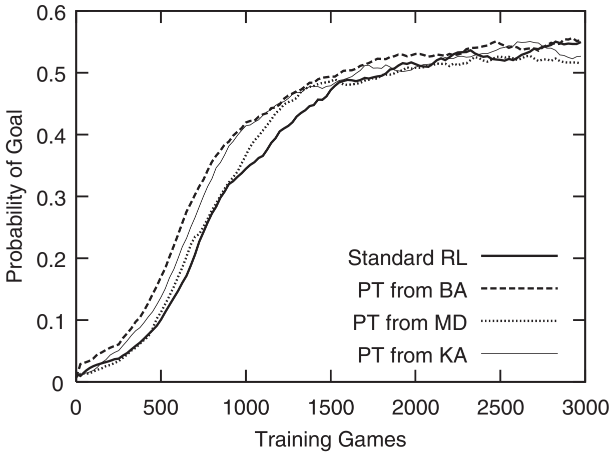


Fig. 5. Probability of scoring a goal while training in 3-on-2 BreakAway with standard RL and policy transfer (PT) from 2-on-1 BreakAway (BA), 3-on-2 MoveDownfield (MD) and 3-on-2 KeepAway (KA)

curve is an average of five transfer runs from five different source runs, for a total of 25 runs (this way, the results include both source and target variance). Because the variance is high, the y-value at each data point is smoothed by averaging over the y-values of the last 10 data points.

The extracted rules are too large and complex to include an example here. However, the results in Fig. 5 show that policy transfer has a small overall positive impact, particularly when the source and target tasks are most similar.

5.2 Skill-Transfer Results

Figure 6 displays results from several skill-transfer experiments. The source and target tasks are the same as in the policy-transfer experiments. Again each transfer curve is an average of 25 runs, consisting of five runs from five different source runs, with the y-value at each point smoothed over the last ten points.

For the experiments in which the target was a different task than the source, we provided some simple handwritten user advice to help with the new required skills. In transfer from KeepAway to BreakAway the important new skills are moving ahead and shooting, and from MoveDownfield to BreakAway only shooting. From one size of BreakAway to another no user advice is needed.

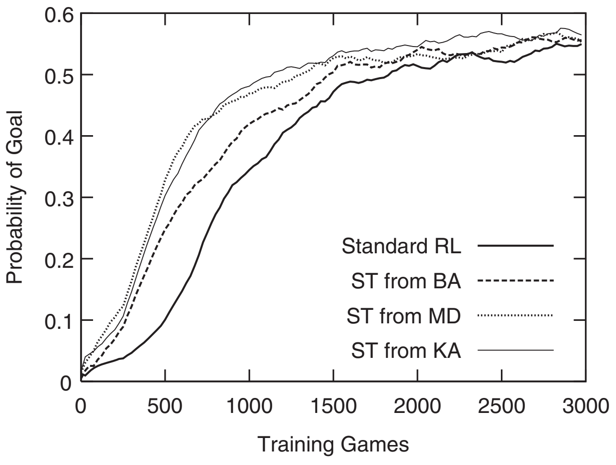


Fig. 6. Probability of scoring a goal while training in 3-on-2 BreakAway with standard RL and skill transfer (ST) from 2-on-1 BreakAway (BA), 3-on-2 MoveDownfield (MD) and 3-on-2 KeepAway (KA)

We took the appropriate subset of user advice from this set:

```
IF      distBetween(a0, GoalRight) < 10 AND
        angleDefinedBy(GoalRight, a0, goalie) > 40
THEN prefer shoot(GoalRight) over other actions

IF      distBetween(a0, GoalLeft) < 10 AND
        angleDefinedBy(GoalLeft, a0, goalie) > 40
THEN prefer shoot(GoalLeft) over other actions

IF      distBetween(a0, goalCenter) > 10
THEN prefer moveAhead over other actions
```

Note that this user advice is not tuned. By adjusting the numerical values in the rules above, it is possible to improve the performance further. However, we assume that users will only give approximate advice.

To give an example of the skill concepts learned, the following is an example rule for *pass* that our skill transfer algorithm extracted from 3-on-2 MoveDownfield:

```
IF      distBetween(Teammate, fieldCenter) ≥ 6,
        distBetween(Teammate, minDistTaker(Teammate)) ≥ 8,
        angleDefinedBy(Teammate, a0, minAngleTaker(Teammate)) ≥ 41 AND
        angleDefinedBy(OtherTeammate, a0, minAngleTaker(OtherTeammate)) ≤ 23
THEN prefer pass(Teammate) over other actions
```

This rule specifies a minimum pass angle and an open distance around the receiving teammate. It also requires that the teammate not be too close to the center of the field and gives a maximum pass angle for the alternate teammate.

The following is an example rule for *shoot* extracted from 2-on-1 Break-Away:

```
IF      distBetween(a0, goalCenter) ≥ 6,
        angleDefinedBy(GoalPart, a0, goalie) ≥ 52,
        distBetween(a0, oppositePart(GoalPart)) ≥ 6,
        angleDefinedBy(oppositePart(GoalPart), a0, goalie) ≤ 33 AND
        angleDefinedBy(goalCenter, a0, goalie) ≥ 28
THEN prefer shoot(GoalPart) over other actions
```

This rule requires a large open shot angle, a minimum distance to the goal, and angle constraints that restrict the goalie's position to a small area.

The results in Fig. 6 show that skill transfer can have a large overall positive impact in most transfer scenarios. The skill-transfer method of rule extraction not only produces more understandable rules than policy transfer, but also leads to better performance in the target task.

6 Summary and Open Problems

Rule extraction can have practical applications beyond explaining a machine-generated solution to humans. Transfer learning, in which the solution to one task is used while learning a related task, is one such application. The transfer learning framework could be also viewed as a way to evaluate rulesets based on their ability to improve learning in a related task.

This chapter discussed ways to transfer rules between SVM-based reinforcement learning tasks. In this context, it is more important to obtain general rules that express the basic skill concepts than to describe the specifics of the source-task model. Therefore, learning by observing behavior is more effective than building rules from complex Q-functions. The use of inductive logic programming is also beneficial because it allows rules to employ first-order logic, which makes them more general.

We use extracted rules as advice for a target task. With an SVM-based reinforcement learner, there is a straightforward method of incorporating advice as a soft constraint. This allows the rules to improve learning in the target task to the extent that they are relevant, but also provides protection against negative transfer effects.

There are several open problems in this area, in both the rule-extraction and advice-taking steps. One is to develop advice-taking methods for relational reinforcement learning (RRL), so that first-order advice can be applied directly without propositionalizing it first. Another is to extract rules that describe multiple-step plans rather than single-step action choices, which might capture more information from the source task than the current approaches do.

The research referenced in this chapter was supported by DARPA grant HR0011-04-1-0007.

References

1. R. Andrews, J. Diederich, and A. Tickle. A survey and critique of techniques for extracting rules from trained artificial neural networks. In *Knowledge Based Systems*, 1995.
2. J. Clouse and P. Utgoff. A teaching method for reinforcement learning. In *Proceedings of the 9th International Conference on Machine Learning*, 1992.
3. N. Cristianini and J. Shawe-Taylor. *An Introduction to Support Vector Machines and Other Kernel-Based Learning Methods*. Cambridge University Press, 2000.
4. K. Driessens and S. Dzeroski. Integrating experimentation and guidance in relational reinforcement learning. In *Proceedings of the 19th International Conference on Machine Learning*, 2002.
5. G. Kuhlmann, P. Stone, R. Mooney, and J. Shavlik. Guiding a reinforcement learner with natural language advice: Initial results in RoboCup soccer. In *AAAI Workshop on Supervisory Control of Learning and Adaptive Systems*, 2004.

6. R. Maclin, J. Shavlik, L. Torrey, and T. Walker. Knowledge-based support vector regression for reinforcement learning. In *IJCAI Workshop on Reasoning, Representation, and Learning in Computer Games*, 2005.
7. R. Maclin, J. Shavlik, L. Torrey, T. Walker, and E. Wild. Giving advice about preferred actions to reinforcement learners via knowledge-based kernel regression. In *Proceedings of the 20th National Conference on Artificial Intelligence*, 2005.
8. O. Mangasarian, J. Shavlik, and E. Wild. Knowledge-based kernel approximation. *Journal of Machine Learning Research 5*, pages 1127–1141, 2004.
9. S. Muggleton and L. De Raedt. Inductive logic programming: Theory and methods. *Journal of Logic Programming 19,20*, pages 629–679, 1994.
10. I. Noda, H. Matsubara, K. Hiraki, and I. Frank. Soccer server: A tool for research on multiagent systems. *Applied Artificial Intelligence*, 12:233–250, 1998.
11. V. Soni and S. Singh. Using homomorphisms to transfer options across continuous reinforcement learning domains. In *Proceedings of the 21st National Conference on Artificial Intelligence*, 2006.
12. A. Srinivasan. *The Aleph Manual*, 2001.
13. P. Stone and R. Sutton. Scaling reinforcement learning toward RoboCup soccer. In *Proceedings of the 18th International Conference on Machine Learning*, 2001.
14. R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.
15. M. Taylor, P. Stone, and Y. Liu. Value functions for RL-based behavior transfer: A comparative study. In *Proceedings of the 20th National Conference on Artificial Intelligence*, 2005.
16. L. Torrey, J. Shavlik, T. Walker, and R. Maclin. Advice-based transfer in reinforcement learning. Technical Report TR06-2, Machine Learning Research Group, U. Wisconsin-Madison, 2006.
17. L. Torrey, J. Shavlik, T. Walker, and R. Maclin. Relational skill transfer via advice taking. In *Proceedings of the 17th European Conference on Machine Learning*, 2006.
18. L. Torrey, T. Walker, J. Shavlik, and R. Maclin. Using advice to transfer knowledge acquired in one reinforcement learning task to another. In *Proceedings of the 16th European Conference on Machine Learning*, 2005.
19. C. Watkins. Learning from delayed rewards. Technical Report PhD Thesis, University of Cambridge, Psychology Dept., 1989.

Rule Extraction from Linear Support Vector Machines via Mathematical Programming

Glenn Fung[1], Sathyakama Sandilya[2], and R. Bharat Rao[1]

[1] Siemens Medical Solutions, USA Glenn.fung,Bharat.Rao@siemens.com
[2] Merrill Lynch, UK sandilya@alumni.princeton.edu

Summary. We describe an algorithm for converting linear support vector machines SVM and any other arbitrary hyperplane-based linear classifiers into a set of non-overlapping rules that, unlike the original classifier, can be easily interpreted by humans.

Each iteration of the rule extraction algorithm is formulated as a constrained optimization problem that is computationally inexpensive to solve. We discuss various properties of the algorithm and provide proof of convergence for two different optimization criteria. We demonstrate the performance and the speed of the algorithm on linear classifiers learned from real-world datasets, including a medical dataset on detection of lung cancer from medical images.

The ability to convert SVMs and other "black-box" classifiers into a set of human-understandable rules, is critical not only for physician acceptance, but also for reducing the regulatory barrier for medical-decision support systems based on such classifiers.

We also present some variations and extensions of the proposed mathematical programming formulations for rule extraction.

1 Introduction

Support Vector Machines (SVMs) [13, 24] and other linear classifiers are popular methods for building hyperplane-based classifiers from data sets, and have been shown to have excellent generalization performance in a variety of applications. These classifiers, however, are hard to interpret by humans.

For instance, when an unlabeled example is classified by the linear classifier as positive or negative, the only explanation that can be provided is that some linear weighted sum of the variables of the example are lower (higher) than some threshold; such an explanation is completely non-intuitive to human experts.

Humans are more comfortable dealing with rules that can be expressed as a hypercube with axis-parallel surfaces in the variable space.

Previous work [19, 22] and more recent work [11] included rule extraction for neural networks but very few work has been done to extract rules from SVMs or any other kind of hyperplane-based classifier. Recently Nunez et al [17] proposed a method to extract rules from an SVM classifier which involves applying a clustering algorithm first to identify groups that later define the rules to be obtained.

We propose a methodology for converting any linear classifier into a set of such non-overlapping rules.

This rule set is (asymptotically) equivalent to the original linear classifier, covers most of the training examples in the hyperplane halfspace. Unlike [17] our method does not require computationally expensive data preprocessing steps (as clustering) and the rule extraction is done in a very fast manner, typically it takes less than a second to extract rules from SVM's trained on thousands of samples.

Our algorithm does not required anything more complicated that solving simple linear programming problems in $2n$ variables where n is the number of input features (after feature selection).

In the next section we briefly discuss the medical relevance of this research. The ability to provide explanations of decisions reached by "black-box" classifiers is not only important for physician acceptance, but it is also a vital step in potentially reducing the regulatory requirements for introducing a medical decision-support system based on such a classifier into clinical practice. Section 3 then describes the commonly used linear support vector machine classifier and gives a linear program for it.

Section 4 provides our rule extraction algorithm; each iteration of the rule extraction algorithm is formulated as one of two possible optimization problems based on different "optimal" rule criteria. The first formulation, which seeks to maximize the volume covered by each rule, is a constrained nonlinear optimization problem whose solution can be found by obtaining the closed form solution of a relaxed associated unconstrained problem.

The second formulation, which maximizes the number of samples covered by each rule, requires us to solve a linear programming problem. In Sect. 5 we discuss finite termination and convergence conditions for our algorithm.

Section 6 summarizes our results on four publicly available datasets, and an additional medical dataset from our previous work [3] in building a CAD system to detect lung cancer from computed tomography volumes. In Sect. 7 we present and discuss several possible extensions for the mathematical programming formulations proposed in this paper. We end with some thoughts on further extensions and applications.

1.1 About Notation

We now describe the notation used in this paper. The notation $A \in R^{m \times n}$ will signify a real $m \times n$ matrix. For such a matrix, A' will denote the transpose of A and A_i will denote the i-th row of A. All vectors will be column vectors. For

$x \in R^n$, $\|x\|_p$ denotes the p-norm, $p = 1, 2, \infty$. A vector of ones in a real space of arbitrary dimension will be denoted by e. Thus, for $e \in R^m$ and $y \in R^m$, $e'y$ is the sum of the components of y. A vector of zeros in a real space of arbitrary dimension will be denoted by 0.

A *separating hyperplane*, with respect to two given point sets $\mathcal{A}$ and $\mathcal{B}$, is a plane that attempts to separate R^n into two halfspaces such that each open halfspace contains points mostly of $\mathcal{A}$ or $\mathcal{B}$. A *bounding plane* to the set $\mathcal{A}$ is a plane that places $\mathcal{A}$ in one of the two closed halfspaces that the plane generates. The symbol $\wedge$ will denote the logical "and" and the symbol $\vee$ will denote the logical "or". The abbreviation "s.t." stands for "such that". For a vector $x \in R^n$, the sign function $sign(x)$ is defined as $sign(x)_i = 1$ if $x_i > 0$ else $sign(x)_i = -1$ if $x_i \leq 0$, for $i = 1, \ldots, n$.

2 Medical Relevance

From the earliest days of computing, physicians and scientists have explored the use of artificial intelligence systems in medicine [20]. A long-standing area of research has been building *computer-aided diagnosis* (CAD) systems for the automated interpretation and analysis of medical images [18]. Despite the demonstrated success of many such systems in research labs and clinical settings, these systems were not widely used, or even available, in clinical practice. The primary barrier to entry in the United States is the reluctance of the US Government to allow the use of "black box" systems that could influence patient treatment.

Although the Food and Drug Administration (FDA) has recently granted approval for CAD systems based on "black-box" classifiers [21], the barrier to entry remains very high.

These systems may only be used as "second-readers", to offer advice after the initial physician diagnosis.

More significantly, these CAD systems must receive pre-market approval (PMA). A PMA is equivalent to a complete clinical trial (similar to the ones used for new drugs), where the CAD system must demonstrate statistically significant improvement in diagnostic performance when used by physicians on a large number of completely new cases. This is a obviously a key area of research in CAD, but not the focus of this paper. The FDA has indicated that the barrier to entry for CAD systems that are able to explain their conclusions, could be significantly lowered. Note, this will not lower the barrier in terms of generalization performance on unseen cases, but the FDA is potentially willing to consider using performance on retrospective or previously seen cases and significantly reduce the number of cases needed for a prospective clinical trial. This is critical, because a full-blown clinical trial can add several years delay to the release of a CAD system into general clinical practice.

Much research in the field of artificial intelligence, and now knowledge discovery and data mining has focused on the endowing systems with the ability to explain their reasoning, both to make the consultation more acceptable to the user, and to help the human expert more easily identify errors in the conclusion reached by the system [4]. On the other hand, when building classifiers from (medical) data sets, the best performance is often achieved by "black-box" systems, such as, Support Vector Machines (SVMs). The research described in this paper will allow us to use the superior generalization performance of SVM's and other linear hyperplane-based classifiers in CAD system, and using the explanation features of the rule extraction algorithm to reduce the regulatory requirements for market introduction of such systems into daily clinical practice.

3 Sparse Hyperplane Classifiers: 1-Norm Support Vector Machines

We consider the problem of classifying m points in the n-dimensional input space R^n, represented by the $m \times n$ matrix A, according to membership of each point A_i in the class A_+ or A_- as specified by a given $m \times m$ diagonal matrix D with plus ones or minus ones along its diagonal. For this problem, depicted in Fig. 1, the linear programming support vector machine [5,13] with a linear kernel (this is a variant of the standard SVM [6,24]) is given by the following linear program with parameter $\nu > 0$:

$$\begin{array}{rl} \min\limits_{(w,\gamma,y)\in R^{n+1+m}} & \nu e'y + \|w\|_1 \\ \text{s.t.} & D(Aw - e\gamma) + y \geq e \\ & y \geq 0, \end{array} \tag{1}$$

where $\|\cdot\|_1$ denotes the 1-norm as defined in the introduction. That this problem is indeed a linear program, can be easily seen from the equivalent formulation:

$$\begin{array}{rl} \min\limits_{(w,\gamma,y,t)\in R^{n+1+m}} & \nu e'y + e't \\ \text{s.t.} & D(Aw - e\gamma) + y \geq e \\ & t \geq w \geq -t \\ & y \geq 0. \end{array} \tag{2}$$

For economy of notation we shall use the first formulation (1) with the understanding that computational implementation is via (2).

The 1-norm term $\|w\|_1$ in (1), which is half the reciprocal of the distance $\frac{2}{\|w\|_1}$ measured using the ∞-norm distance [12] between the two bounding planes (see Fig. 1), maximizes this distance, often called the "margin". then two planes bound the two classes with a "soft margin" (i.e. bound approximately with some error) determined by the nonnegative error variable y, that is:

$$\begin{array}{l} A_i w + y_i \geq \gamma + 1, \text{ for } D_{ii} = 1, \\ A_i w - y_i \leq \gamma - 1, \text{ for } D_{ii} = -1. \end{array} \tag{3}$$

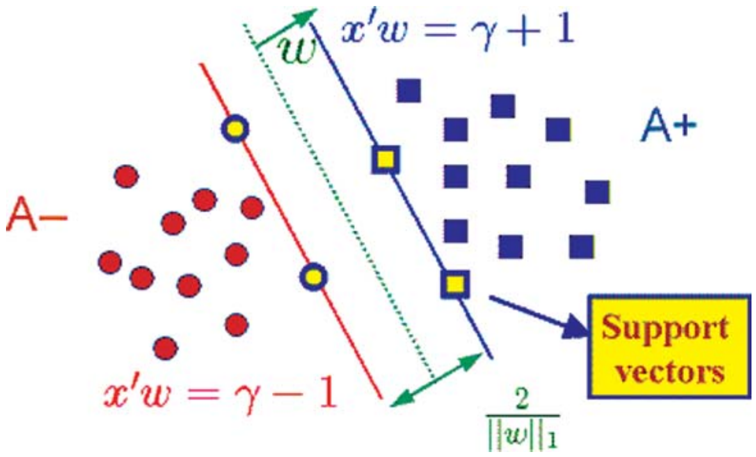


Fig. 1. The LP–SVM classifier in the w-space of R^n. The plane of equation (3) approximately separating points in $A+$ from points in $A-$

The 1-norm of the error variable y is minimized parametrically with weight ν in (1), resulting in an approximate separating plane. This plane classifies data as follows:

$$sign(x'w - \gamma\ ) \begin{cases} = 1, & \text{then } x \in A_+, \\ = -1, & \text{then } x \in A_-, \end{cases} \tag{4}$$

where $sign(\cdot)$ is the sign function defined in the Introduction. Empirical evidence [5] indicates that the 1-norm formulation has the advantage of generating very sparse solutions. This results in the normal w to the separating plane $x'w = \gamma$ having many zero components, which implies that many input space features do not play a role in determining the linear classifier. This makes this approach suitable for feature selection in classification problems.

Since our rule extraction algorithm depends directly on the features used by the hyperplane classifier, sparser normal vectors w will lead to rules depending on a fewer number of features.

4 Rule Extraction from Hyperplane Classifiers

In the previous section, we described a linear programming SVM formulation to generate hyperplane classifiers. We now present an algorithm to extract rules of the form:

$$\wedge_{i=1}^{n} l_i \leq x_i < u_i \Longrightarrow \textit{ class membership}$$

to approximate these classifiers. Note that every rule form defined above defines an hypercube in the n dimensional space with edges parallel to the axis. Rule of this form are very intuitive and can be easily interpreted by humans.

Our rule extraction approach can be applied to any linear classifier regardless of the algorithm or criteria used to construct the classifier, including Linear Fisher Discriminant (LFD) [15], Least squares SVMs (LS–SVMs) [23] or Proximal SVMs (PSVM) [7]. Denote by $P_{-}(w, \gamma, I)$ the problem of constructing rules for the classifier for the region:

$$I = \{x \text{ s.t. } w'x < \gamma,\, l_i \leq x_i \leq u_i,\ 1 \leq i \leq n\}$$

based on the classification hyperplane $w'x = \gamma$ obtained by solving problem (1). Note that the problem of rule extraction $P_{+}(w, \gamma, I')$ where

$$I' = \{x \text{ s.t. } w'x > \gamma,\, l_i \leq x_i \leq u_i,\ 1 \leq i \leq n\}$$

is the same as $P_{-}(-w, -\gamma, I)$. We now establish that this is equivalent to solving the problem with positive hyperplane coefficients, $\gamma = 1$ and the feature domain being the unit hypercube. Consider a diagonal matrix T constructed in the following way:

$$T_{ii} = \frac{sign(w_i)}{u_i - l_i},\ \ i \in \{1, \ldots, n\} \tag{5}$$

and a vector b with components $b = \{u_i \text{ if } w_i < 0,\ l_i \text{ if } w_i > 0\}$.

We now define a transformation of coordinates such that $y = T(x - b)$. Note that

$$\begin{aligned} w_i > 0 &\Rightarrow 0 \leq y_i = T_{ii}(x_i - l_i) = \frac{x_i - l_i}{u_i - l_i} \leq 1 \\ w_i < 0 &\Rightarrow 0 \leq y_i = T_{ii}(x_i - u_i) = \frac{-(x_i - u_i)}{u_i - l_i} = \frac{(u_i - x_i)}{u_i - l_i} \leq 1 \end{aligned} \tag{6}$$

hence, I is transformed to $[0,1]^n$. Furthermore $x = T^{-1}y + b$, and hence, the hyperplane of interest becomes

$$w'T^{-1}y = \gamma - w'b$$

which is equivalent to:

$$\tilde{w}y = \left(\frac{w'T^{-1}}{\gamma - w'b}\right) y = 1 \tag{7}$$

Thus the problem becomes $P_{-}(\tilde{w}, 1, I_0)$ in the new domain, where $I_0 = [0,1]^n$. In mapping the original problem to the unit hypercube the measure of volume Although the objective function being optimized is different, it is merely a scaled version of the original problem, and thus the optimum remains the same.

Note that the components of $\tilde{w}$ are positive as $w'b < \gamma$ and $w_i T_{ii} > 0$.

For the rest of this paper we will concentrate in finding rules with the following properties:

– The hypercube defined by the extracted rule

$$\wedge_{i=1}^{n} l_i \leq x_i < u_i$$

is a subset of the bounded region $I = \{x \text{ s.t. } w'x < \gamma\}$.

– The resulting hypercube cube defined by the extracted rule contains one vertex that lies in the separating hyperplane $w'x - \gamma = 0$. This assumption allows to obtain set of disjoint rules that are easy to generate and simplifies the problem considerable.

Figure 2 illustrates an example in two dimensions where the halfspace $w'x < \gamma$ is almost totally covered by rules represented by hypercubes with a vertex in the hyperplane $w'x - \gamma = 0$.

Given a region I we can define the "optimal" rule according to different criteria, Next we present two of them.

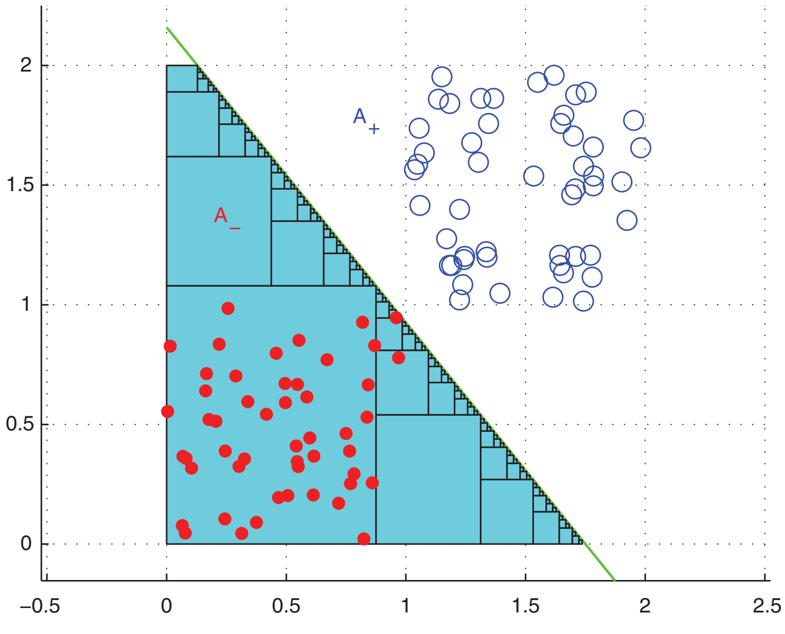


Fig. 2. Two-dimensional example where the non-overlapping rules covering the halfspace ($\{x \text{ s.t. } w'x < \gamma\}$) are represented as cyan rectangles

4.1 Volume Maximization Criteria

An optimal rule can be defined as the rule that covers the hypercube with axis-parallel faces with the largest possible volume. Since the log function is strictly increasing, $\arg\max f(x) = \arg\max \log(f(x))$, we can find the rule that maximizes the log of the volume of the region that it encloses (instead of the volume). Assuming that the linear transformation T was already applied and that one corner of the region lies on the hyperplane, this rule can be found by solving the following problem:

$$\max_{x \in R^n} \log(\prod_{i=1}^{n} x_i) \quad \text{s.t.} \quad \sum_{i=1}^{n} w_i x_i = \gamma, 0 \leq x \leq 1. \tag{8}$$

The Lagrangian function for this nonlinear constrained optimization problem is:

$$L(x, \lambda, \theta) = \log(\prod_{i=1}^{n} x_i) - \lambda(w'x - \gamma) - \sum_{i=1}^{n} \theta_i (x - 1) + \sum_{i=1}^{n} \delta_i x. \tag{9}$$

The KKT optimality conditions for problem (8) are given by:

$$\begin{aligned} \frac{1}{x_i} - \lambda w_i - \theta_i + \delta_i &= 0 \ \forall i \in \{1, \ldots, n\}, \\ w'x &= \gamma, \\ 0 \leq x_i \leq 1, \ \lambda \geq 0, \ \theta_i \geq 0, \ \delta_i &\geq 0 \ \forall i \in \{1, \ldots, n\}, \\ \theta_i (x_i - 1) &= 0 \ \forall i \in \{1, \ldots, n\}, \\ \delta_i x_i &= 0 \ \forall i \in \{1, \ldots, n\}. \end{aligned} \tag{10}$$

In order to find a solution for problem (8) we will first consider solutions for the relaxed equality constrained problem:

$$\max_{x \in R^n} \log(\prod_{i=1}^{n} x_i) \quad \text{s.t.} \quad \sum_{i=1}^{n} w_i x_i = \gamma. \tag{11}$$

The KKT optimality conditions for problem (11) (which are very similar to the KKT conditions of problem (8)) are given by:

$$\frac{1}{x_i} - \lambda w_i = 0, \ i \in \{1, \ldots, n\}, \quad wx - \gamma = 0. \tag{12}$$

From the KKT optimality conditions (12) we obtained the following closed form solutions for the relaxed optimization problem:

$$\tilde{x_i} = \frac{1}{\lambda w_i} = \frac{\gamma}{n w_i} \ i \in \{1, \ldots, n\}, \quad \tilde{\lambda} = \frac{n}{\gamma}. \tag{13}$$

A solution x^* of the original optimization problem (8) can be obtained from the solution (13). Let's define x^* as follows:

$$x_i^* = \left\{ \begin{array}{ll} \frac{1}{\lambda^* w_i} \text{ if} & \tilde{x}_i \leq 1, \ i \in \{1, \dots, n\} \\ \quad 1 \ \text{ otherwise} & \end{array} \right\}, \tag{14}$$

where,

$$\lambda^* = \frac{n_I}{\gamma - \sum_{\{i \in A\}} w_i} \tag{15}$$

where $A = \{i \, | \tilde{x}_i > 1\}$ and n_I is $n - |A|$ with λ^* defined as above we have that:

$$\begin{aligned} wx^* - \gamma = \sum_{i=1}^{n} w_i x_i^* &= \sum_{i \in I} w_i x_i^* + \sum_{i \in A} w_i x_i^* - \gamma \\ &= \textstyle\sum_{i \in I} \frac{w_i}{\lambda^* w_i} + \displaystyle\sum_{i \in A} w_i - \gamma \\ &= \frac{n_I}{\lambda^*} + \sum_{i \in A} w_i - \gamma \\ &= n_I \frac{\gamma - \sum_{i \in A} w_i}{n_I} + \sum_{i \in A} w_i - \gamma \\ &= \gamma - \gamma = 0, \end{aligned} \tag{16}$$

if $0 \leq x_i^* \leq 1, \forall i \in \{1, \dots, n\}$, then x^* is the optimal solution for problem 8, otherwise define $\tilde{x} = x^*$ and recalculate x^* until $0 \leq x_i^* \leq 1, \forall i \in \{1, \dots, n\}$. This iterative procedure can be seen as a gradient projection method for which convergence is well established [1, 2].

4.2 Point Coverage Maximization Criteria

Another optimal rule can be defined as the rule that covers the hypercube with axis-parallel faces with that contains the largest possible number of training points in the halfspace. Given a transformed problem $P_-(\tilde{w}, 1, I_0)$, we want to find x^* such that $w'x^* - \gamma = 0$ and $|C|$ (cardinality of C)is maximal, where:

$$C = (A_- \cap \{x | \ w'x < 1\}) \cap \{x | \ 0 \leq x \leq x^*\}.$$

The following Linear programming formulation is an approximation to this problem:

$$\begin{array}{lll} \min\limits_{x,y} & e'y & \\ \text{s.t.} & w'x & = 1 \\ & A_{.i} - ey_i \leq x_i & \forall i \in \{1, \dots, n\}, \\ & & 0 \leq x \leq 1, \\ & & y \geq 0. \end{array} \tag{17}$$

Note that the variable $y \geq 0$ acts as a slack or error variable that is minimized in order in order for the rule to cover the largest possible amount of points.

We can now use either one of the optimal rule definitions described in Sects. 4.1 and 4.2 to propose an iterative procedure that extract as many rules as we require to describe adequately the region of interest. We first demonstrate that in a n-dimensional feature space, extracting one such a rule results in n new similar problems to solve. Let the first rule extracted for the transformed problem $P_-(\tilde{w}, 1, I_0)$ be $\wedge_{i=1}^{n}(0 \leq x_i < x_i^*)$. The remaining volume on this side of the hyperplane that is not covered, is the union of n nonintersecting regions similar to the original region, namely

$$I_i = \begin{cases} x \in R^n, \text{s.t.} & 0 \leq x_j < x_j^* \ \forall 1 \leq j < i, \\ & x_i^* \leq x_i < 1, \\ & 0 \leq x_j < 1 \ \forall j > i, \end{cases} \tag{18}$$

that is, the rule inequalities for the first $i-1$ components of x are satisfied, the inequality that relates to the i^{th} component is not satisfied, and the rest are free. Consider i, j with $j > i$. For each $x \in I_j$, we have $0 \leq x_i < x_i^*$ and for each $x \in I_i$, we have $x_i^* \leq x_i < 1$. Hence, I_i are nonintersecting, and the rules that we arrive at for each I_i will be "independent". Now we extract the optimal rule for each of these regions that contains a training data point using a depth first search. Note that the problem for I_i is $P_-(\tilde{w}, 1, I_i)$, and we can now use the same transformation as described in (5)–(7) to transform each of the n subproblems $P_-(\tilde{w}, 1, I_i)$ to problems equivalent to the original problem $P_-(\tilde{w}, 1, I_0)$.

Next, we state our algorithm to obtain a set of rules R that cover all the training points belonging to A_- such that $w'x < \gamma$. Let R be the set containing all the extracted rules, and U be the set containing the indices of the points uncovered by the rules in R. R and U are initialized to $\emptyset$ and A_- respectively, d_{max} (which bounds the maximum depth of the depth first search, typically less than 20) is assigned, and w, γ are obtained by solving the LP-SVM (1) before **ExtractRules** is invoked for the first time.

Algorithm 4.1 *ExtractRules*(w, γ, I, d): Algorithm for rule extraction from linear classifiers.

Given parameters d_{max} that bounds the depth on the depth-first search, (typically $d_{max} < 20$), Initial hyperplane parameters (w, γ) obtained by solving the linear programming SVM formulation as described in (1) and $I = \{x \text{ s.t. } w'x < \gamma,\ l_i \leq x_i \leq u_i,\ i \in \{1, \ldots, n\}\}$. Let $0 \leq x_i \leq 1,\ \ i \in \{1, \ldots, n\}$, and $w_i \geq 0,\ \ i \in \{1, \ldots, n\}$ be obtained by mapping the original data.

```
1. If $d = d_{max}$, stop.
2. Transform problem $P_-(w, \gamma, I)$ into $P_-(\tilde{w}, 1, I_0)$ using the linear transformation described in Sect. 5, (5)–(10).
3. Obtain $y^*$ by solving problem $P_-(\tilde{w}, 1, I_0)$ using either (14)–(15) or (17).
4. Calculate $x^* = T^{-1}y^* + b$, get new rules $\tilde{R}(x^*)$, update $R \leftarrow R \cup \tilde{R}(x^*)$.
5. Let $C = \{x \in U \text{ st. } \tilde{R}(x^*) \text{ is true}\} = U \cap \tilde{R}(x^*)$, this is, a set containing the indices of the points in $U$ that are covered by the new obtained rule.
```

```
6. Update $U \leftarrow U - C$. If $U = \emptyset$, stop. Else $d \leftarrow d + 1$.
7. for $k = 1$ to $n$ do
     • Calculate $\hat{I}_k = T^{-1} I_k + b$. If $U \cap \hat{I}_k \neq \emptyset$ apply recursively
       ExtractRules($w, \gamma, \hat{I}_k, d$), where $\hat{I}_k$ is one of the $n$ remaining regions
       of interest uncovered by rule $\tilde{R}(x^*)$ as defined in (18).
```

5 Algorithm Convergence Properties

We now derive the rate at which the volume covered by the rules extracted for $P(w, 1, I_0)$ converges to the total volume of the region of interest.

Lemma 5.1 Volume of the region $\{x \text{ s.t. } w'x < \gamma, x_i \geq 0\}$
The volume of the region $\{x \text{ s.t. } w'x < \gamma, x_i \geq 0\}$ *is*

$$V_n(w, \gamma) = \frac{\prod_{i=1}^{n} \frac{\gamma}{w_i}}{n!}.$$

Proof. We show this by induction. For $n = 2$, this is the area of a right-angled triangle with sides γ/w_1 and γ/w_2, which is $\gamma^2/2w_1w_2$. Now, assume that this is true for $n = k$.

$$\begin{aligned}
V_{k+1}(w, \gamma) &= \int_0^{\gamma/w_1} \dots \int_0^{(\gamma - w_1x_1 - \dots - w_kx_k)/w_{k+1}} dx_1 dx_2 \dots dx_{k+1} \\
&= \int_0^{\gamma/w_1} dx_1 \int_0^{(\gamma - w_1x_1)/w_2} dx_2 \dots \int_0^{(\gamma - w_1x_1 \dots - w_kx_k)/w_{k+1}} dx_{k+1} \\
&= \int_0^{\gamma/w_1} dx_1 V_k(w_{-1}, \gamma - w_1x_1) \\
&= \int_0^{\gamma/w_1} dx_1 \frac{1}{k!} \prod_{i=2}^{k+1} \frac{\gamma - w_1x_1}{w_i} \\
&= \frac{1}{k!} \prod_{i=2}^{k+1} \frac{1}{w_i} \int_0^{\frac{\gamma}{w_1}} dx_1 (\gamma - w_1x_1)^k \\
&= \frac{1}{k!} \left(\prod_{i=2}^{k+1} \frac{1}{w_i}\right) \frac{\gamma^{k+1}}{(k+1)w_1} = \frac{1}{(k+1)!} \prod_{i=1}^{k+1} \frac{\gamma}{w_i}
\end{aligned}$$

where w_{-i} contains all components of w except the ith. □

Lemma 5.2 Volume bound
For any $S \subseteq \{1, 2, \dots, n\}$, *the volume of a region defined by* $w'x < 1$ *and* $0 \leq x_i < 1, 1 \leq i \leq n$ *is bounded by*

$$\frac{1}{|S|!} \prod_{i \in S} \frac{1}{w_i}$$

Proof. We can assume without loss of generality that S is $\{1, 2, \ldots, k\}$ (if it is not, the coordinates may be permuted so that it is). The volume of interest, say V is given by

$$\begin{aligned}
V = &\int_0^{\min(1,1/w_1)} dx_1 \int_0^{\min(1,(1-w_1x_1)/w_2)} dx_2 \ldots \\
&\ldots \int_0^{\min(1,(1-w_1x_1-\ldots-w_{n-1}x_{n-1})/w_n)} dx_n \\
&\leq \int_0^{\min(1,1/w_1)} \ldots \int_0^{\min(1,(1-w_1x_1-\ldots-w_{k-1}x_{k-1})/w_k)} \ldots \\
&\ldots \int_0^1 \ldots \int_0^1 dx_1 dx_2 \ldots dx_n \\
&\leq \int_0^{1/w_1} \ldots \int_0^{(1-w_1x_1-\ldots-w_{k-1}x_{k-1})/w_k} dx_1 dx_2 \ldots dx_k \\
= &\frac{1}{k!} \prod_{i=1}^{k} \frac{1}{w_i},
\end{aligned}$$

where the first two inequalities are because the upper limit in the integral is replaced by an upper bound, and the last equality comes from the previous lemma with $\gamma = 1$. □

Lemma 5.3 Volume Coverage
At each "stage", the algorithm covers at least $\alpha = \frac{n!}{n^n}$ of the volume yet to be covered. Hence, the volume remaining after k stages is at most $(1-\alpha)^k V_0$.

Proof. The volume covered by the rule is given by

$$\begin{aligned}
V_{rule} &= \prod_{i=1}^{n} x_i^* = \Big(\prod_{i \notin A} \frac{1}{\lambda^* w_i}\Big)\Big(\prod_{i \in A} 1\Big) \\
&= \prod_{i \notin A} \frac{1}{w_i} \frac{1 - \sum\limits_{i \in A} w_i}{n - |A|} \\
&\geq \prod_{i \notin A} \frac{1}{w_i} \frac{1 - |A|/n}{n - |A|} \\
&= \prod_{i \notin A} \frac{1}{w_i} \frac{1}{n},
\end{aligned}$$

where A as before is the set of active constraints, and the inequality above comes from the fact that for $i \in A$, $\frac{1}{nw_i} > 1$ (the original solution to the relaxed problem violates the constraints). Using the result of the previous lemma, and setting $S = \{1, \ldots, n\} \backslash A$, we have

$$\frac{V_{rule}}{V_{total}} \geq \frac{\prod_{i \notin A} \dfrac{1}{w_i}\dfrac{1}{n}}{\frac{1}{(n-|A|)!}\prod_{i \notin A}\frac{1}{w_i}} = \frac{(n-|A|)!}{n^{n-|A|}} \geq \frac{n!}{n^n}$$

the last inequality arises because the bound is monotonically increasing in $|A|$ with it being the smallest when $|A| = 0$. $\square$

Lemma 5.4 *At each stage, the algorithm reduces the largest distance from an interior point yet to be covered to the separating hyperplane by a factor of* $1 - 1/n$.

Proof. We establish the lemma for one stage of $P(w, 1, I_0)$ (a simple scaling argument would extend it to a general γ and I, and hence to further stages of the problem as well). The largest distance from the plane in I_0

$$d^0_{max} = \sup_{x \in I_0, w'x<1} (1 - w'x)/||w|| = 1/||w||.$$

In region I_i, as $x_i \geq x_i^*$ and $w'x$ is monotonically increasing in each coordinate

$$d^i_{max} = \sup_{x \in I_i, w'x<1} (1 - w'x)/||w|| = (1 - w_i x_i *)/||w||,$$

when $i \in A$, then I_i has no interior points. When $i \notin A$, $\tilde{x_i} = w_i x_i^* = 1/n$. Hence,

$$d^i_{max} = (1 - 1/n)/||w|| = (1 - 1/n)d^0_{max} \quad \square$$

Theorem 5.5 Finite termination *After extracting t rules, the remaining volume is at most* $(1-\alpha)^{\log_n t-1}$ *of the original volume. Moreover, the rule extraction algorithm covers in finite time any dataset that has all points in the interior of I.*

Proof. As described before, each rule extraction leads to n further "subproblems". Hence, the number of rules to be extracted in stage k is n^{k-1}, and the number of rules extracted up to and including stage k is $\frac{n^k-1}{n-1}$. Hence, if t rules have been extracted and k stages are complete,

$$t < \frac{n^{k+1}-1}{n-1} \Rightarrow t < n^{k+1} \Rightarrow k > \log_n t - 1.$$

Hence, at least $\log_n t - 1$ stages are complete, and hence, by a previous lemma, at most $(1-\alpha)^{\log_n t-1}$ of the volume remains (which converges to 0 as $t \to \infty$). Moreover, by the previous lemma we have that at the end of stage k,

$$d_{max} = (1 - 1/n)^k \gamma/||w||.$$

Hence, for a data point x, we have that x is covered when

$$(\gamma - w'x)/||w|| > (1 - 1/n)^k \gamma/||w||,$$

i.e. when

$$k \geq \log_{(1-1/n)} (1 - w'x/\gamma).$$

Hence, the entire data set A_- is covered when

$$k \geq \log_{(1-1/n)} (1 - \max_{x \in A_-} (w'x)/\gamma),$$

i.e., when

$$t = n^{1+\log_{(1-1/n)} (1-\max_{x \in A_-} (w'x)/\gamma)}.$$

We now use this to establish termination of the algorithm for a given data set in finite time. Let us assume the contrary, i.e. that there is a point $x^{\#}$ such that $w'x^{\#} < \gamma$ and it is not covered in the rule extraction process. By the previous lemma, we have that $y = x^{\#} + (\gamma - w'x^{\#})w/2||w||$ is not covered (as it is greater than x). Moreover, any point in the hypercuboid $x_i^{\#} \leq x_i < y_i$ is not covered by the rules. Hence the volume of the uncovered region is at least $\prod_{i=1}^{n} (y_i - x_i^{\#})$, which is a contradiction of the previous part of the theorem. Hence, the point $x^{\#}$ gets covered after a finite number of iterations. □

6 Numerical Testing

To show the effectiveness of our rule extraction algorithm, we performed experiments in five real-world datasets. Three of the datasets are publicly available datasets from the UCI Machine Learning Repository [16]: Wisconsin Diagnosis Breast Cancer (WDBC), Ionosphere, and Cleveland heart. The fourth dataset is a dataset related to the nontraditional authorship attribution problem related to the federalist papers [10] and the fifth dataset is a dataset used for training in a computer aided detection (CAD) lung nodule detection algorithm, we refer to this set as the Lung CAD dataset.

Experiments for the five datasets were performed to test the capability of Algorithm 4.1 to cover training points correctly classified by the SVM hyperplane. For each experiment, we obtained a separating hyperplane using the 1−norm linear programming SVM (LP-SVM) formulation as described in (1). The state of the art optimization software CPLEX was used to solve the corresponding linear programming problems. Ten-fold cross validation was used as a tuning procedure to determine the SVM parameter ν. In All the experiments, the resulting hyperplane classifier was sparse, this means that the set $\{w_i \text{ s.t. } w_i \neq 0,\ 1 \leq i \leq n\}$ was "small", this was expected because of the effect of the 1−norm regularization term on the coefficients w_i.

Having a sparse hyperplane implies that the dimensionality of the training dataset can be reduced by discarding the features corresponding to $w_i = 0$ since they do not play any role in the classification.

Once the hyperplane was obtained we applied Algorithm 4.1 using one of the two criteria for optimal rules described in Sects. 4.1 and 4.2. The first criteria is based in finding rules that maximizes the volume of the region covered by the rule, we will refer to this variant of Algorithm 4.1 as Volume Maximization (**VM**). The second criteria is to find rules that attempt to cover a many points of the training set as possible. We will call this variant of Algorithm 4.1 Point Coverage Maximization (**PCM**).

Results for both VM and PCM are reported in Tables 1 and 2 including: total number of optimization problems solved, total execution time, total number of extracted rules and percentage of correctly classify points by the hyperplane that were covered by the extracted rules.

It is important to note that the results reported included only rules that covered more than one point. We considered that rules that covered only one point did not have any generalization capability and therefore were discarded. In general, the algorithm can be tuned to discard rules that do not cover enough points according to a number predefined by the user.

Empirical results on the five datasets as reported in Tables 1 and 2 show the effectiveness of both the VM and PCM variants of our proposed algorithm. In most cases our algorithms covered more of 90% of the training points using only a few rules. As was expected, the VM variant seems to solve more "easy" optimization problems and generate more rules. On the other hand, the PCM variant solved fewer optimizations problems (linear programming problems) but that were slightly harder to solve, generating fewer rules.

Note that Tables 1 and 2 appear at the end of this paper (the references). Next, we will discuses in more detail the results obtained for the WDBC dataset and the lungcad dataset since medical diagnosis applications is of special interest to us.

Table 1. Results using maximal area formulation, # of optimization problems solved, total execution time (in s), number of rules and % of correctly classified points covered are shown for both classes A_- and A_+ on six datasets

Data Set $m \times n, card(A_-), card(A_+)$ # of features	# prob. solved $\hat{A_-}$ $\hat{A_+}$	Time $\hat{A_-}$ $\hat{A_+}$	# of points $\hat{A_-}$ $\hat{A_+}$	# rules $\hat{A_-}$ $\hat{A_+}$	Coverage % $\hat{A_-}$ $\hat{A_+}$
Lung CAD	33	0.14	124	18	97.6
$274 \times 34, 137, 137$	43	0.17	102	20	100.0
5					
WPBC	53	0.23	214	28	100.0
$683 \times 9, 444, 239$	12	0.11	435	9	100.0
5					
Ionosphere	46	0.17	70	19	100.0
$351 \times 34, 225, 126$	29	0.19	224	11	100.0
6					
Cleveland	102	0.30	53	10	79.3
$297 \times 13, 214, 83$	68	0.20	195	22	98.4
6					
Federalist	22	0.17	50	4	90.0
$106 \times 70, 50, 56$	23	0.19	52	6	92.00
6					

Table 2. Results using the maximal coverage formulation, # of optimization problems solved, total execution time (in s), Number of rules and % of correctly classified points covered are shown for both classes A_- and A_+ on six datasets

Data Set $m \times n, card(A_-), card(A_+)$ # of features	# prob. solved $\hat{A_-}$ $\hat{A_+}$	Time $\hat{A_-}$ $\hat{A_+}$	# of points $\hat{A_-}$ $\hat{A_+}$	# rules $\hat{A_-}$ $\hat{A_+}$	Coverage % $\hat{A_-}$ $\hat{A_+}$
Lung CAD	5	0.09	124	4	98.4
$274 \times 34, 137, 137$	10	0.16	102	9	94.1
5					
WPBC	10	0.17	214	7	98.1
$683 \times 9, 444, 239$	3	0.14	435	3	99.8
5					
Ionosphere	11	0.17	70	7	87.2
$351 \times 34, 225, 126$	5	0.09	224	2	98.2
6					
Cleveland	32	0.33	53	7	81.1
$297 \times 13, 214, 83$	11	0.19	195	11	97.44
6					
Federalist	2	0.08	50	2	100.0
$106 \times 70, 50, 56$	13	0.25	52	11	96.2
6					

6.1 WDBC Dataset

The first experiment relates to the publicly available WDBC dataset that consists of 683 patient data. The classification task associated with this dataset is to diagnose breast masses based solely on a Fine Needle Aspiration (FNA). Doctors identified nine visually assessed characteristics or attributes of an FNA sample which they considered relevant to diagnosis (for more detail please refer to [14]).

After applying the LP–SVM algorithm and discarding the features corresponding to the $w_i = 0$, we ended up with a hyperplane classifier in five dimensions that achieved 95.0% tenfold testing set correctness.

After applying our **ExtractRules-PCM** algorithm to cover the 214 points in A_- correctly classified by the hyperplane we obtained a total of seven non empty, non-singleton rules that cover 99.8% of the points. Similarly we obtained a total of three non empty, non-singleton rules that cover 98.1% of points in A_+ correctly classified by the hyperplane. For example after using the fact that all the features values are integers between one and ten we obtained the following rule that covered 383 of the 435 positive training points:

$$\begin{aligned} (\mathit{Cell\ Size} \le 3) \ \wedge \ & (\mathit{Bare\ Nuclei} \le 1) \\ & \wedge\ (\mathit{Normal\ Nucleoli} \le 7) \\ \Rightarrow \ \mathit{mass\ is\ benign} & \end{aligned}$$

6.2 The Lung CAD Dataset

The second experiment relates to a set of data used in a computer aided detection (Lung CAD) system for pulmonary nodule detection on thin slice multidetector CT scans. The Lung CAD algorithm performs the following processing steps:

(a) Lung segmentation
(b) Candidate generation
(c) Feature calculation at each candidate location
(d) Classification
(e) Presenting CAD findings to a physician for review

The task of the candidate generation step, is to reduce the search space by quickly generating a list of suspicious locations for different types of nodules at a high sensitivity without considering the specificity. For this, shape based characteristics are used to generate a candidate list. For each candidate in the list a set of features is calculated. Those features are based on the intensity, the shape, the curvature, and the location. The goal of the last processing step is to increase the specificity without decreasing the sensitivity by pruning the list of candidates. For this, a classifier is used.

Our dataset consists on 274 candidates represented by 34 numerical features. Each datapoint corresponds to a candidate labeled as a nodule or not a nodule. The LP–SVM algorithm generated a classifier in only five features with 82.5% tenfold testing set correctness. Our **ExtractRules-PCM** algorithm extracted a total of ten nonempty, non-singleton rules that cover 94% of positive training points correctly classified for the hyperplane, similarly we obtained a total of only five nonempty, non-singleton rules that cover 98.4% of the points in A_+ correctly classified for the hyperplane.

7 Other Mathematical Programming Formulations

Next, we present and briefly discuss several natural extensions or variations of the mathematical programming formulations presented in this paper.

7.1 Conditioning Rules by Using Prior Knowledge

Let's illustrate this case by using an example, let's suppose that after solving formulation (2) for the dataset used in Sect. 6.1, we obtained a linear classifier that only depends on three features: *Cell Size*, *Bare Nuclei* and *Normal Nucleoli*. Suppose also that using doctor's knowledge we suspect that having the following condition *Cell Size* ≤ 2 is a good indicator of the mass to be benign, but the ranges of values of other variables needed to have a more solid implication or rule are unknown. This problem can be addressed by incorporating this kind of prior knowledge represented by a logical rule (*Cell*

Size ≤ 2 probably implies benign mass) in the form of linear constraints in formulation (17).

Let's define the set K the subset of indices of the features for which a threshold t_k is given and U the set of features for which these threshold are unknow (in formulation 17 we assume that $K = \emptyset$). similarly to Sect. 4.2 we assume that we are given a transformed problem $P_{-}(\tilde{w}, 1, I_0)$, but the case when we are given a transformed problem $P_{+}(\tilde{w}, 1, I_0)$ is analogous. We want to find $x^*_U \in \Re^{|U|}$ such that $w'x^* - \gamma = 0$ and $|C|$ is maximal. where, $x^* = x^*_U, (t_1, \ldots, t_{|K|}$ and C is defined in the following way:

$$C = (A_{-} \cap \{x|\ w'x < 1\}) \cap \{x|\ 0 \leq x_i \leq x^*_{Ui}\ \ i \in U \text{and } 0 \leq x_i \leq t_i\ \ i \in K\}$$

An approximate solution to this problem can be found by incorporating the following set of linear constraints to formulation (17);

$$x_i = t_i \forall i \in K$$

which after simplification is equivalent to the following linear program:

$$\begin{array}{lll}\min\limits_{x_U, y} & e'y & \\ \text{s.t.} & \sum_{i \in U} w_i x_i = 1 - \sum_{i \in K} w_i t_i & \\ & A_{.i} - ey_i \ \leq x_i & \forall i \in U, \\ & A_{.i} - ey_i \ \leq t_i & \forall i \in K, \\ & & 0 \leq x_U \leq 1 \\ & & y \geq 0. \end{array} \tag{19}$$

7.2 Creating a Rule that Covers an Specific Given Point or Set of Points

For some applications it may be the case that a rule that explains a single point or a given set of points is required. This set of points may be part of the training set or may be not. Lets call P the set of points for which we want to find a covering rule. We can enforce this requirement in the respective formulations for both the volume maximization and the point coverage criteria. This can be achieved by simply adding the following set of linear constraints to both formulations (8) and (17).

$$p_i \leq x_i \quad \forall i \in 1, \ldots, n \text{ and } \forall p \in P.$$

Note that in the case of formulation (17) that corresponds to the point coverage criteria, adding this set of constraints will generate redundant constraints if P is a subset of the training set T. Enforcing this set of constraints for points $p \in P \cap T$, is equivalent to simply remove the slacks variables that correspond to these points in formulation (17).

7.3 Rule Extraction and Knowledge-Based SVMS for Incremental Learning

Next, we present an introduction to the approach proposed in [8,9] to incorporating prior knowledge in the form of polyhedral *knowledge sets* into a linear programming SVM classifier formulation.

Knowledge-Based SVMs

Let's assume that we the following *knowledge sets* in the form of polyhedral sets (intersection of a finite family of closed halfspaces) are given:

$$\begin{array}{l} k \text{ sets belonging to } A+ : \{x \mid B^i x \le b^i\}, \quad i=1,\ldots,k \\ \ell \text{ sets belonging to } A- : \{x \mid \ C^i x \le c^i\},\ i=1,\ldots,\ell \end{array} \tag{20}$$

By Proposition 2.1 [8] this knowledge is equivalent to the following requirements with respect to the bounding planes (9):

$$\begin{array}{l} \text{There exist } u^i,\ i=1,\ldots,k,\ v^j,\ j=1,\ldots,\ell, \text{ such that:} \\ {B^i}' u^i + w = 0,\ {b^i}' u^i + \gamma + 1 \le 0,\ u^i \ge 0,\ i=1,\ldots,k \\ {C^j}' v^j - w = 0,\ {c^j}' v^j - \gamma + 1 \le 0,\ v^j \ge 0,\ j=1,\ldots,\ell \end{array} \tag{21}$$

All what is needed to do in order to incorporate the knowledge sets (20) into the SVM linear programming formulation (2), is to add the conditions (21) as constraints to (2) as follows:

$$\begin{array}{rl} \min\limits_{w,\gamma,y,u^i,v^j} & \nu e' y + \|w\|_1 \\ \text{s.t. } D(Aw - e\gamma) + y & \ge e \\ y & \ge 0 \\ {B^i}' u^i + w & = 0 \\ {b^i}' u^i + \gamma + 1 & \le 0 \\ u^i & \ge 0,\ i=1,\ldots,k \\ {C^j}' v^j - w & = 0 \\ {c^j}' v^j - \gamma + 1 & \le 0 \\ v^j & \ge 0,\ j=1,\ldots,\ell \end{array} \tag{22}$$

This linear programming formulation will ensure that each of the knowledge sets $\{x \mid B^i x \le b^i\},\ i=1,\ldots,k$ and $\{x \mid C^i x \le c^i\},\ i=1,\ldots,\ell$ lie on the appropriate side of the bounding planes (9). However, there is no guarantee that such bounding planes exist that will precisely separate these two classes of knowledge sets, just as there is no *a priori* guarantee that the original points belonging to the sets $A+$ and $A-$ are linearly separable.

We therefore add error variables $r^i,\ \rho^i,\ i=1,\ldots,\ k,\ s^j,\ \sigma^j,\ j=1,\ldots,\ \ell$, just like the error variable y of the SVM formulation (2), and attempt to drive these error variables to zero by modifying our last formulation above as follows:

$$
\begin{aligned}
\min_{w,\gamma,y,u^i,r^i,\rho^i,v^j,s^j,\sigma^j} \quad & \nu e'y + \\
& \mu\Big(\sum_{i=1}^{k}(r^i+\rho^i) + \sum_{j=1}^{\ell}(s^j+\sigma^j)\Big) + \|w\|_1 \\
\text{s.t.} \quad & D(Aw - e\gamma) + y \geq e \\
& y \geq 0 \\
& -r^i \leq B^{i\prime}u^i + w \leq r^i, \\
& b^{i\prime}u^i + \gamma + 1 \leq \rho^i, \\
& u^i, r^i, \rho^i \geq 0,\ i = 1,\ldots,k, \\
& -s^j \leq C^{j\prime}v^j - w \leq s^j, \\
& c^{j\prime}v^j - \gamma + 1 \leq \sigma^j, \\
& v^j, s^j, \sigma^j \geq 0,\ j = 1,\ldots,\ell.
\end{aligned}
\tag{23}
$$

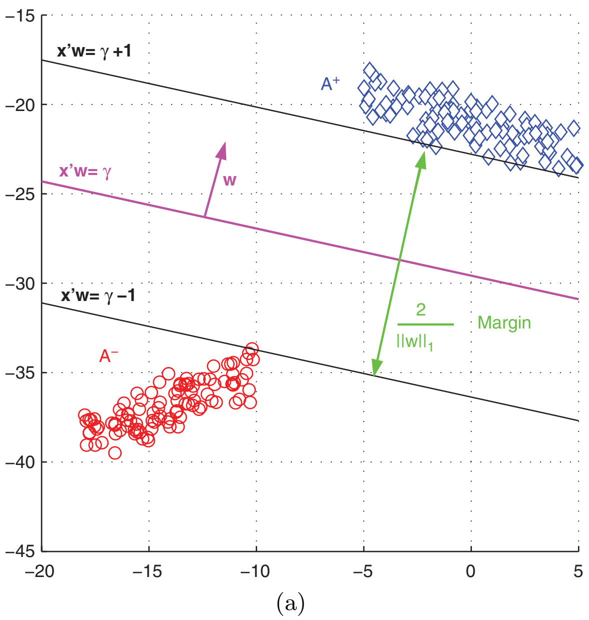
This knowledge-based linear programming formulation incorporates the knowledge sets (20) into the linear classifier with weight μ, while the (empirical) error term $e'y$ is given weight ν. As usual, the value of these two parameters, ν, μ, are chosen by means of a tuning set extracted from the training set. If we set $\mu = 0$ then the linear program (23) degenerates to (2), the linear program associated with an ordinary linear SVM.

The geometry of incorporating knowledge sets into a classification problem can be illustrated by considering a synthetic example in R^2 with $m = 200$ points, 100 of which are in $A+$ and the other 100 in $A-$. Figure 3 depicts ordinary linear separation using the linear SVM formulation (2).

It is important to note that if $\nu = 0$, then the linear program (23) generates a linear SVM that is strictly based on knowledge sets, but not on any specific training data. This might be a useful paradigm for situations where training datasets are not easily available, but expert knowledge, such as doctors' experience in diagnosing certain diseases, is readily available.

We can combine the rule extraction algorithm presented in Sect. 4 with the knowledge-based SVM formulation to propose an algorithm to perform incremental SVM learning. The algorithm is mainly motivated by the following two facts:

1. It is is usually the case that after applying Algorithm 4.1, the number of rules obtained is much smaller that the number of original training points used to train the SVM classifier, especially when using the maximum point coverage criteria. This is in fact strongly supported by the experimental section of this paper. The resulting set of rules can be seen as a compacted (compressed) representation of the points in the original training set.
2. By using the knowledge-based SVM formulation described above, a set of rules represented as a polyhedral set can be combined with a set of training points to obtain an optimal hyperplane classifier that takes into account both the rules sets and the training points for learning.



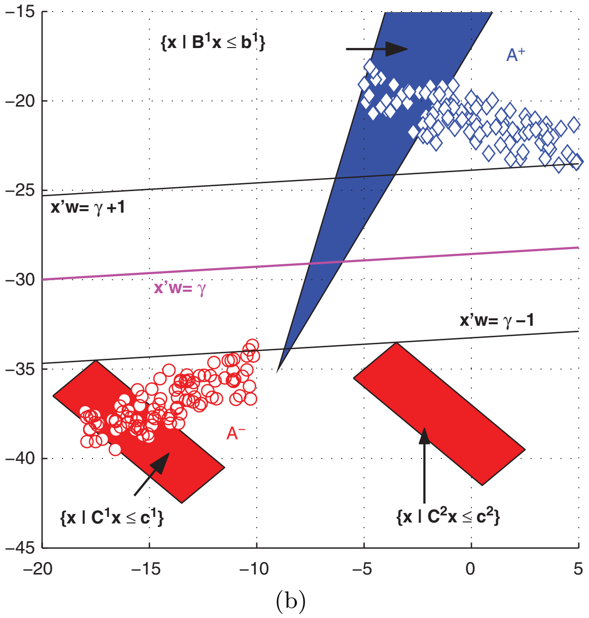


Fig. 3. (**a**) A linear SVM separation for 200 points in R^2 using the linear programming formulation (2). (**b**) A linear SVM separation for the same 200 points in R^2 as those in (a) but using the linear programming formulation (23) which incorporates three knowledge sets: $\{x \mid B^1x \le b^1\}$ into the halfspace of A^+, and $\{x \mid C^1x \le c^1\}$, $\{x \mid C^2x \le c^2\}$ into the halfspace of A^-, as depicted above. Note the substantial difference between the linear classifiers $x'w = \gamma$ of both figures

Let's assume now that we are interested in learning an SVM classifier and that we are in the incremental setting, this is, the training data comes continuously in batches through time: A_{t_1}, $A_{t_2}, \ldots$, and eventually the total accumulated dataset gets too large to be handled by a traditional learning algorithm, in this case formulation 2.

the main idea is that at any time t_k only a set of rules R_k obtained using Algorithm 4.1 is kept as a compact representation of past training data. When a new set of training data $A_{t_{k+1}}$ is available, this training data is combined with the set of rules R_k to obtained an updated classifier by using the knowledge-based SVM formulation. once the new classifier is obtained, the set of rules is updated.

Next we present a more detailed version of the algorithm:

Algorithm 7.1 ***IncrementalLeaning*(w, γ, I, d): Algorithm for incremental SVM learning.**
Given A_{t_o} and the corresponding labels. $iter = 0, R_0 = \emptyset$

```
1. If $iter = 0$, Solve formulation (2) to obtain an initial sparse hyperplane classifier $w_{iter}$.
2. If $iter > 0$, Solve formulation (17) using the new training data $A_{t_{iter}}$ and the compact set of rules $R_{iter}$ to obtain new sparse hyperplane classifier $w_{iter}$.
3. Use Algorithm 4.1 and $w_{iter}$ to generate an updated set of rules $R_{iter+1}$.
4. make $iter = iter + 1$ and got to 1.
```

Note that step 3 requires to use Algorithm 7.1 to obtain a new set of rules, the simplest way to do this is by using the volume maximization criteria (VMC) and the stopping criteria has to be set based on the amount of volume covered a each step.

Experiments and details on the formulations presented on this section are still work in progress and will presented in future work.

8 Conclusion and Future Directions

We have described an efficient algorithm for converting any arbitrary linear classifier into a rule set that can be easily interpreted by humans. We presented two variants of our algorithm based on different criteria for selecting "optimal rules". One main advantage of our algorithm is that it only involves solving relatively simple optimization problems in a few variables. We also discussed various properties and provided a detailed convergence analysis of the algorithm. Empirical results on several real-world data sets demonstrate the efficacy and speed of our method.

We plan to extend our numerical results to include comparisons to other rule-based classification methods. We are also considering other mathematical programming formulations where the rules can overlap since overlapping rules may have an advantage that may depend on the specific problem.

The incorporation of the feature selection into the rule extraction problem is also a possibility we are exploring at this moment. This approach would generate rules that depend on different features instead of depending on the same preselected subset of features.

So far we have focused on developing rule sets that are human interpretable models that are equivalent to the original linear classifier. An equally important use of our method would be to provide an explanation of the classification for a new unlabeled (test) example. The most obvious way is to present the user with the specific rule that includes the test example. For instance, when working with physicians, we have found that an explanation of a classification label which is in terms of a bounding hypercube, is far more understandable than "explaining" a label because some weighted sum of the variables is less than some constant.

The interesting case arises when no rule covers the test example. The obvious extension to execute **ExtractRules** on the region I which contains the test example, until a covering rule is found. However, the resulting rule may cover a very small volume around the test example, rendering the explanation useless. An alternate approach is not to build a rule set that is equivalent to the entire classifier, but instead to revise the original problem defined in (8) to extract just one rule – the largest possible hypercube (rule) which contains the test example. Such a rule, however, may not have much explanatory value because in most cases the test example will lie on one of the surfaces of the hypercube.

A more satisfactory explanation for a test sample may be provided by a rule where the example lies well within the interior of the rule, far away from the bounding spaces. The rule that provides the "optimal" explanation, can be created by drawing a normal from the test sample to the hyperplane, and the intersection of the normal with the hyperplane defines the corner of a uniquely defined bounding hypercube (rule), which centrally contains the test sample. Additionally, we can provide a confidence associated with the explanation (rule); ideally the explanation rule should cover all training examples in A_+ (A_-), contain only all positive (negative) training samples, be as large as possible (the volume ratio with respect to the rule created by **ExtractRules**), and for the test sample to be as far from the hyperplane. All these factors may be used to adjust the confidence associated with the rule (for the specific test sample) by weighting it using some scoring scheme. In general, these criteria may be applied to any explanatory rule, not just the "optimal" explanatory rules created as defined above.

References

1. D. P. Bertsekas. *Nonlinear Programming.* Athena Scientific, Belmont, MA, 1995.
2. Dimitri P. Bertsekas. Projected Newton methods for optimization problems with simple constraints. *SIAM Journal on Control and Optimization*, 20:221–246, 1982.

3. F. Beyer, L. Zierott, J. Stoeckel, W. Heindel, and D. Wormanns. Computer-assisted detection (cad) of pulmonary nodules at mdct: Can cad be used as concurrent reader? In *Proceeding of the 11th European Congress of Radiology*, Viena, Austria, March 2005. To appear.
4. E. H. Shortliffe B. G. Buchanan. *Rule-Based Expert Systems: the MYCIN experiments of the Stanford Heuristic Programming Project*. Addison-Wesley, Reading, MA, 1984.
5. P. S. Bradley and O. L. Mangasarian. Feature selection via concave minimization and support vector machines. In J. Shavlik, editor, *Machine Learning Proceedings of the Fifteenth International Conference (ICML '98)*, pages 82–90, San Francisco, California, 1998. Morgan Kaufmann. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-03.ps.
6. V. Cherkassky and F. Mulier. *Learning from Data - Concepts, Theory and Methods*. John Wiley & Sons, New York, 1998.
7. G. Fung and O. L. Mangasarian. Proximal support vector machine classifiers. In F. Provost and R. Srikant, editors, *Proceedings KDD-2001: Knowledge Discovery and Data Mining, August 26-29, 2001, San Francisco, CA*, pages 77–86, New York, 2001. Asscociation for Computing Machinery. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/01-02.ps.
8. G. Fung, O. L. Mangasarian, and J. Shavlik. Knowledge-based support vector machine classifiers. Technical Report 01-09, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, November 2001. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/01-09.ps, NIPS 2002 Proceedings, to appear.
9. G. Fung, O. L. Mangasarian, and J. Shavlik. Knowledge-based nonlinear kernel classifiers. Technical Report 03-02, Data Mining Institute, Computer Sciences Department, University of Wisconsin, Madison, Wisconsin, March 2003. ftp://ftp.cs.wisc.edu/pub/dmi/tech-reports/02-03.ps. Conference on Learning Theory (COLT 03) and Workshop on Kernel Machines, Washington D.C., August 24-27, 2003, submitted.
10. Glenn Fung. The disputed federalist papers: Svm feature selection via concave minimization. In *TAPIA '03: Proceedings of the 2003 conference on Diversity in computing*, pages 42–46. ACM Press, 2003.
11. F. J. Kurfes. Neural networks and structured knowledge: Rule extraction and applications. *Applied Intelligence (Special Issue)*, 12(1-2):7–13, 2000.
12. O. L. Mangasarian. Arbitrary-norm separating plane. *Operations Research Letters*, 24:15–23, 1999. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/97-07r.ps.
13. O. L. Mangasarian. Generalized support vector machines. In A. Smola, P. Bartlett, B. Schölkopf, and D. Schuurmans, editors, *Advances in Large Margin Classifiers*, pages 135–146, Cambridge, MA, 2000. MIT Press. ftp://ftp.cs.wisc.edu/math-prog/tech-reports/98-14.ps.
14. O. L. Mangasarian, W. N. Street, and W. H. Wolberg. Breast cancer diagnosis and prognosis via linear programming. *Operations Research*, 43(4):570–577, July-August 1995.
15. S. Mika, G. Rätsch, J. Weston, B. Schölkopf, and K.-R. Müller. Fisher discriminant analysis with kernels. In Y.-H. Hu, J. Larsen, E. Wilson, and S. Douglas, editors, *Neural Networks for Signal Processing IX*, pages 41–48. IEEE, 1999.
16. P. M. Murphy and D. W. Aha. UCI machine learning repository, 1992. www.ics.uci.edu/∼mlearn/MLRepository.html.

17. Haydemar Nuñez, Cecilio Angulo, and Andreu Catal. Rule extraction from support vector machines. In *ESANN'2002 proceedings - European Symposium on Artificial Neural Networks*, pages 107–112. d-side, 2002.
18. K. Preston. Computer processing of biomedical images. *Computer*, 9:54–68, 1976.
19. A. Tickle R. Andrews, and J. Diederich. A survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, 8:373–389, 1995.
20. L. B. Lusted and R. S. Ledley. Reasoning foundations of medical diagnosis. *Science*, 130:9–21, 1959.
21. J. Roehrig. The promise of cad in digital mammography. *European Journal of Radiology*, 31:35–39, 1999.
22. G. Towell & J. Shavlik. The extraction of refined rules from knowledge-based neural networks. *Machine Learning*, 13:71–101, 1993.
23. J. A. K. Suykens and J. Vandewalle. Least squares support vector machine classifiers. *Neural Processing Letters*, 9(3):293–300, 1999.
24. V. N. Vapnik. *The Nature of Statistical Learning Theory.* Springer, New York, second edition, 2000.

Rule Extraction Based on Support and Prototype Vectors

Haydemar Núñez[1], Cecilio Angulo[2], and Andreu Català[2]

[1] Artificial Intelligence Laboratory, School of Computing, Central University of Venezuela, Caracas, Venezuela
[2] Knowledge Engineering Research Group, Technical University of Catalonia, Rambla de l'Exposició s/n. E-08800 Vilanova i laGeltrú, Spain

The support vector machine (SVM) is a modelling technique based on the statistical learning theory (Cortes and Vapnik 1995; Cristianini and Shawe-Taylor 2000; Vapnik 1998), which has been successfully applied initially in classification problems and later extended in different domains to other kind of problems like regression or novel detection. As a learning tool, it has demonstrated its strength especially in the cases where a data set of reduced size is at hands and/or when input space is of a high dimensionality. Nevertheless, a possible limitation of the SVMs is, similarly to the neuronal networks case, that they are only able of generating results in the form of black box models; that is, the solution provided by them is difficult to be interpreted from the point of view of the user.

In the neuronal networks research area there has been a wide activity addressed to solve this situation by developing methods able to transfer the knowledge acquired by a neuronal network during the learning phase to a more amenable representation (Andrews et al. 1995; Craven and Shavlik 1997; Tickle et al. 1998; Tickle et al. 2000). The objective of these rule extraction methods is to use the neuronal networks like a tool for solving the problem, obtaining benefit from the advantages that offer these learning paradigms (like its good generalization property, its ability to process nonlinear relations and their high tolerance to the noise and the imprecision in the input data), as well as adding the possibility to open the black box, which would allow to obtain an explanatory result of the problem under study and a simpler solution to be understood by the user. Hence, the extraction of rules improves the adjustment of the neuronal networks to solve problems of data mining (Mitra et al. 2002; Witten and Frank 2005), when the primary target is to discover unknown and implicit relations in large databases that, in many cases, is necessary to be expressed in a comprehensible format.

Following this line of work, a solution to the lack of transparency of the models generated for the support vector machines would be the development of specific techniques of rule extraction directed to this kind of learning machines.

An interesting property of the SVM is that the hypothesis that it generates is built on the basis of a subgroup of training vectors called support vectors. These vectors constitute the key elements of the learning set since they are the points nearest to the decision limit and, therefore, represents the most informative patterns for building the solution. This explicit dependency of the learned model on the support vectors will facilitate the work of its interpretation.

A rule extraction method for the interpretation of support vector machines is presented in this work which uses the support vectors, along with prototype vectors generated by any clustering algorithm, for building a set of regions that fit the limit of the decision function defined by the SVM, so that it can be transferred to interpretable rules by the user (Núñez et al. 2002a, 2003). These regions can be built in two types: ellipsoids, which will generate equation-type rules, and hyper-rectangles, built from ellipsoids parallel to the axes of the variables, rising to a more comprehensible language in the form of interval rules.

In Sect. 1, the algorithm for the extraction of rules from a SVM is presented in detail, starting with the explanation about how generating an ellipsoid of maximal coverage that adjusts to the decision function generated for the SVM. Next, it is described the generation of a set of rules for a class. This algorithm will be modified to derive interval rules. The description of the mechanism ends with the description of the most interesting features of the algorithm. Several experiments on standard databases are described for different domains in Sect. 2, in order to evaluate the performance of the proposed rule extraction method, in particular its ability to extract the knowledge retained in a trained SVM. Since it is proved that the algorithm is strongly dependent on the initial conditions of the clustering algorithm used to derive the prototype vectors, in Sect. 3 a different approach is described for obtaining these prototype vectors based on the support vectors, overcoming in this form all the randomness due to the nowadays avoided clustering algorithm. Finally, some conclusions are derived and future works are sketched.

1 Combining Support Vectors and Prototype Vectors to Extract Rules

The methods for the extraction of rules from neuronal networks can be classified according to three basic features (Andrews et al. 1995; Craven and Shavlik 1997):

- The representation language used to describe the model estimated by the network.
- The form how the method explores the network to derive the rules. In this sense, local methods analyse the structure of the network at level of both, the hidden units and the output units to extract the rules, whereas

global methods extract rules on the basis of the input–output mapping generated by the network, with not analysis of its internal structure. Hybrid techniques have also been settled out in the between of these two methods.
- The portability of the method; that is, whether the technique is applicable independently of the network architecture and its training regime.

These features could be also used to define a rule extraction method for support vector machines and, similar to the case of neuronal networks, the main problem to be solved is to transfer the knowledge acquired by a SVM during the learning to a description in a new and comprehensible representation language. A key point to be considered is the functional equivalence between the new model and the SVM model from which it was extracted, by providing the same predictions.

For the development of such a translation technique, it is important to identify how the knowledge is codified for the hypothesis generated by a SVM. The solution provided by these learning machines is an expansion of kernel functions on the basis of the number of support vectors,

$$f_a(x) = sign\left(\sum_{i=1}^{sv} \alpha_i y_i K(x, x_i) + b\right). \tag{1}$$

It could be affirmed then that knowledge is expressed in the form of:

- A set of support vectors $SV = \{(\mathbf{x}_\mathrm{i}, \mathrm{y}_\mathrm{i})_{\mathrm{i}=1\ldots\mathrm{sv}}\}$, which are the data from the learning set LS nearest the limit of separation between classes
- A set of values $A = \{\alpha_{\mathrm{i}=1\ldots\mathrm{N}}\}$ associated to the data, which indicate whether or not a pattern is a support vector without error $(0 < \alpha < C)$, a support vector with error $(\alpha = C)$, or a pattern not considered in building the decision function $(\alpha = 0)$
- A kernel function K and its associated parameters, such as degree in a polynomial function or width in a Gaussian kernel

The support vectors, although in general are a small group of patterns into the learning set, are the most informative samples for the classification task. Being these vectors the points nearest the limit of separation between classes, would turn out advantageous then to use them in the extraction technique in the form of a set of class delimiters establishing the borders of regions defined in the input space that can be transferred to rules. In the method detailed in this work, these regions are a form of ellipsoids and the new representational language is a rule equation of the type 'if-then' using like antecedent or premise to the mathematical equation of the ellipsoid and like consequent, the label of the class associated to the data covered by this one, as it is shown in Fig. 1.

Transferring to this new representation the knowledge captured by the SVM during the learning entails the determination of a set of ellipsoids that fit to the form of the decision limit, avoiding as much as possible the overlapping between classes in the new model. In this sense, it is beneficial for

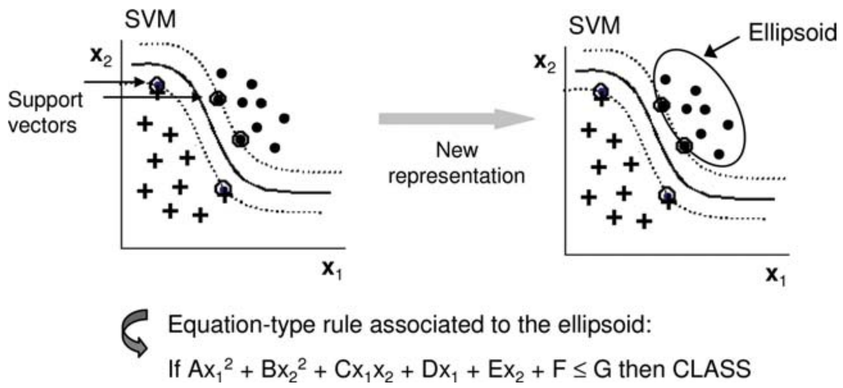

SVM
x_2
Support vectors
x_1
New representation
SVM
Ellipsoid
x_2
x_1
Equation-type rule associated to the ellipsoid:
If $Ax_1^2 + Bx_2^2 + Cx_1x_2 + Dx_1 + Ex_2 + F \leq G$ then CLASS

Fig. 1. Representational language used for the rule extraction

building these regions to use, in addition to the support vectors, the information provided by the vector of parameters α, since it indicates whether a support vector is in the region associated to another class or within the margin of separation between classes (those with a value of the associated parameter equal to C). In reference to the kernel functions, it is considered advisable that the method is independent of this characteristic in order that it can be applied to the widest range of SVM models.

Once defined the representational language of the extraction method, the problem to be approached is how to build these ellipsoids for a class from the information provided by a trained SVM. It will be two aspects that will be taken into account to perform this task:

- Ellipsoids fitting. The set of ellipsoids must fit the form of the discriminant function defined by the SVM, exhibiting as low overlapping between classes as possible to avoid problems of multiple instances in the set of rules. This premise will allow building rules with a high precision.
- Ellipsoids coverage. It is advisable to built ellipsoids covering as much data as possible with the purpose of producing a compact set of rules.

It is explained in the next, in a detailed form, the extraction method for building the associated ellipsoids to a class. It will be presented in a two-stage incremental procedure: first it will be assumed that it is possible to represent a whole class with a single ellipsoid. Next, it will be described how to generate a set of rules when this premise is not fulfilled, as it is usually the case.

1.1 Building an Ellipsoid and Its Associated Rule Equation

It is possible to define an ellipsoid associated to an input data set by determining the covariance matrix of the set and finding their eigenvectors and eigenvalues (Strang 1998). In this form, a set of axes for an ellipsoid following the directions of greater variance of the data is obtained, the vertices being determined from the own values. Nevertheless, the idea underlying in the proposed method is that the ellipsoids should adjust to the form of the limit of decision generated by the SVM. In order to obtain an ellipsoid with these characteristics, its orientation is defined by the support vectors, which are explicitly used for determining the associated set of axes of the ellipsoid and vertices by means of geometric methods. Hence, the problem associated to determining such an ellipsoid in the m-dimensional input space, can be defined as:

Given

- A set of support vectors $\boldsymbol{SV} = \left\{(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \Re^{m+1}, \boldsymbol{i} = 1 \ldots \boldsymbol{sv}\right\}$ obtained by using some SVM training procedure
- A set of remaining training data $\boldsymbol{D} = \left\{\left(\boldsymbol{x}_j, \boldsymbol{y}_j\right) \in \Re^{m+1}, \left(\boldsymbol{x}_j, \boldsymbol{y}_j\right) \notin \boldsymbol{SV}\right\}$, defined by excluding support vectors from the original training data
- A set of parameters $\mathrm{A} = \{(\alpha_{\boldsymbol{k}}) \in \Re, k = 1 \ldots N\}$, obtained by the SVM training procedure, being different to zero those associated to the support vectors

Determine

- A centre $\boldsymbol{p}$
- A set of orthonormal vectors, $\boldsymbol{E} = \{(\boldsymbol{e}_i), i = 1 \ldots m\}$
- A set of pair of vertices, $\boldsymbol{V} = \{(\boldsymbol{v}_{i1}, \boldsymbol{v}_{i2}), i = 1 \ldots m\}$

To define an ellipsoid which orientation is explicitly determined by the support vectors.

The algorithm in pseudo-code to derive an ellipsoid is in Table 1. First step in the algorithm is the initialization of the output values. Output sets

Table 1. Algorithm for deriving an ellipsoid

```
{Input: SV, D, A}
Initialize p, E, V
Build_Ellipsoid
    $\{e_1, v_{11}\}$ = Determine_First_Axis_Vertex
    $v_{12}$ = Determine_Second_Vertex
    $E = E \cup e_1, V = V \cup \{(v_{11}, v_{12})\}$
    p = Update_Prototype
    For k = 2 to m
        $\{e_i, v_{i1}\}$ = Determine_Next_Axis_Vertex
        $v_{i2}$ = Determine_Second_Vertex
        $E = E \cup e_i, V = V \cup \{(v_{i1}, v_{i2})\}$
    End_For
End_Build_Ellipsoid
{Output: p, E, V}
```

are set to null and the first considered centre of the ellipsoid is defined like the centre of gravity of all the data in the class,

$$p = \frac{1}{N} \sum_{i=1}^{N} x_i, \boldsymbol{x}_i \in LS. \tag{2}$$

Once the initial prototype is calculated, now the algorithm to build the ellipsoid determines, using the Determine_First_Axis_Vertex procedure, the first axis of the ellipsoid as well as the first vertex in this axis. Using Determine_Second_Vertex, the second vertex along the first axis is defined. E is a matrix of column vectors containing the orthonormal vectors defining the axes of the ellipsoid. V holds for the vertices associated to each axis. Next, the algorithm enters into a loop in which, both, the axis and its first vertex are determined each iteration by means of the Determine_Next_Axis_Vertex procedure, as well as the second vertex through the Determine_Second_Vertex procedure. It will be now detailed each one of the procedures used in the algorithm.

Determine_First_Axis_Vertex Procedure

This procedure determines both, the vector that will be used like the first axis and the first vertex considered on this axis. The vector $e_1 \in E$ is built from the prototype $\boldsymbol{p}$ and the support vector without error, i.e. $\alpha < C$ having maximal distance to the prototype,

$$q_{11} = \left\{ x | \, \|x - p\| = \text{argmax} \left(\|x_j - p\| \right), x_j \in SV, \alpha_j < C \right\}. \tag{3}$$

As it were already indicated, an ellipsoid as general as possible should be built, but fitting the form of the decision limit defined by the SVM.

Whether more than a support vector fulfils this condition, one of them is randomly selected. In the opposite, it is also possible that a support vector without error does not exist. In this case, the pattern in the set D with maximal distance to the prototype is selected,

$$q_{11} = \left\{ x | \, \|x - p\| = \text{argmax} \left(\|x_j - p\| \right), x_j \in D \right\}. \tag{4}$$

The unitary vector is defined as,

$$e_1 = \frac{q_{11} - p}{\|q_{11} - p\|}. \tag{5}$$

And the first vertex along this axis is the selected end-point,

$$v_{11} = q_{11}. \tag{6}$$

Determine_Second_Vertex Procedure

Through this procedure the second vertex on the axis is determined in the following form: first, a search region for support vectors is built by considering the subset of support vectors accomplishing,

$$SV_{i2} = \left\{ x \in SV | \arccos\left(\frac{(x-p)\cdot e_i}{\|x-p\|} \right) \geq \frac{3\pi}{8} \right\}. \tag{7}$$

Support vectors in this zone are those such that its projection with respect to the prototype over the axis is greater that any other projection over orthogonal axes to this one. The support vector related with the second vertex is that in SV_{i2} without error and larger distance to the prototype $\boldsymbol{p}$,

$$q_{12} = \{x | \, \|x-p\| = \operatorname{argmax}\left(\|x_j - p\| \right), x_j \in SV_{i2}, \alpha_j < C\}. \tag{8}$$

Again, when more than a support vector exists fulfilling the specific condition, one of them is randomly selected. For the case that no support vector exists in the zone, the pattern in D that satisfies the established criterion is selected.

Since the more general ellipsoid is searched, the second vertex is determined as,

$$v_{i2} = p - \|q_{i2} - p\| \, e_i. \tag{9}$$

When it does not exist neither data in the SV_{i2} set nor in the set D, the vertex is defined by using the first vertex as,

$$v_{i2} = p - \|v_{i1} - p\| \, e_i. \tag{10}$$

Update_Prototype Procedure

Once vertices are both determined on the first axis, the prototype is updated to be its midpoint. In this form, the original averaged initial centre $\boldsymbol{p}$ is replaced through the geometric situation of the support vectors by,

$$p = \frac{v_{11} + v_{12}}{2}. \tag{11}$$

Determine_Next_Axis_Vertex Procedure

This procedure iteratively determines orthonormal vectors to the set E included in the $m - i + 1$ linear manifold containing the centre $\boldsymbol{p}$. They are selected guided by the support vectors. Hence, first the set of support vectors

$$SV_{i1} = \left\{ x \in SV | ang_x \leq \frac{\pi}{4} \right\}, \tag{12}$$

is firstly determined, where ang_x is the angle determined by the vector $(x - p)$ and the $m - i + 1$ linear subspace,

$$ang_x = \arccos\left(\sqrt{1 - \sum_{j=1}^{i-1}\left(\frac{(x-p)\cdot e_j}{\|x-p\|}\right)^2}\right). \tag{13}$$

From this set, the vector with greater distance to the prototype is selected,

$$q_{\mathrm{i}1} = \{x|\ \|x-p\| = \mathrm{argmax}\left(\|x_j - p\|\right), x_j \in SV_{i1}, \alpha_j < C\}. \tag{14}$$

When no support vector exists in the searching zone, then the pattern in D that fulfils the maximal distance criterion is selected. In order to determine the unitary vector, a projection of the former vector with respect to the prototype on the $m - i + 1$ linear subspace is used. It is built by using the projection matrix $\mathbf{M}_p$ onto the subspace generated by E,

$$\mathbf{M}_p = EE^T. \tag{15}$$

Hence, the projection is defined as,

$$pq_{i1} = (q_{i1} - p) - \mathbf{M}_p\,(q_{i1} - p) \tag{16}$$

the orthonormal vector is built as,

$$e_i = \frac{(pq_{i1} - p)}{\|pq_{i1} - p\|} \tag{17}$$

and the first vertex in this vector will be defined as,

$$v_{i1} = p + \|q_{i1} - p\|\, e_i. \tag{18}$$

A key point to be solved is related with the infeasibility to find in the searching region a support vector or a pattern for determining an orthogonal vector to the set E. When it is not possible to find such a pattern in a certain iteration k, then it will be also impossible to find a pattern accomplishing the criteria in the successive iterations, that is, when the linear subspace will be smaller. In this situation, it has not more sense to evaluate our proposed algorithm of finding for the axe defined from support vectors. The ellipsoid is projected in this case on the k-dimensional linear manifold and a spheroid is obtained in this subspace.

In order to obtain a set of k orthonormal vectors in the k-dimensional subspace, an equation system can be solved and one of the infinite solutions be selected. Nevertheless, it is proposed to use a procedure based on the Gram-Schmidt orthogonalization (Strang 1998) which is described in the following.

Given a standard set of unitary vectors,

$$U = \{(u_i)\}_{i=1,\ldots,m} \tag{19}$$

an auxiliary set B_e, initialized as U is defined. Each time that a vector in E is determined, the nearest vector in B_e is searched,

$$proy_{\max} = \arg\max_{u \in B_e} (u \cdot e_i) \tag{20}$$

$$u_{e_i} = \{u_j \in B_e | u_j \cdot e_i = proy_{max}\} \tag{21}$$

and it is removed from the set B_e.

In the case that it is impossible to find a point (a support vector or a general pattern) that fulfils the criterion determining an axis, the k orthonormal vectors to the set E are determined from the set B_e in the following form,

```
For $l = m - k + 1$ to $m$
    $\mathbf{M}_p = EE^T$
    $j = 1$
    $e_l = u_j - \mathbf{M}_p u_j; e_l = \frac{e_l}{\|e_l\|}; E = E + \{e_l\}$
    $j = j + 1$
End_For
```

This procedure allows obtaining a fast unique solution without solving a system of equations, using basic vector operations and matrix product. In order to determine the vertices on each axis it is necessary to find the radius of the spheroid, which can be obtained from the radii associated to the axes in E. Several choices are possible, to use the greater radius, to use the smaller radius or the average of the radii. Since the most general ellipsoid is desired, the greater radius criterion will be used. When overlapping with data of other classes due to this generalist criterion appears, the ellipsoid can be specialized later by using a procedure for determining a set of rules that will be described later.

Another point to be solved is about the fact that, with the exception of the first axis, it will be usual that the vertices on the axes define a different radius, therefore one of them must be refined. Two criteria to decide which vertex to refine can be: using the radius defined by the vertex derived from a support vector or considering always the greater one from radii. Again, if overlapping with data of other classes appears when applying this heuristic, then the ellipsoid will specialize.

Generating the Rule

Once generated the ellipsoid, the rule equation can be derived from the centre $\boldsymbol{p}$, the set of orthonormal vectors E and the set of vertices V in the following form:

Let's suppose a 3-dimensional input space, then the ellipsoid is defined as,

$$\left(\frac{x'_1}{r_1}\right)^2 + \left(\frac{x'_2}{r_2}\right)^2 + \left(\frac{x'_3}{r_3}\right)^2 \leq 1 \tag{22}$$

with $r_i = \|v_{i1} - p\|$, $p = (p_1, p_2, p_3)$ and

$$\begin{bmatrix} x'_1 \\ x'_2 \\ x'_3 \end{bmatrix} = E^T \cdot \begin{bmatrix} x_1 - p_1 \\ x_2 - p_2 \\ x_3 - p_3 \end{bmatrix} . \tag{23}$$

Developing (23), it is obtained an expression in the form

$$Ax_1^2 + Bx_2^2 + Cx_3^2 + Dx_1x_2 + Ex_1x_3 + Fx_2x_3 + Gx_1 + Hx_2 + Ix_3 + J \leq K. \tag{24}$$

Generating an equation-type rule with a form,

$IF\ Ax_1^2 + Bx_2^2 + Cx_3^2 + Dx_1x_2 + Ex_1x_3 + Fx_2x_3 + Gx_1 + Hx_2 + Ix_3 + J \leq K\ THEN$ CLASS.

1.2 Generating a Set of Rules

Is an ellipsoid enough to describe the distinctive zone of a class? A positive answer is possible for some cases, but it will not be thus for the general case. On the other hand, it is difficult to establish a priori the number of necessary rules to represent the model of a SVM.

Two basic premises exist on which the building of the new model is based: the generalization or covering of the rules and the accuracy or precision of the ellipsoids to fit the shape of the decision surface defined by the SVM. The proposed procedure to build an ellipsoid tries to satisfy them. However, when one ellipsoid is not enough to describe the data in a class, the question is how to determine a number of regions that exhibit these characteristics. For instance, it can be observed in Fig. 2a that the ellipsoid generated from the midpoint of the data invades the zone associated to the other class due to the curvature of the decision limit. By dividing the ellipsoid (as it is shown in Fig. 2b) overlapping is reduced and the two new regions fit better to the decision surface than the original region.

Therefore, to generate a set of rules, the extraction method will initially build one ellipsoid which will be divided (specialized) until a group of more specific ellipsoids covering the data in the class and fitting the shape of the separation surface defined by the SVM is obtained.

A mechanism for determining the centre of each ellipsoid must be available in order to be able executing our proposed procedure. Initially, the centre was built as the midpoint of the data in the class. Nevertheless, it must be defined how to find new centres and which data to use to build each ellipsoid when it is mandatory to divide the initial ellipsoid in two or more regions. The derivation of the centres or prototypes can be performed using a clustering algorithm (Duda et al. 2001; Kaufman and Rousseeuw 1990), which divides the data set of a class in a predetermined number of disjoint partitions and it determines a prototype or representative centre for each one of the partitions.

On the other hand, it is also necessary to establish conditions for determining when to divide an ellipsoid. Division criteria can be based on an index

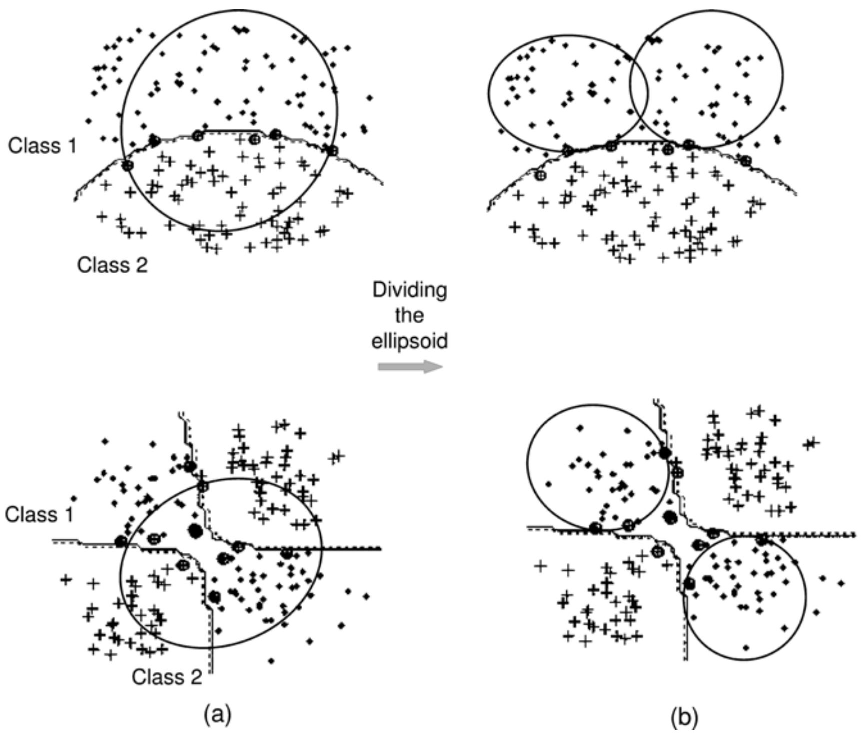


Fig. 2. (**a**) Generating one ellipsoid for a class. (**b**) By dividing the ellipsoid, two regions are obtained fitting better the decision function

of overlapping between classes, which could be implemented by means of geometric methods. It can be noticed in Fig. 2 that some support vectors exist belonging to another class within the defined ellipsoid. Since support vectors are the data nearest the decision function, they are the most informative about the shape of this separation surface. A first proposed criterion suggesting when to divide an ellipsoid is:

- *Criterion 1*: To divide an ellipsoid when support vectors of other classes exist in the region covered by it.

In this form, support vectors are used, not only for defining the ellipsoids, but also to verify the overlapping between ellipsoids of different classes. Overlapping also appears when the generated prototypes belong to another class (see Fig. 3); this fact can be verified by using the SVM function to generate the class label for these artificial points, which leads to a second partition criterion:

- *Criterion 2*: To divide an ellipsoid when the generated prototype belongs to another class.

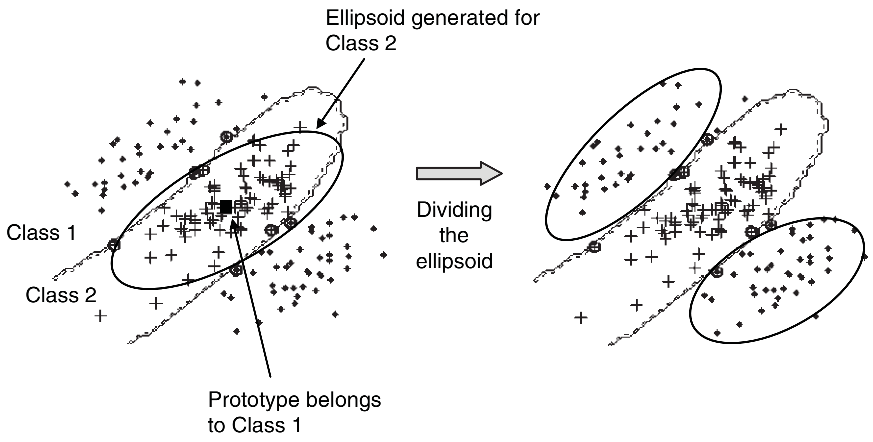


Fig. 3. The prototype generated like the midpoint of the data belongs to another class

Finally, overlapping between classes can also appear when at least one of the vertices generated by the algorithm to build an ellipsoid belongs to another class, which also can be verified using the trained SVM function. This third criterion of division is translated like:

– *Criterion 3*: To divide an ellipsoid when at least one of the vertices belongs to another class.

These three criteria can be formalized like a test of partition. Whether this test is positive when being applied to an ellipsoid, it indicates that it is very probable that it covers data from other classes.

It has been showed that overlapping can be reduced and fitting to the decision function increased by increasing the number of regions describing a class, however it should be now addressed how many regions are needed to describe data in a class. This number of regions could be user defined, providing in this form control on the size of the set of rules. Also it could be considered to divide ellipsoids until some established criterion is reached, for instance a good level of prediction.

According to this analysis, the extraction method for generating the set of ellipsoids associated to a class follows an iterative scheme. Starting by the prototype of the class (midpoint of the data) the initial ellipsoid is derived. Next, the partition test is applied on this region; when the answer to the test is negative the ellipsoid is transferred to a rule. Otherwise, a clustering algorithm is applied for determining two new prototypes with the data of the initial partition (data of the class); one ellipsoid is built for each one of them using the data of the respective partitions. The partition test is applied to these two new regions and the process is repeated. In this form, in the k-th iteration, *tp* regions have a positive test of partition and *tn* will be with negative answer. These last ones are transferred to rules. In the next iteration

$k + 1$, data from the tp regions are used to generate $tp + 1$ new prototype vectors which will lead to $tp + 1$ new regions. The procedure ends when all the partition tests have a negative answer or when the maximum number of iterations is reached (externally defined to control the number of generated rules). In Table 2 the complete algorithm to be used for deriving a set of rules is described, where:

– Determine_Prototypes (Data, Number_regions): It is a function for determining an equal number of prototypes to Number_regions for the data, using a clustering algorithm. For each prototype it also returns the

Table 2. Algorithm for deriving a set of rules

```
{Input: $SV$, $D$, SVM function}
Initialize_Generating_Rules
Do for each Class
   Number_regions = 1
   Data = Data_class
   [prototypes, partition] = Determine_Prototypes (Data, Number_regions)
   Ellipsoid = Build_Ellipsoid (prototypes, partition)
   Ellipsoid_rules = Ellipsoid
   Condition(1) = Partition_Test (Ellipsoid)
   Number_regions = 2
   While ((Condition(i) = 1) $\wedge$ Iterations $<$ max_iterations)
       [prototypes, partition] = Determine_Prototypes (Data,
       number_regions)
       For i = 1 to number_regions
          Ellipsoid(i) = Build_Ellipsoid (prototypes(i), partition(i))
          Condition(i) = Partition_Test (Ellipsoid (i))
       End_For
       Number_new_regions = 1
       New_data = [ ]
       For i = 1 to Number_regions
          If Condition(i) = 0 $\vee$ Iterations = max_iterations
             ellipsoid_rules = ellipsoid(i)
          Else
             Number_new_regions = Number_new_regions + 1
             New_data = New_data + Partition(i)
          End_If
       End_For
       Data = New_data
       Number_regions = Number_new_regions
   End_While
End_Do
{Output: rules}
```

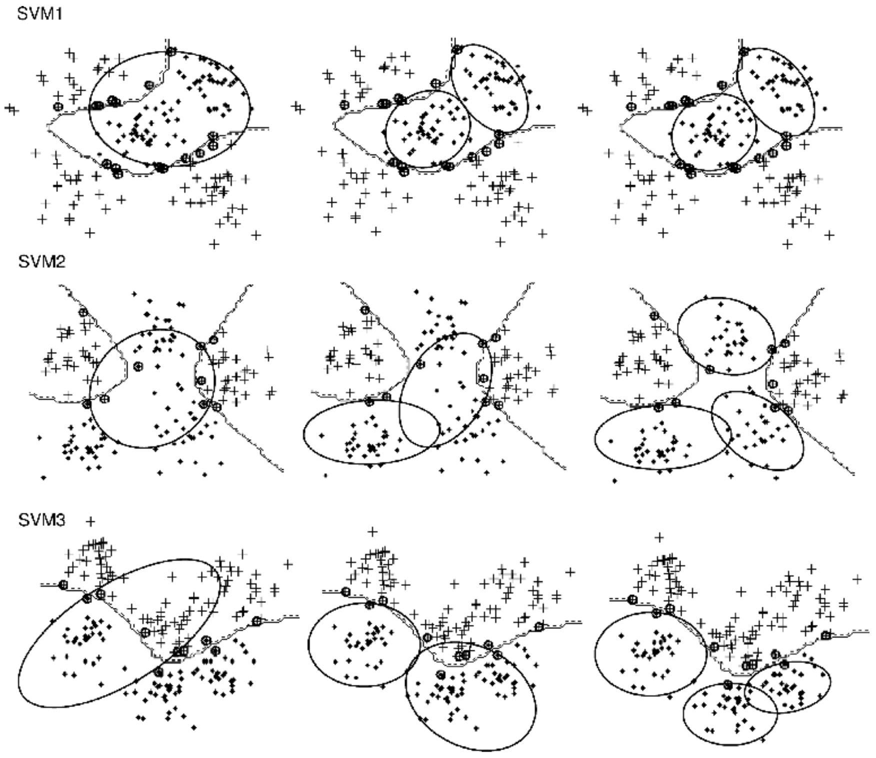


Fig. 4. Several examples of ellipsoids generated for learned SVM. Iteration *left* to *right*

respective partition. In the first iteration, the prototype will be the midpoint of the data.

- Build_Ellipsoid (prototypes, partition): This function builds an ellipsoid from a prototype and data from a partition.
- Partition_Test (Ellipsoid): It returns the logical answer of the partition test on an ellipsoid.

Figure 4 shows several examples of the iterative application of the rule extraction algorithm.

1.3 Simplified Representational Language for the Model

The ellipsoids and their equation-type rules define a representational language obtained by the rule extraction method to describe the model generated by the trained SVM. However, it will be showed that it is possible to derive more interpretable type rules by using like premise a set of constraints over the values of each one of the variables, to be satisfied so that the consequent one

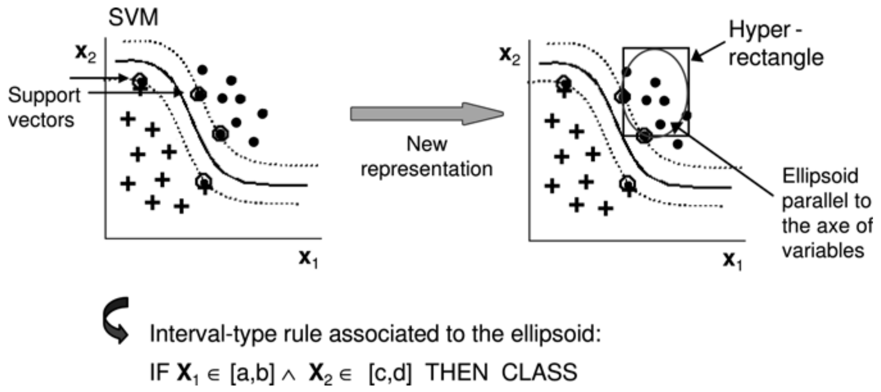

SVM
x_2
Support vectors
x_1
New representation
x_2
Hyper - rectangle
Ellipsoid parallel to the axe of variables
x_1
Interval-type rule associated to the ellipsoid:
IF $X_1 \in [a,b] \wedge X_2 \in [c,d]$ THEN CLASS

Fig. 5. Interval-type rule for SVM

is fulfilled (class label). This second representational language, called interval-type rules, can be observed in Fig. 5: it is associated to a convex region in the form of a hyper-rectangle generated from an ellipsoid parallel to the axes of the variables.

In order to derive the interval-type rules, the procedures Determine_First_Axis_Vertex and Determine_Next_Axis_Vertex of the Algorithm shown in Table 1 are modified. Key difference is about how the axes of the ellipsoid rising to a hyper-rectangle are built, by using the standard set U of unitary vectors. Support vectors are used to establish the order in selecting unitary vectors of the set B_e according to a criterion of proximity with the end points, as well as to build the vertices on the axes. This heuristic will allow that the ellipsoids parallel to the axes of the variables fit the shape of the separation surface defined by the SVM, without overlapping with the distinctive zone of other classes. These two modified procedures are described in the following.

Determine_First_Axis_Vertex Procedure

Given the standard set of unitary vectors U, it is initially defined $B_e = \mathrm{U}$, and $\boldsymbol{q}_{11}$ is determined as defined in Sect. 1.1. Next, it is calculated the projection of the vector $(\boldsymbol{q}_{11} - \boldsymbol{p})$ onto any vector in B_e. First selected unitary vector will be that in this set with higher projection, selecting in this form the axis with higher information about the support vector, that is,

$$proy_{max} = \arg\max_{u \in B_e} \left((q_{11} - p) \cdot u \right) \tag{25}$$

$$e_1 = \left\{ u_i \in B_e | \left(q_{\mathbf{11}} - p \right) \cdot u_i = proy_{max} \right\} . \tag{26}$$

The selected vector is removed from the set B_e. The vertex $\boldsymbol{v}_{11}$ is the projection of the vector $(q_{11} - p)$ on the next axis $\boldsymbol{e}_1$,

$$v_{11} = p + \left(e_1 \cdot (q_{11} - p) \right) e_1 . \tag{27}$$

Determine_Next_Axis_Vertex Procedure

In order to determine the next axe, the support vectors to be considered are those forming with the centre an angle lower than 45° with respect to the $m - i + 1$ linear manifold containing the centre and that is orthogonal to the set of vectors E, which leads to a subset defined as,

$$SV_{i1} = \left\{ x \in SV | ang_x \leq \frac{\pi}{4} \right\}, \tag{28}$$

where

$$ang_x = \arccos\left(\sqrt{1 - \sum_{j=1}^{i-1} \left(\frac{(x-p)\cdot e_j}{\|x-p\|}\right)^2}\right). \tag{29}$$

From this subset, the support vector without error with maximal distance to the prototype is selected,

$$q_{\mathrm{i}1} = \{x| \, \|x-p\| = \mathrm{argmax}\left(\|x_j - p\|\right), x_j \in SV_{\mathrm{i}1}, \alpha_j < C\}. \tag{30}$$

When there are no support vectors in the searching zone, the pattern in D satisfying the already established criterion for these vectors is selected. In order to determine the unitary vector, the usual projection onto each vector of B_e is considered, and the higher projected one is selected,

$$proy_{max} = \underset{u \in B_e}{\arg\max}\left((q_{\mathrm{i}1} - p)\cdot u\right) \tag{31}$$

$$e_{\mathrm{i}} = \{u_k \in B_e | \, (q_{i\mathbf{1}} - p)\cdot u_k = proy_{max}\}. \tag{32}$$

The selected vector is removed from B_e. The vertex is calculated as usual,

$$v_{i1} = p + (e_i \cdot (q_{i1} - p))\, e_i. \tag{33}$$

When it is impossible to find a point (a support vector or a general pattern) that fulfils the criterion determining an axis, the k unitary vectors to the set E are determined from the set B_e in the following form,

$$\boldsymbol{r}_{\boldsymbol{esferoide}} = \underset{(i=1\ldots(m-k))(j=1,2)}{argmax}\left(\|\boldsymbol{v_{ij}} - \boldsymbol{p}\|\right).$$

```
$For\ l = (m-k+1)\ to\ m$
    $\boldsymbol{j = 1}$
    $\boldsymbol{e_l = u_j}$
    $\boldsymbol{v_{l1} = p + r_{esferoide}\, e_l}$
    $\boldsymbol{v_{l2} = p - r_{esferoide}\, e_l}$
    $\boldsymbol{E = E + \{e_l\}}$
    $\boldsymbol{j = j + 1}$
$End\_For$
```

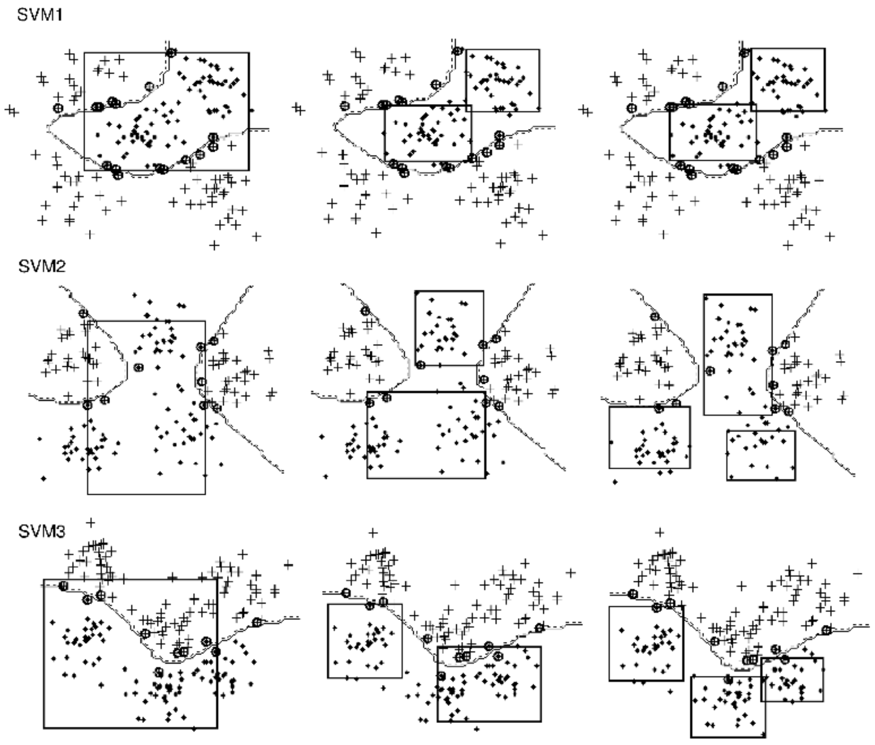


Fig. 6. Several examples of hyper-rectangles generated for learned SVM. Iteration *left* to *right*

The greater radius criterion is used for obtaining an as general ellipsoid as possible. Figure 6 shows some examples for hyper-rectangles associated to ellipsoids generated using the described iterative procedure.

Once derived the ellipsoid, its translation to an interval-type rule is performed by using a set of ordered vertexes $\boldsymbol{V} = \{(\boldsymbol{v}_{i1}, \boldsymbol{v}_{i2}), i = 1 \ldots m\}$ as follows,

$$IF\ x_1 \in [v_{11}, v_{12}] \wedge x_2 \in [v_{21}, v_{22}] \wedge \cdots \wedge x_m \in [v_{m1}, v_{m2}]\ THEN\ Class.$$

Determination of the set of interval-type rules for a class is processed similarly to that procedure described in Table 2; nevertheless, some modifications should be considered:

- The function Build_Ellipsoid is now based on the algorithm for building an ellipsoid parallel to the axis of the variables.
- Verification of the Criterion 1 in the function Partition_Test is performed on the hyper-rectangle associated to the ellipsoid in order to reduce the overlapping when deriving the interval-type rule.

1.4 Classification by Using the Set of Rules

Once obtained the model of the learned SVM in the new representation (equation-type or interval-type rules) it must be considered how this description will be used to classify a new pattern. Several scenarios can appear for the new pattern being evaluated:

- *It is covered only for one rule.* This is the most favourable case, allowing classifying a new entry without ambiguity.
- *It is covered for no rule.* In this case it is possible to define a default rule for classifying all these cases not covered for any rule (Mitchell 1997; Witten and Frank 2005). Using a similarity measure for determining the proximity of a pattern to a rule is an alternative choice; the assigned class label to the data will be those associated to the nearest rule (Domingos 1991).
- *It is covered for more than a rule.* Several solutions can be considered in this overlapping situation. A first one is ordering the covering rules using some quality measure and classifying the new instance according to the first fired rule (Berthold and Hand 1999; Witten and Frank 2005). A second one is applying a weighted classification scheme using all the covering rules similarly to the fuzzy logic algorithms. Other solutions include using a frequency based scheme that assigns the class associated to the most active rule, or, inversely, assign the pattern to the most specific rule (Salzberg 1991). Finally, it can be attempted to avoid multiple covering by refining the rules until a disjoint partition is achieved.

The proposed method classify an instance assigning it the label associated to the nearest rule (Domingos 1991), following the nearest neighbour technique. A distance measure between a pattern and a rule is defined depending on the type of rule, equation or interval. The distance between an equation rule and an instance is defined as,

$$D\left(R, x\right) = EQ\left(x\right), \tag{34}$$

where $EQ(\boldsymbol{x})$ is the result of evaluating the mathematical equation of the ellipsoid on the pattern $\boldsymbol{x}$. For the interval-type rules, the distance definition is based on a distance component for each attribute defined as follows,

$$\delta_i = \begin{cases} 0 & if \quad l_{i,\inf} \le x_i \le l_{i,\sup}, \\ x_i - l_{i,\sup} & if \qquad x_i > l_{i,\sup}, \\ l_{i,\inf} - x_i & if \qquad x_i < l_{i,\inf}, \end{cases} \tag{35}$$

where $l_{i,inf}$ y $l_{i,sup}$ are the lower and upper bounds of the interval, respectively, of the i-th component. Hence,

$$D\left(R, x\right) = \sum_{i=1}^{m} \delta_i. \tag{36}$$

When an instance is covered for more than a rule, the following heuristic is used to solve the overlapping: the most specific ellipsoid or hyper-rectangle containing the instance is selected, that is, that with the lowest volume (Salzberg 1991). For interval-type rules this volume is calculated as,

$$V(R) = \prod_{i=1}^{m} (l_{i,\mathrm{sup}} - l_{i,\mathrm{inf}}). \tag{37}$$

For equation-type rules, the volume associated to rules is compared by using an approximated measure based on the radius for each axis of the ellipsoid, in the following form,

$$V(R) = \prod_{i=1}^{m} r_i. \tag{38}$$

2 Experiments

The proposed rule extraction methods have been evaluated through experimentation on ten databases from the UCI repository (Blake and Merz 1998), considered a standard benchmark for the machine learning community. Features defining these bases are showed in Table 3: number of input variables, type of variables, number of patterns and number of classes.

The algorithms associated to the rule extraction method were developed under the Matlab v6.5 programming environment. SVM's training was completed using the software package "OSU Support Vector Machines Toolbox" version 3.00 (Ma and Zhao 2002). For multi-class classification, the one-versus-rest technique was employed for determining the SVM decision function (Vapnik 1998), i.e. a classifier was trained for each class and obtained support vectors were stored to be used next in the rule extraction algorithm. For

Table 3. Features describing the ten databases used for experimentation

Code	Databases	No. patterns	No. attributes	Type attributes	No. classes
1	IRIS	150	4	Numerical (continuous)	3
2	WISCONSIN	699	9	Categorical	2
3	WINE	178	13	Numerical (continuous)	3
4	SOYBEAN	47	35	Numerical (discrete)	4
5	NewTHYROID	215	5	Numerical (continuous)	3
6	MUSHROOM	8,124	22	Categorical	2
7	SPECT	267	23	Binary	2
8	MONK3	432	6	Categorical	2
9	ZOO	101	16	Categorical	7
10	HEART	270	13	Mixed	2

all the experiments, the k-means algorithm (Duda et al. 2001) was used for determining the centres or prototypes of the ellipsoids.

A key point is to determine the performance indexes to be used for evaluating the rule extraction algorithm (Andrews et al. 1995; Mitra et al. 2002; Zhou 2004). Goal is the extraction of the embedded knowledge in the trained SVM and represent it in the language defined by the method, so an interesting parameter is to determine the functional equivalence between both methods, known like *fidelity* parameter, being calculated like the percentage of data where both, the SVM and the rule set produce the same results,

$$Fidelity = 100 \cdot \frac{N_{agreed}}{Data_{total}}, \tag{39}$$

where N_{agreed} is the number of times that both, SVM and rule set predicts the same result.

A second main feature to be considered is the generality or covering of the rules on the data set, as well as their accuracy, defined through the error. Hence, two more parameters will be measured to determine the performance of the algorithm:

– Covering: Percentage of samples covered by the set of rules

$$Covering = 100 \cdot \frac{Data_{covered}}{Data_{total}}, \tag{40}$$

where $Data_{covered}$ is the number of samples covered by the rules and $Data_{total}$ is the size of the dada set

– Error: Mean quadratic error of the rule set on the data

In order to estimate these performance features, ten stratified cross-validation experiments on ten partitions were performed (Witten and Frank 2005) and the mean values on the test set were taken for the performance comparison. Obtained results for each database are shown in Table 4, for the equation-type and interval-type rules. It have been displayed the accuracy (Error), the percentage for the features measuring the equivalence or fidelity (Equ.), the covering (Cov.) and the number of obtained rules (NR).

Results on the application of the rule extraction method to support vector machines trained with real data databases in different domains, shown in general a high percentage of equivalence between the SVM and the extracted set of rules (upper to 90%). This high level indicates that the proposed methods are able to capture the embedded knowledge in the support vector machine.

It was also observed a high dependency of both, the quality and the quantity of the rules generated by the extraction method on the initial conditions for the k-means clustering algorithm. It is well-known that the final result of a clustering algorithm highly depends on the random choice of the prototypes (Duda et al. 2001), so it was a predictable result, but it is not a desirable behaviour. In this sense, a new proposal for reducing this randomness will be explained below, and it is still an open research area.

Table 4. Performance values of the rules for each database

Database	Error	Equation-type rules				Interval-type rules			
	SVM	Error	*Equ.*	Cov.	NR	Error	Equ.	Cov.	NR
1	0.046	0.041	98.67	82.33	6.1	0.038	97.59	80.66	3.8
2	0.045	0.039	98.65	86.45	15.3	0.041	96.54	94.45	14.5
3	0.023	0.018	98.40	78.34	5.9	0.023	97.87	80.96	8.9
4	0.022	0.022	100.00	33.00	6.0	0.028	97.70	84.50	6.0
5	0.052	0.049	97.13	80.24	7.3	0.047	95.33	72.99	10.8
6	0.002	0.003	96.05	28.87	25.5	0.010	99.06	98.19	30.8
7	0.102	0.117	96.26	21.49	14.0	0.093	97.33	45.00	28.0
8	0.023	0.034	97.45	27.55	7.0	0.023	99.07	100.00	10.0
9	0.042	0.043	99.09	32.02	9.8	0.043	98.77	79.01	8.6
10	0.164	0.158	96.93	58.35	6.7	0.155	96.39	66.52	18.7

They are listed below, as a particular example, the set rules generated for the IRIS database using both rule extraction regimes

Equation-type rules

R1: **IF** $(6.16\mathrm{X_1}^2 + 2.68\mathrm{X_2}^2 + 9.84\mathrm{X_3}^2 + 15.96\mathrm{X_4}^2 - 3.63\mathrm{X_1X_2} - 3.62\mathrm{X_1X_3} - 1.63\mathrm{X_1X_4} - 0.47\mathrm{X_2X_3} + 2.31\mathrm{X_2X_4} - 0.14\mathrm{X_3X_4} + 43.6\mathrm{X_1} + 0.01\mathrm{X_2} - 8.78\mathrm{X_3} - 7.52\mathrm{X_4} + 116.49 \le 2.64)$ **THEN** Iris-setosa

R2: **IF** $(1.29\mathrm{X_1}^2 + 4.79\mathrm{X_2}^2 + 3.31\mathrm{X_3}^2 + 5.28\mathrm{X_4}^2 + 1.69\mathrm{X_1X_2} - 2.02\mathrm{X_1X_3} + 0.97\mathrm{X_1X_4} - 2.50\mathrm{X_2X_3} - 1.84\mathrm{X_2X_4} - 2.09\mathrm{X_3X_4} - 11.77\mathrm{X_1} - 22.53\mathrm{X_2} - 6.34\mathrm{X_3} - 5.08\mathrm{X_4} + 78.30 \le 0.84)$ **THEN** Iris-virsicolor

R3: **IF** $(4.65\mathrm{X_1}^2 + 3.60\mathrm{X_2}^2 + 6.75\mathrm{X_3}^2 + 5.74\mathrm{X_4}^2 + 0.67\mathrm{X_1X_2} - 1.46\mathrm{X_1X_3} + 1.14\mathrm{X_1X_4} - 0.04\mathrm{X_2X_3} - 0.48\mathrm{X_2X_4} - 2.89\mathrm{X_3X_4} - 56.84\mathrm{X_1} - 25.00\mathrm{X_2} - 47.34\mathrm{X_3} - 9.05\mathrm{X_4} + 333.49 \le 1.81)$ **THEN** Iris-virsicolor

R4: **IF** $(9.91\mathrm{X_1}^2 + 5.63\mathrm{X_2}^2 + 12.13\mathrm{X_3}^2 + 9.26\mathrm{X_4}^2 - 3.21\mathrm{X_1X_2} - 3.89\mathrm{X_1X_3} + 7.69\mathrm{X_1X_4} + 4.90\mathrm{X_2X_3} + 0.03\mathrm{X_2X_4} + 0.57\mathrm{X_3X_4} - 127.63\mathrm{X_1} - 41.29\mathrm{X_2}\ 137.18\mathrm{X_3} - 96.32\mathrm{X_4} + 1,052.49 \le 3.54)$ **THEN** Iris-virginica

R5: **IF** $(16.25\mathrm{X_1}^2 + 49.58\mathrm{X_2}^2 + 19.01\mathrm{X_3}^2 + 66.59\mathrm{X_4}^2 - 11.35\mathrm{X_1X_2} - 8.13\mathrm{X_1X_3} - 5.03\mathrm{X_1X_4} + 4.98\mathrm{X_2X_3} - 54.77\mathrm{X_2X_4} - 10.81\mathrm{X_3X_4} - 109.50\mathrm{X_1} - 128.41\mathrm{X_2} - 140.12\mathrm{X_3} - 22.13\mathrm{X_4} + 886.57 \le 18.80)$ **THEN** Iris-virginica

Interval-type rules

R1: **IF** $(\mathrm{X_1} \in [4.40, 5.80] \wedge \mathrm{X_2} \in [2.30, 4.40] \wedge \mathrm{X_3} \in [1.00, 1.95] \wedge \mathrm{X_4} \in [0.20, 0.51])$ **THEN** Iris-setosa

R2: **IF** $(\mathrm{X_1} \in [5.40, 6.10] \wedge \mathrm{X_2} \in [2.70, 3.00] \wedge \mathrm{X_3} \in [3.90, 4.57] \wedge \mathrm{X_4} \in [0.97, 1.64])$ **THEN** Iris-versicolor

R3: **IF** $(\mathrm{X_1} \in [4.90, 6.00] \wedge \mathrm{X_2} \in [1.90, 2.93] \wedge \mathrm{X_3} \in [2.99, 4.00] \wedge \mathrm{X_4} \in [1.00, 1.27])$ THEN Iris-versicolor

R4: **IF** ($X_1 \in [6.10, 7.00] \wedge X_2 \in [2.30, 3.30] \wedge X_3 \in [4.28, 4.95] \wedge X_4 \in [1.31, 1.50]$) **THEN** Iris-versicolor
R5: **IF** ($X_1 \in [4.90, 6.70] \wedge X_2 \in [2.02, 3.67] \wedge X_3 \in [4.90, 5.57] \wedge X_4 \in [1.29, 2.63]$) **THEN** Iris-virginica
R6: **IF** ($X_1 \in [6.30, 7.70] \wedge X_2 \in [2.50, 4.00] \wedge X_3 \in [5.40, 7.19] \wedge X_4 \in [1.60, 2.67]$) **THEN** Iris-virginica

A 100% equivalence is obtained with the trained SVM for these two sets of rules and no test data is classified wrong.

3 Eliminating Randomness from the Clustering Algorithm

It has been realized during the experimentation that the method is very sensible to the used prototype vectors and, therefore, the quality and amount of the obtained rules varies depending on the location of the centres of the ellipsoids. These centres have been obtained from an initial solution provided by the clustering algorithm based on k-means, which randomly depends on the ordination of the provided training points (Duda et al. 2001). From the point of view of the extraction method this randomness is an obstacle, especially if it is necessary to extract several rules by class. Hence, it is required to apply the method several times on the trained SVM (with different initial conditions for the clustering algorithm) to be able to obtain a good solution, because the set and performance of the extracted rules show a high variance from an experiment to another one.

This situation of randomness and dependency of the method on the centres is a problem to be solved. It would be possible to evaluate in an empirical form different clustering algorithms for each database and to select that providing a greater stability. Nevertheless, a novel direct technique will be proposed for the determination of unique initial conditions for the clustering algorithm based on the support vectors provided by the learning machine. In short, the algorithm works as follows: if m prototypes are needed for the j-labelled class in the k-th iteration of the extraction algorithm, the algorithm proposes clustering the available data around m support vectors selected according to some criterion; once established the disjoint partitions, the midpoint of each one of them would be an initial centre for the clustering algorithm (Núñez et al. 2002c).

Let $\mathrm{SV_{jk}}$ be the set of support vectors in class j for the iteration k and let $\mathrm{D_{jk}}$ be the set of data in class j for the iteration k, then:

- Select m support vectors from $\mathrm{SV_{jk}}$
- Determine initial partitions $\mathrm{P_i}$, assigning each instance in $\mathrm{D_{jk}}$ to the nearest support vector according to the Euclidean distance,

$$\mathrm{P_i} \leftarrow \mathbf{x},\ \text{if } ^{\mathrm{d}}(\mathbf{x}, \mathbf{sv}_{\mathrm{i}}) = \underset{\mathrm{p=1\ldots m}}{\operatorname{argmin}} \left[\mathrm{d}\left(\mathbf{x}, \mathbf{sv}_{\mathrm{p}}\right)\right] \quad \forall \mathbf{x} \in \mathrm{D_{jk}} \tag{41}$$

– Next, calculate the midpoint for each partition,

$$u_i = \frac{\sum_{r=1}^{n_i} x_r}{n_i}\ , \tag{42}$$

where $\mathrm{n_i}$ is the number of patterns in the partition $\mathrm{P_i}$. The points $\mathbf{u}_{\mathrm{i}}$ will determine the initial conditions for the selected k-means algorithm

The criteria that could be used for the selection of the support vectors are the following ones (where it is only taken into account those vectors without error, i.e. with a value for the associated α lower than C):

- *Scheme of partition* EP_1: Support vectors are ordered in a descendent way according to its average similarity with data in the class (Kaufman and Rousseeuw 1990), so that the first m support vectors are selected.
- *Scheme of partition* EP_2: The m closest support vectors are chosen; it is aimed with this heuristic that the initial partitions are directed to the zones with greater curvature of the decision surface defined by the SVM.
- *Scheme of partition* EP_3: Support vectors are ordered in descendent form according to the value of the parameter α and the first m vectors are selected, on the hypothesis that larger is the value of the parameter, more informative is the associated pattern (Guyon et al. 1996).

For all these schemes of partition, in the case that only q support vectors are available with $q < m$, then the remaining vectors will be determined form the set $\mathrm{D_{jk}}$ by selecting those $m - q$ patterns with higher average similarity to the data.

One of these criteria has been empirically evaluated on trained support vector machines with the real databases of the UCI repository. In order to establish a direct comparison with the previously exposed results in Sect. 2, the same performance parameters were used, and they were identically calculated. Contrarily to the precedent results, now they are obtained by running a single iteration for the extraction algorithm, because SVM are unequivocally determined and so the centres from the proposed clustering procedure.

Table 5 shows the results obtained by using the scheme of partition EP_1. It can be observed that the performance is in average very similar to that obtained in Sect. 2. Therefore, using some of these schemes is a valid alternative route to be considered for generating the set of rules from a trained SVM trained in a determinist form. Possible extensions could be considered by defining some hybrid approaches that uses more than a partition scheme, with a decision module selecting the best set of generated rules.

Table 5. Performance values for the set of rules applied on each database using the scheme of partition EP_1

Database	Error SVM	Equation-type rules				Interval-type rules			
		Error	Equ.	Cov.	NR	Error	Equ.	Cov.	NR
1	0.046	0.041	98.00	80.40	6.4	0.037	97.00	81.40	4.5
2	0.045	0.039	97.85	86.03	16.1	0.042	96.00	90.33	15.9
3	0.023	0.020	98.33	78.40	6.2	0.024	92.90	79.42	9.8
4	0.022	0.022	100.00	25.00	6.6	0.020	98.00	73.00	6.2
5	0.052	0.048	97.04	78.46	8.0	0.045	94.82	72.03	11.3
6	0.002	0.003	96.51	29.81	25.4	0.009	99.10	97.56	31.4
7	0.102	0.143	90.78	20.01	12.0	0.112	96.01	56.34	32.00
8	0.023	0.025	96.99	40.05	8.0	0.022	99.53	95.60	11.00
9	0.042	0.044	99.02	29.33	10.3	0.045	98.16	79.00	8.8
10	0.164	0.160	97.13	57.40	6.5	0.167	97.30	62.24	19.3

4 Conclusions and Further Research

A method has been developed transforming the knowledge captured by a support vector machine during its learning in a representation based on rules, with the aim of equipping it with the capacity of explanation.

The algorithm proposed for the extraction of rules is based on the combination, using geometry elements, of the support vectors obtained from the SVM with prototype vectors derived from a clustering training regime to determine a set of ellipsoidal regions in the input space, later transferred to rules in the form of equation or interval rules. The hypothesis lying in this hybrid procedure is that, when using the support vectors, the defined regions adjust to the shape of the separation surface defined by the SVM with a minimal overlapping between classes.

An iterative procedure is followed for determining the set of rules, starting with the construction of a general ellipsoid that is successively specialized in more reduced ellipsoids in order to fit the shape of the decision function determined by the SVM. The partition criterion is also based on both, the support vectors and the decision function. The final number of rules, derived from the ellipsoids or hyper-rectangles, can be defined either, externally or through a stopping performance criterion.

Experimental results obtained when applying the rule extraction method on real databases from different domains shown a high degree of equivalence between the SVM and the extracted set of rules on test patterns. It can be so concluded that the proposed method is able to cope the acquired knowledge of the SVM during the learning phase.

None requirement is imposed in the initial method derivation about specific training regimes employed to train the SVM, kernel functions, nor clustering algorithm. Nevertheless, experimentation demonstrated that the quality and number of the generated rules with this method is highly dependent, due to the randomness of the clustering algorithm, on the location of the prototype vectors to be used as centres of the regions. A good average solution is only provided by the algorithm after some iteration with different initial conditions for the clustering algorithm.

A totally novel solution has been proposed to this problem by determining unequivocally the initial conditions from the unique set of support vectors. Three associated schemes of partition have been proposed to initialize the proposed clustering algorithm and build in a deterministic form the set of rules from a trained SVM. Empirical results showed the opportunity of such schemes or a hybridization of them to reduce the sensibility of the method to the clustering algorithm.

Starting from these proposed solutions to increase the explicative power of a learned SVM in the form of a set of rules, it is possible to plan new developments. For example, it would be interesting to study a direct extension of the rule extraction method to regression problems. In reference to the representational language, it could be studied using another one to express the new model, for example generating fuzzy rules from the ellipsoids. It would be also profitable developing algorithms for rule simplification that can be applied to improve, when it is required, the understand ability of the knowledge that has been extracted of the SVM.

Finally, it should be realized that the method can be extended to extract rules of other models, such as radial basis function networks (RBFN). The extraction algorithm could be designed so that the prototype vectors would be replaced by the centres of the RBF nodes (Núñez et al. 2002b), and the borders of the rules or their activation rates would be determined by the support vectors.

References

Andrews R, Diederich J, Tickle AB (1995) A Survey and Critique of Techniques For Extracting Rules From Trained Artificial Neural Networks, Knowledge Based Systems, 8, pp. 373–389

Berthold M, Hand D (1999) Intelligent Data Analysis An Introduction. Springer-Verlag

Blake CL, Merz CJ (1998) UCI Repository of Machine Learning Data-Bases. University of California, Irvine. Dept. of Information and Computer Science. (http://www.ics.uci.edu/~mlearn/MLRepository.html)

Cortes C, Vapnik V (1995) Support-Vector Networks. Machine Learning 20:273–297

Craven M, Shavlik J (1997) Using Neural Networks for Data Mining. Future Generation Computer Systems 13:211–229

Cristianini N, Shawe-Taylor J (2000) An Introducction to Support Vector Machines and other kernel-based learning methods. Cambridge University Press

Domingos P (1991) Unifying Instance-Based and Rule-Based Induction. Machine Learning 24:141–168

Duda R, Hart P, Stork D (2001) Pattern Recognition. 2nd edn. John Wiley & Sons, Inc

Guyon I, Martíc N, Vapnik V (1996) Discovery Information Patterns and Data Cleaning. In: Fayyad V, Piatetsky G, Smyth P, Uthurusamy R (eds). Advances in Knowledge Discovery and Data Mining. MIT Press

Kaufman L, Rousseeuw PJ (1990) Finding Groups in Data. An Introduction to Cluster Analysis. John Wiley & Sons, Inc

Ma J, Zhao Y (2002) OSU Support Vector Machines Toolbox, version 3.0. http://www.csie.ntu.edu.tw/∼cjlin/libsvm

Mitchell T (1997). Machine Learning. McGraw-Hill

Mitra S, Pal SK, Mitra P (2002) Data Mining in Soft Computing Framework: A survey. IEEE Transactions on Neural Networks 13(1):3–14

Núñez H, Angulo C, Català A (2002a) Rule extraction from support vector machines. Proc. 10th European Symposium on Artificial Neural Networks, pp. 107–112

Núñez H, Angulo C, Català A (2002b) Rule Extraction from Radial Basis Function Networks by Using Support Vectors. Lecture Notes in Artificial Intelligence 2527:440–449

Núñez H, Angulo C, Català A (2002c) Support Vector Machines with Symbolic Interpretation. 7th Brazilian Symposium on Neural Networks, IEEE, pp. 142–147

Núñez H, Angulo C, Català A (2003) Hybrid Architecture based on Support Vector Machines. Lecture Notes in Computer Science 2686:646–653

Salzberg S (1991) A Nearest Hyper rectangle Learning Method. Machine Learning 6:251–276

Strang G (1998) Introduction to linear algebra. 3rd. edition. Wellesley-Cambridge Press

Tickle A, Andrews R, Mostefa G, Diederich J (1998) The Truth will come to light: Directions and Challenges in Extracting the Knowledge Embedded within Trained Artificial Neural Networks. IEEE Transactions on Neural Networks 9(6):1057–1068

Tickle A, Maire F, Bologna G, Andrews R, Diederich J (2000) Lessons from Past, Current Issues, and Future Research Directions in Extracting the Knowledge Embedded Artificial Neural Networks. In: Wermter S, Sun R (eds) Hybrid Neural Systems. Springer-Verlag

Vapnik V (1998) Statistical Learning Theory. John Wiley & Sons, Inc

Witten I, y Frank E (2005) Data Mining. Practical Machine Learning Tools and Techniques with Java Implementations. Second edition. Morgan Kaufmann Publishers

Zhou Z (2004) Rule Extraction: Using Neural Networks or For Neural Networks? Journal of Computer Science and Technology. 19(2):249–253

SVMT-Rule: Association Rule Mining Over SVM Classification Trees

Shaoning Pang[1] and Nik Kasabov[2]

[1] Knowledge Engineering & Discovery Research Institute, Auckland University of Technology, Private Bag 92006, Auckland 1020, New Zealand spang@aut.ac.nz
[2] Knowledge Engineering & Discovery Research Institute, Auckland University of Technology, Private Bag 92006, Auckland 1020, New Zealand nkasabov@aut.ac.nz

1 Introduction

Since support vector machines (SVM) [7–9] demonstrate a good accuracy in classification and regression, rule extraction from a trained SVM (SVM-Rule) procedure is important for data mining and knowledge discovery [1–6,29,31]. However, the obtained rules from SVM-Rule in practice are less comprehensible than our expectation because there is a big number of incomprehensible numerical parameters (i.e., support vectors) turned up in those rules. Compared to SVM-Rule, decision-tree is a simple, but very efficient rule extraction method in terms of comprehensibility [33]. The obtained rules from decision tree may not be so accurate as SVM rules, but they are easy to comprehend because that every rule represents one decision path that is traceable in the decision tree.

The method of rule extraction from SVM trees (SVMT-Rule) achieves rule extraction over a decision tree of SVM, where rules are extracted not only from support vectors from the SVMs aggregated in the tree, but also from the tree structure in the way of decision tree rule. The benefits of SVMT-Rule are that the decision-tree rule provides better comprehensibility, and the support-vector rule retains the good classification accuracy. Furthermore, the SVMT-Rule is capable of performing a very robust classification on such datasets that have serious, even overwhelming, class-imbalanced data distribution, which profits from the super generalization ability of SVMT due to the aggregation of groups of SVMs.

This chapter exploits knowledge discovery by constructing a SVM classification tree, and decodes the SVMT into linguistic association rules.

In the literature, rule extraction based on single SVM focuses on interpolating the support vector and hyper-plane boundary from a trained SVM to a set of linguistic rule expression. Nunez et al. [1] extract a rule by first clustering support vectors by k-Mean, then for each cluster, choosing the prototype vector (i.e., the center of the cluster), and the support vector furthest to the prototype vector to build an ellipsoid using a geometric method. Finally, the ellipsoid is translated into a linguistic 'if-then' rule by an symbolic interpretation. Nunez's method used a reduced number of support vectors, but made the generated ellipsoid rule overlaps each other. In addition, this method requires to have parameters such as the number of cluster and initial cluster centers as the prior knowledge. As an improvement over Nunez's method, Zhang et al. [2] proposed a hyper-rectangle rule extraction (HRE) method. Instead of clustering support vectors, HRE clusters the original data using a support vector clustering (SVC) method to find the prototypes of each class samples, then constructs hyper-rectangle by the obtained prototypes and support vectors from the trained SVM. Because of the merits of the SVC algorithm, HRE can generate high quality rules even when training data contains outliers. Later, Fu et al. developed RulExSVM (rule extraction from support vector machine) [3, 6]. RulExSVM generates rules straightforwardly based on each of the support vectors. The rule condition is structured as a conjunction of several attribute conditions, each of them is built upon a hyper-rectangle associated with a certain support vector. RulExSVM is easy, but needs a tuning and pruning phase to reduce rules with overlap and outliers.

The above SVM rule extraction methods are regarded as incomprehensible techniques because they are completely support vector based rule extraction methods, and the knowledge of obtained rules is concealed in a number of numerical support vectors that are normally not transparent to the user. To mitigate this problem, the decision tree is a good rule extraction example that is recommended here for SVM rule extraction because every rule generated by a decision tree represents a certain decision path that has a comprehensible rule antecedent and rule consequence. For instance, C4.5 Rule [17, 30] interpolates every path from the root to a leaf of a trained C4.5 decision tree to an rule by regarding all the test conditions as the conjunctive rule antecedents while regarding the class label held by the leaf as the rule consequence. The comprehensibility of the C4.5 Rule is better than that of the C4.5 decision tree, because the C4.5 Rule has the knowledge over a decision tree fused and grouped, and comes with a concise knowledge structure.

Unlike decision trees, the SVMT is a type of SVM aggregation method towards combining a family of concurrent SVMs to achieve the problem-solving for traditional computational intelligence [11]. SVMT is different from the well known SVM ensemble [10, 18] aggregation in that (1) SVM ensemble takes the number of SVMs for aggregation and the structure of aggregation as the prior knowledge that is assumed to be known in advance, whereas SVMT has this prior knowledge learned automatically from data. (2) SVMT has a even more outstanding generalization ability than SVM ensemble in particular when it is confronted by tasks with the difficulty of a serious class-imbalance and/or class-overlap [12]. To extract rules from data, the presented SVMT-Rule in this chapter derives a new type SVM classification tree in the form of DFS-SVMT and BFS-SVMT, and models rule extraction as a rule induction over the resulting decision-tree structure and support vectors from local SVMs.

This organization of the chapter is given as follows. Section 2 gives the mathematical derivation of SVM classification trees. Section 3 presents the algorithms of SVM classification trees construction. Section 4 describes the detailed techniques for association rule extraction using SVMT (i.e., SVMT-Rule) method. In Sect. 5, we described the experiments on a synthetic data and two applications to cancer diagnosis and mobile telecom fraud detection. Finally, the conclusions and outlines of future work are pointed out in Sect. 6.

2 SVM Classification Tree

The SVM classification tree (SVMT) was first proposed in [12], in the context of face membership authentication application [10]. SVMT was shown to be capable of reducing the classification difficulty due to class-overlap by a recursive divide-and-conquer procedure, thus is useful in pattern recognition problems with large training samples but with noise information suppressed effectively via feature extraction. From the viewpoint of SVM aggregation, the SVM ensemble [18] assumes that the number of SVMs in aggregation should be known in advance as the prior knowledge, but this is often difficult to determine in real applications. SVMT has solved the difficulty of the SVM ensemble with the determination of the number of SVMs in a SVMT learned automatically from data.

However, because the spanning of SVMT in [12] is completely data-driven, this type of SVMT grows easily an overfitting due to noise, and come up with a large size decision tree, which is not optimal for decision making and rule extraction [35]. To mitigate this difficulty, a new type of SVMT with spanning order preference are modeled as follows.

2.1 Two-Class SVM Tree

Mathematically, a 2-SVMT can be formulated as a composite structural model as follows: Given a two-class dataset D for classification, and a predefined data partitioning function $\mathcal{P}$ on D, the whole dataset D can be divided through an optimized $\mathcal{P}^*$ into N partitions $\{g_1, g_2, \ldots, g_N\}$.

Then, a 2-SVMT can be modeled as,

$$\begin{aligned}&\mathcal{T}_{2-SVMT}(\mathcal{P}, f_{Svm_i^{<2>}}, f_{Sgn_j}^{<1>}, f_{Sgn_k}^{<2>}, \boldsymbol{x}),\\ &i = 1, \ldots, I,\ j = 1, \ldots, J,\ k = 1, \ldots, K,\end{aligned} \tag{1}$$

where $Svm_i^{<2>}$ is a local two-class SVM classifier. $Sgn_j^{<1>}$ and $Sgn_k^{<2>}$ represent a regional "one-class classifier" of class 1 and class 2, respectively. I,J,K such that $I + J + K = N$, correspond to the number of three data-partition types: partition with data from class 1 and class 2, partition with only data from class 1, and partition with only data from class 2, respectively. I,J,K, and N are determined after the SVM tree model generation. Figure 1 gives an example of two-class SVM and one-class SVM over data partition with data distributed in one-class and two-class, respectively.

In the case that g_i contains two classes data, a typical two-class SVM f_{Svm}, as Fig. 1a, is applied to model a local f_i on g_i as,

$$f_{Svm^{<2>}} = \sum_{i=1}^{l} y_i((\boldsymbol{w}_i)^T \varphi(\boldsymbol{x}_i) + b_i^*), \tag{2}$$

where φ is the kernel function, l is the number of training samples, and w, b is optimized through

$$\begin{gathered} \min \frac{1}{2}(\boldsymbol{w}_i)^T \boldsymbol{w}_i + C(\sum_{t=1}^{L} (\xi^i))^k, \\ y_i((\boldsymbol{w}_i)^T \varphi(\boldsymbol{x}_t) + b_i) \geq 1 - (\xi^i), \end{gathered} \tag{3}$$

where C and k are used to weight the penalizing variable ξ, $\varphi(.)$ acts the role of kernel function.

In another case, when g_i contains only data from one-class, g_i can be modeled strictly as a one-class classifier [13–16], where outlier samples are identified as negative samples among the positive samples, the origin data of the partition. Following [13], a one-class SVM function can be modeled as an outlier classifier as in Fig. 1b by setting positive on the origin data of the partition S and negative on the complement $\bar{S}$:

$$f_{Sgn^{<i>}} = \begin{cases} +1 \; if \; \boldsymbol{x} \in S, \\ -1 \; if \; \boldsymbol{x} \in \bar{S}, \end{cases} \tag{4}$$

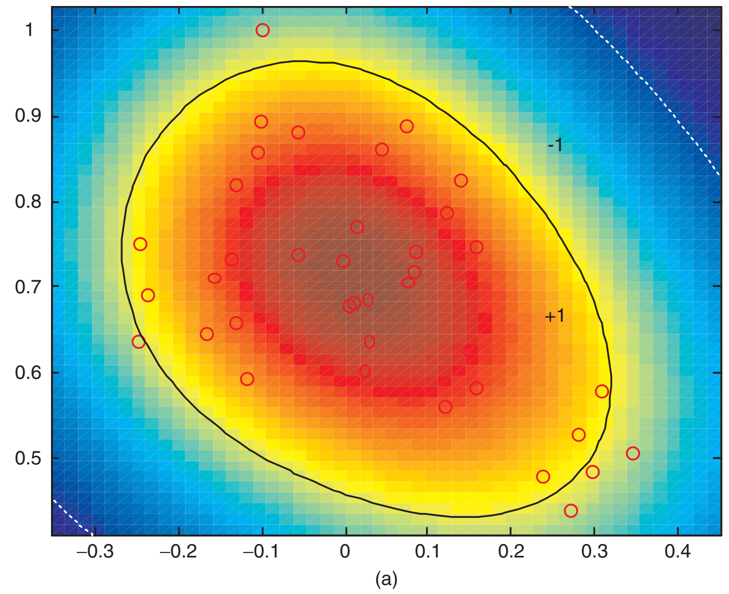
where i represents class label "1" or "2" in binary classification. In practice, the above one-class classifier also can be simplified as: if $\boldsymbol{x}$ by $\mathcal{P}$ belongs to class i, then the output of the one-classifier is assigned as i.

Thus, we can have the decision function of 2-SVMT $\hat{f}$ as,

$$\hat{f}(\boldsymbol{x}) = \begin{cases} f_{Sgn^{<1>}} \; if & \mathcal{P}(\boldsymbol{x}) \in class\ 1 \\ f_{Sgn^{<2>}} \; if & \mathcal{P}(\boldsymbol{x}) \in class\ 2, \\ f_{Svm^{<2>}} \; otherwise & \end{cases} \tag{5}$$

where one-class classifier $Sgn^{<1>}$ and $Sgn^{<2>}$ are one-class regional decision maker of 2-SVMT, and $Svm^{<2>}$ is two-class regional decision maker of 2-SVMT.

Clearly, error may occur in the classification of constructed $\hat{f}$, as $\hat{f}$ may differ from the true classification function f. Thus, a suitably chosen real-value



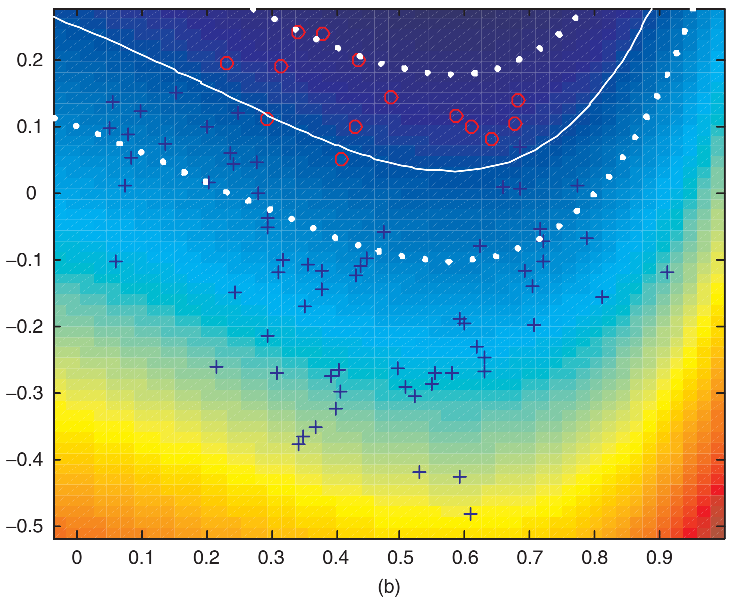


Fig. 1. Example of two-class SVM and one-class SVM over one data partition in the case that the data is in two classes, or all in one class. (**a**) one-class SVM; (**b**) two-class SVM

loss function $\mathcal{L} = \mathcal{L}(\hat{f}, f)$ is used to capture the extent of this error. Loss $\mathcal{L}$ is therefore data dependent:

$$\mathcal{L} = |\hat{f} - y|, \tag{6}$$

where $y = f(\boldsymbol{x})$. As $\hat{f}$ is applied to datasets drawn from the whole data D under a distribution of g. The expected loss can be quantified as,

$$E[\mathcal{L}] = |\hat{f} - y|g(D)dD. \tag{7}$$

In experiments, g can be realized by applying a tenfold cross validation policy on D. Since we have no information of the real classification function f, for simplicity we can fix y as the class label of the training dataset.

Thus, given data drawn under a distribution g, constructing an SVM tree here requires choosing a function f^* such that the expected loss is minimized:

$$f^* = \arg\min_{\hat{f} \in \mathcal{F}} |\hat{f} - y|g(D)dD, \tag{8}$$

where, $\mathcal{F}$ is a suitably defined SVM tree functions.

Substituting $\hat{f}$ to (6), the loss function of 2-SVMT is,

$$\mathcal{L} = \sum_{i=1}^{I} |f_{Svm_i^{<2>}} - f_i|\varrho_{2i} + \sum_{j=1}^{J} |f_{Sgn_j}^{<1>} - f_j|q_{1j} + \sum_{k=1}^{K} |f_{Sgn_k}^{<2>} - f_k|q_{2k}, \tag{9}$$

where f_i,f_j and f_k are the local true values of f. ϱ_{2i} represents the distribution probability of the ith partition that contains both class 1 and class 2 data. q represents the distribution probability of a partition that contains the data of one class (i.e., either class 1 or class 2).

In (9), $\mathcal{L}$ is determined by the performance of every regional classification from a regional SVM classifier or a one-class classifier. Given every SVM in 2-SVMT with the same linear kernel and penalty parameter, loss function $\mathcal{L}$ now depends only on the data distribution of local regions. In other words, the data partitioning method $\mathcal{P}$ eventually determines the loss function of the resulting SVM tree.

In this sense, the construction of SVMT f^* is equivalent to seeking a partition function $\mathcal{P}^*$ that is able to minimize the following loss function,

$$\begin{aligned}\mathcal{P}^* &= \arg\min_{\mathcal{P} \in [\mathcal{P}]} |\hat{f} - y|g(D)dD \qquad (10)\\ &= \arg\min_{\mathcal{P} \in [\mathcal{P}]} \sum_{i=1}^{I} |f_{Svm_i^{<2>}} - f_i|\varrho_{2i} + \sum_{j=1}^{J} |f_{Sgn_j}^{<1>} - f_j|q_{1j} + \sum_{k=1}^{K} |f_{Sgn_k}^{<2>} - f_k|q_{2k},\end{aligned}$$

where, $[\mathcal{P}]$ is a set of suitably defined 2-split data partition functions $\mathcal{P}$. The first term of the above function is the risk from SVM classification, and the remaining two terms are the risks from one-class classifier f_{Sgn}.

The risk from f_{Sgn} can be minimized through the training of an SVM one-classifier. In the case that Sgn is simplified by making every decision of classification through $\mathcal{P}$, the classification risk can be simply evaluated by the loss function of $\mathcal{P}$. Due to the loss function of $\mathcal{P}$ has been minimized when the data partitioning is performed. For a binary partitioning $[X_1, X_2] = \mathcal{P}(X)$, the risk can be the same minimized by a SVM training using $X1$ and $X2$ as class "+1" and "−1" respectively. So, (10) can be updated as,

$$\begin{aligned}\mathcal{P}^* &= \arg\min_{\mathcal{P}\in[\mathcal{P}]} |\hat{f} - y| g(D) dD \qquad (11)\\ &= \arg\min_{\mathcal{P}\in[\mathcal{P}]} \sum_{i=1}^{I} |f_{Svm_i^{<2>}} - f_i| \varrho_{2i}.\end{aligned}$$

Thus, we can solve the above function by finding out every individual partition that has the minimized SVM classification risk,

$$\mathcal{P}^* = \arg\min_{\mathcal{P}\in[\mathcal{P}]} |f_{Svm_i^{<2>}} - f_i| \varrho_{2i}. \qquad (12)$$

Equation (12) is proportional to the size of the region (i.e., number of samples in the region), thus in practice (12) can be implemented by seeking data partitions with different sizes and a minimized SVM classification risk. To this end, $\mathcal{P}^*$ is modeled as the following recursive supervised and scalable data partitioning procedure,

$$\begin{aligned}&[g_1, g_2, \cdots, g_j, \cdots g_N] = \mathcal{P}^n(X, \rho_0) \qquad (13)\\ &Subject\ to: \ L_{Svm}(g_j) < \xi \ or, \ g_j \in one\ \ class,\end{aligned}$$

where X is the input dataset. ρ_0 is the initial partitioning scale, and $L_{Svm} = |f_{Svm_i} - f_i|$ is a local two-class SVM classifier loss function. ξ is the SVM class separability threshold. $\mathcal{P}^n$ represents a recursive partitioning function such that $\mathcal{P}^n(X, \rho, L_{Svm}) = \mathcal{P}(\mathcal{P}^{n-1}(X, \rho_{n-1}, L_{Svm}), \rho_n, L_{Svm})$. For every iteration, optimal partitions are extracted out, and the remaining samples go to the next iteration of partitioning.

As a result of (12) and (13), partitions with L_{Svm} minimized is produced by $\mathcal{P}$ at different scales. Each of them builds a regional two-class decision maker $f_{Svm^{<2>}}$. Meanwhile, partitions that contain only one class data are also being produced. This builds the regional one-class decision makers $f_{Sgn^{<1>}}$ and $f_{Sgn^{<2>}}$.

3 The Spanning of SVM Tree

To construct a SVM tree, a completely data-driven spanning approach is, to split data whenever there exists class mixture in the current data partition [12], and to perform such data partitioning recursively until all data from different classes mostly goes to a different partition. As mentioned above, the SVM tree can be grown up automatically without use of any prior knowledge. However, this often grows easily the overfitting of learning, and comes up with a huge size SVM tree. To deal with this difficulty, we construct a SVM tree using the following two spanning order preferences.

3.1 Depth-First Spanning Tree

The idea of depth-first spanning is to expand the tree to a node of the next layer as quickly as possible before fanning out to other nodes of the current layer.

The procedure is as follows: at a data partitioning scale ρ, once has one data partition judged as classification optimal (either a one-class partition or a partition with $L_{Svm} < \delta$), then the tree expansion goes to the next layer, and one terminal node is produced at the current layer of the tree.

Thus (13) can be realized by repeating the following binary data partitioning,

$$\begin{aligned} [X_\rho, X'] &= \mathcal{P}(X, \rho, L_{Svm}) \\ Subjectto &: L_{Svm} > \xi, \end{aligned} \tag{14}$$

where X_ρ is the one optimal partition at scale ρ generated by $\mathcal{P}$, and X' is the remaining dataset such that $X = X_\rho \cup X'$.

3.2 Breadth-First Spanning Tree

Breadth-first is another tree spanning approach. The basic idea is to fan out to as many nodes as possible of current layer before expanding the tree to a node of the next layer.

This can be explained as, at a data partitioning scale ρ, if there are maximum υ data partitions judged to be classification optimal (either a one-class partition or a partition with $L_{Svm} < \delta$), then υ nodes are produced at current layer of the tree.

In this case, (13) uses a multiple data partitioning,

$$\begin{aligned} [\{X_1, X_2, \ldots, X_\varrho\}, X'] &= \mathcal{P}(X, \rho, L_{Svm}) \\ Subjectto &: L_{Svm} > \xi, \end{aligned} \tag{15}$$

where $\{X_1, X_2, \ldots, X_\upsilon\}$ is a set of optimal partitions at scale ρ, and X' is the remaining dataset such that $X = X_1 \cup X_2 \ldots \cup X_\upsilon\} \cup X'$. υ is the number of optimal partition at scale ρ, and it is determined by partitioning function $\wp$ and the used partitioning scale ρ.

3.3 The SVMT Algorithms

As constructing a SVMT, (10) is implemented by a 2-step recursive procedures. First, a partition method is employed to split the data at a certain scale into two or more partitions. Next, for those partitions that are optimal for the two-class pattern classification, a one-class classifier or a SVM classifier is built to serve as a node of the 2-SVMT.

Algorithm 1 and 2 below present the detailed steps of constructing a Depth-first spanning (DFS) 2-SVMT, and Breadth-first spanning (BFS) 2-SVMT, respectively.

[Algorithm 1. Depth-First Spanning 2-SVMT]

```
Inputs: $\mathcal{X}_{train}$ (Training dataset), $\rho_0$ (the initial partition scale), $\xi$ (the
    threshold of SVM loss function), $\mathbf{K}$ (the used type of SVM Kernel)
Outputs: $\mathcal{T}$ (a trained DFS 2-SVMT)
Step 1: Initialize $\mathcal{T}$ as the root node, and current partitioning scale $\rho$ as $\rho_0$.
Step 2: If $\mathcal{X}_{train}$ is empty then iteration stops, return constructed SVMT $\mathcal{T}$.
Step 3: Perform (15) data partitioning with current partitioning scale $\rho$, and
    obtain data partition $X_\rho$.
Step 4: If $X_\rho$ belongs to one-class, then train a one-class SVM as (4), and
    append a one-class SVM node to $\mathcal{T}$.
Step 5: Otherwise, $X_\rho$ belongs to two-class, and the classification loss func-
    tion of the partition is greater than $\xi$, train a standard two-class SVM as
    (2) and (4) over this partition, and append a two-class SVM node to $\mathcal{T}$.
Step 6: If no new node is added to $\mathcal{T}$, then adjust current partitioning scale
    by $\rho = \rho - \triangle\rho$.
Step 7: Set $\mathcal{X}_{train}$ as $\mathcal{X}_{train} - X_\rho$, and go to Step 2.
```

[Algorithm 2. Breadth-First Spanning 2-SVMT]

```
Inputs: $\mathcal{X}_{train}$ (Training dataset), $\rho_0$ (the initial partition scale), $\xi$ (the
    threshold of SVM loss function), $\mathbf{K}$ (the used type of SVM Kernel)
Outputs: $\mathcal{T}$ (a trained BFS 2-SVMT)
Step 1: Initialize $\mathcal{T}$ as the root node, and current partitioning scale $\rho$ as $\rho_0$.
Step 2: If $\mathcal{X}_{train}$ is empty then iteration stops, return constructed SVMT $\mathcal{T}$.
Step 3: Perform (16) data partitioning with current partitioning scale $\rho$, and
    obtain data partition $\{X_1, X_2, \ldots, X_\varrho\}$.
Step 4: For each data partition $X_i$, do the following operations
    (a) If $X_i$ belongs to one-class, then train a one-class SVM as (4), and
        append a one-class SVM node to $\mathcal{T}$.
    (b) Otherwise, $X_i$ belongs to two-class such that the Classification loss
        function of $X_i$ is greater than $\xi$, train a standard two-class SVM as
        (2) and (4) over this partition, and append a two-class SVM node
        to $\mathcal{T}$.
Step 5: If no new node is added to $\mathcal{T}$, then adjust current partitioning scale
    by $\rho = \rho - \triangle\rho$.
Step 6: Set $\mathcal{X}_{train}$ as $\mathcal{X}_{train} - \{X_1, X_2, \ldots, X_\varrho\}$, and go to Step 2.
```

To test the above constructed 2-SVMTs, a input sample $\mathbf{x}$ is first judged by the test function $\mathcal{T}(x)$ at the root node in the SVM tree. Depending on the decision made by the root node, $\mathbf{x}$ will be branched to one of the children of the root node. This procedure is repeated until a leaf node or a SVM node is reached, then the final classification decision is made for the testing sample $\mathcal{T}(x)$ by a node one-class classifier or a single SVM classification. Algorithm 3 illustrates the testing procedure of a 2-SVMT (DFS 2-SVMT or BFS 2-SVMT).

[Algorithm 3. 2-SVMT Testing Algorithm]

```
Inputs: $T$ (a trained 2-SVMT), $\mathbf{x}$ (a testing sample)
Output: $C$ (class label of the testing sample)
Step 1: Set the root node as the current node.
Step 2: If the current node is an internal node, then start the following loop
    operations:
    (a) Branch $\mathbf{x}$ to the next generation of $T$ by performing current node test
        function $\mathcal{T}_{current}(\mathbf{x})$.
    (b) Find the next node that $\mathbf{x}$ is branched to in $T$;
    (c) Set the node from (b) as the new current node, and go to Step 2.
Step 3: The above loop is stopped when a terminal node is reached, and the
    class label $C$ is computed by a decision test $\mathcal{T}_{current}(\mathbf{x})$. Here, the current
    node is a terminal node, which is either an SVM node or a node with a
    single class.
```

In the above SVMT algorithms, $\mathbf{K}$ specifies the type of SVM used in 2-SVMT construction. ρ_0 determines an initial resolution for 2-SVMT to start zooming in the data and to construct an SVMT over the data. ξ is the permitted minimum loss function value, which gives a criterion of partition selection for optimal SVM classification.

The default ρ can be set as the biggest the partitioning scale that the used partitioning method allows. For example, for the K-Mean partitioning method, ρ_0 can be set as 2; for ECM partitioning method, ρ_0 can be set as 0.9. Given a serious class-imbalance and class-overlap of data, a finer scale can be taken to enable 2-SVMT to separate classes in a more accurate resolution. The default ξ is 0. In practice, a small error is given. Certainly, the smaller ξ, the more time cost that it takes to get the 2-SVMT trained.

3.4 Coping with Class Imbalance and Class Overlap

The above 2-SVMTs cope with class-imbalance and class-overlap in two parallel ways:

1. The recursive data zooming-in schema, where larger size data partitions (particularly for those partitions of overweighed class, e.g., class 1 for Fig. 5 dataset) are separated out in the first few rounds. This reduces the imbalance interference for learning of the remaining data. Meanwhile, data with class-overlap/class-mixture is being left for the next round, thus 2-SVMT is able to zoom-in the data, and makes decisions only at a scale and region, where decision can be reached.
2. 2-SVMT model also can be adjusted to favor the smaller class (the class with less samples than the other) by defining L_{Svm} merely on the smaller class,

$$L_{Svm} = \frac{1}{l}\sum_{i=1}^{l}(1 - (y_i\lambda_i^* K(\mathbf{x}, \mathbf{x}_i) + b^*)). \tag{16}$$

   Note that l here is the number of smaller-class samples, and $\mathbf{x}$ is an input sample of the smaller class.

4 SVMT Rules Extraction

4.1 Logic Association Rules

According to the literature of rule mining, given a n-dimensional data space D, a standard logic rule is formed as

$$IF\ \boldsymbol{x} \in g_i\ THEN\ Class(x) = C_k, \tag{17}$$

where the IF part is the rule antecedent, and the THEN part is the rule consequence. The rule means that If x belongs to the subspace g_i, the it is assigned a class label C_k.

Typically, there is a conjunction of logical predicate function $T(\boldsymbol{x}) = \{True, False\}$ to tell if the condition part of the above rule is satisfied or not. In common case, the predicate function T is a test on a single attribute of D. For example, if $\boldsymbol{x}$ has attribute a_i value that belongs to an interval $x_i \in [a_i-, a_i+]$, then $\boldsymbol{x} \in g_i$.

However, when interpolating the knowledge from the decision tree structure of an SVMT, the predicate function T is extended to be on multiple attributes of D because every subspace g_i in an SVMT is obtained through a supervised data partitioning (13), and both the loss function and the partitioning function of (13) are a multivariate computation. Thus, the predicate function T will be a multivariate function determined by the choice of partitioning function $\mathcal{P}$

$$T(\boldsymbol{x}) = \mathcal{P}(g_i, \boldsymbol{x}) \tag{18}$$

$T(\boldsymbol{x})$ judges if $\boldsymbol{x}$ is partitioned into g_i. It could be a linear or nonlinear distance function depending on the choice of partitioning function.

On the other hand, when interpolating a support vector from a local SVM into a linguistic rule, the predicate function T is a conjunction of a set of tests on several relaxant attributes,

$$T(\boldsymbol{x}) = x_i \in [a_i-, a_i+] \cup \cdots \cap x_j \in [a_j-, a_j+]. \tag{19}$$

4.2 SVM Nodes Interpolation

As discussed above, a trained SVMT has two types of SVM nodes:

(1) Two-class SVM node $V = \{g, \rho, f_{svm}\}$, where g contains two classes data with $L_{svm} < \xi$ satisfied
(2) One-class SVM node $V^{<1>} = \{g, \rho, f_{svm^{<1>}}\}$ or $V^{<2>} = \{g, \rho, f_{svm^{<2>}}\}$, where g contains either class 1 or class 2 data

Two-Class SVM Node

For a two-class SVM node (called SVM node at the rest of writing), rule extraction according to [6] is based on hyper-rectangular boundary built upon support vector of a trained SVM. Given support vector s_m of class l, the hyper-rectangular boundary for s_m is,

$$\{s_{mi} + \lambda_{2i} \geq x_i \geq s_{mi} - \lambda_{1i}, \}, \tag{20}$$

where $1 \geq \lambda_{pi} \geq 0$, $p = \{1, 2\}$, and $i = 1, \ldots, n$ the sequence number of dimension.

Set L_o and H_o as the lower limit and upper limit of the hyper-rectangular rule along the ith dimension respectively, where,

$$L_o(i) = s_{mi} - \lambda_{1i} \tag{21}$$

and

$$H_o(i) = s_{mi} + \lambda_{2i}. \tag{22}$$

Given a trained SVM, the rule based on support vector s_m can be generated by Fu's algorithm [6] as,

[Algorithm 4. Two-Class Support Vector Rule Extraction]

```
Input:  $s_m$ (support vector)
Output:  $L_o$ (the lower limit of the hyper-rectangular rule), $H_o$ (the upper
    limit of the hyper-rectangular rule)
Step 1:  Set $l = 1$, refers to dimension $l$
```

Step 2: Calculate x_l subject to $f(x) = 0$ and $x_j = s_{mj}$ ($j = 1, \ldots, n$, and $j \neq l$) by the Newton's method

Step 3: Determine L_o and H_o according to the solutions of the problem in Step 2. The number of solutions of x_l may be different under different data distribution:

(a) If there is no solution, i.e., there is no cross point between the line extended from s_m along dimension l and the decision boundary, then $L_o(l) = 0, H_o = 1$

(b) If there is one solution: if $s_{ml} \geq x_l$, $L_o(l) = x_l$, and $H_o(l) = 1$, else $L_o(l) = 0$, and $H_o(l) = x_{l2}$

(c) If there are two solutions, x_{l1} and $x_{l2}(x_{l1} \leq x_{l2}) : L_o(l) = x_{l1}$, and $H_o(l) = x_{l2}$

(d) If there are more than two solutions the nearest neighbor x_{l1} and x_{l2} of x_{l2} are chosen from the solutions under the condition $x_{l1} \leq S_{mj}$,$x_{l2} \geq x_{l2}$ and the data points $\boldsymbol{x}|x_j = s_{mj}, j = 1, \ldots, n, j \neq l, x_{l1} \leq x_l \leq x_{l2}$ are with the same class label with the support vector $s_m : L_o(l) = x_{l1}$, and $H_o(l) = x_{l2}$

Step 4: $l = l + 1$, if $l < n$, go to Step 2, else end.

Fig. 2 gives an example of hyper-rectangular rule extraction from a trained two-class SVM in SVMT, where the black circles and points represent the

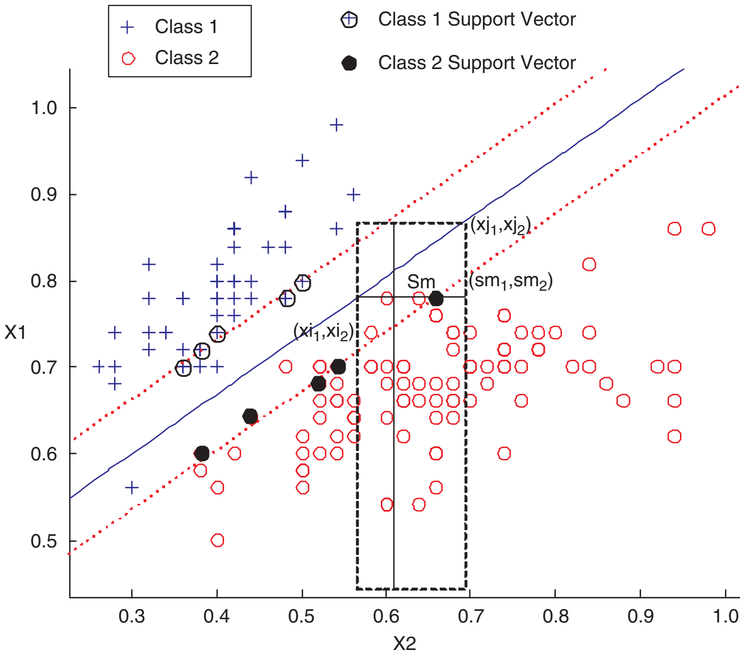


Fig. 2. Example of hyper-rectangular rule extraction from a trained two-class SVM

support vectors from class 1 and class 2 respectively. Given a support vector $s_m = (s_{m1}, s_{m2})$ from class 2, there is two cross points (x_{i1}, x_{i2}) and (x_{j1}, x_{j2}) between the extend lines of s_m and the decision boundary. According to Algorithm 4, a hyper-rectangular rule with $L_o(1) = x_{i1}$, $H_o(2) = x_{j1}$, $L_o(2) = x_{i2}$ and $Ho(2) = x_{j2}$ is obtained. The coverage area of this rule is identified as the dashed rectangle in Fig. 2.

One-Class SVM Node

For one-class SVM node, the one-class SVM approximates the origin of the class by distinguishing the outlier data from the major distribution of the class data. As another format of such knowledge approximation, the obtained hyper-rectangle rules are required to cover the class data as much as possible.

Given two support vectors s_m and s_n from a trained one-class SVM, and $S_{mi} \neq s_{ni}$ along ith dimension, L_o and H_o are computed as,

$$L_o(i) = min(s_{mi}, s_{ni}) \tag{23}$$

and

$$H_o(i) = max(s_{mi}, s_{ni}). \tag{24}$$

[Algorithm 5. One-class Support Vector Rule Extraction]

```
Input: $s_m$ and $s_n$ (two support vectors)
Output: $L_o$ (the lower limit of the hyper-rectangular rule), $H_o$ (the upper limit of the hyper-rectangular rule)
Step 1: Set $l = 1$, refers to dimension $l$
Step 2: Calculate $x_l$ subject to $f(x) = 0$ and $x_j = s_{mj}$ ($j = 1, \ldots, n$, and $j \neq l$) by the Newton's method.
Step 3: Determine $L_o$ and $H_o$ according to the solutions of the problem in Step 2. The number of solutions of $x_l$ may be different under different data distribution:
    (a) If there is no solution, i.e., there is no cross point between the line extended from $s_m$ along dimension $l$ and the decision boundary, then $L_o(l) = min(0, s_{ml}, s_{nl}) H_o = max(1, s_{ml}, s_{nl})$
    (b) If there is one solution: $L_o(l) = min(x_l, s_{ml}, s_{nl})$, and $H_o(l) = max(1, s_{ml}, s_{nl})$, else
    (c) If there are two solutions: $L_o(l) = min(x_{l1}, x_{l2}, s_{ml}, s_{nl})$, and $H_o(l) = max(x_{l1}, x_{l2}, s_{ml}, s_{nl})$
    (d) If there are more than two solutions the two Neighbor $x_{l1}$ and $x_{l2}$ nearest to $s_m$ and $s_n$ are chosen, $L_o(l) = min(x_{l1}, x_{l2}, s_{ml}, s_{nl})$, and $H_o(l) = L_o(l) = max(x_{l1}, x_{l2}, s_{ml}, s_{nl})$.
Step 4: $l = l + 1$, if $l < n$, go to Step 2, else end.
```

Figure 3a illustrates an example of the above pairwise support vector rectangular rule extraction. According to Algorithm 5, given a n-dimensional one-class data with m obtained support vectors from one-class SVM, $n.m(m-1)/2$ rules are generated. In other words, a small number of support vectors produce a large number of rules, and they are for the same partition, as well as the same class of data. The resulting rules certainly have much redundancy.

Since the data from one-class SVM, either outliers or the origin of the class, both are in the same class. From the viewpoint of classification, there is no need to separate one from another. In this way, the above rule extraction of (23) and (24) can be simplified as,

$$L_o(i) = min(s_{1i}, s_{2i}, \ldots, s_{mi}, \ldots) \tag{25}$$

$$H_o(i) = max(s_{1i}, s_{2i}, \ldots, s_{mi}, \ldots). \tag{26}$$

Correspondingly, Algorithm 5 is modified to extract rectangular rules only from the pair of vectors (maximum and minimum). For example, Fig. 3b illustrates such rule extraction, where a set of smaller rectangles of Fig. 3a are replaced with one rectangle, which is built on a pair of support vectors, and which covers most of the one-class data.

[Algorithm 6. Simplified One-class Support Vector Rule Extraction]

```
Input: $s_1, s_2, \ldots, s_m$ (set of support vector from a trained one-class SVM)
Output: $L_o$ (the lower limit of the hyper-rectangular rule), $H_o$ (the upper
    limit of the hyper-rectangular rule)
Step 1: Set $l = 1$, refers to dimension $l$
Step 2: Calculate $L_o(l)$ by (25)
Step 3: Calculate $H_o(l)$ by (26)
Step 4: $l = l + 1$, if $l < n$, go to Step 2, else end.
```

4.3 SVMT-Rule

Over the encoded SVMT in Sect. 3, the rule knowledge is decoded through decision-tree rule extraction and SVM node knowledge interpolation. Figure 4 gives the block diagram of the SVMT Rule extraction method, where an SVM classification tree is first modeled on the training data, and followed by three rule extraction procedures over the obtained SVMT, including rule extraction over the tree structure, SVM node, and one-class SVM node, respectively. Consequently, the final rule set is obtained by merging all the obtained rules via a redundancy pruning procedure. Algorithm 7 presents the final algorithm of rule extraction over SVMT, where individual steps have been detailed in the above subsections.

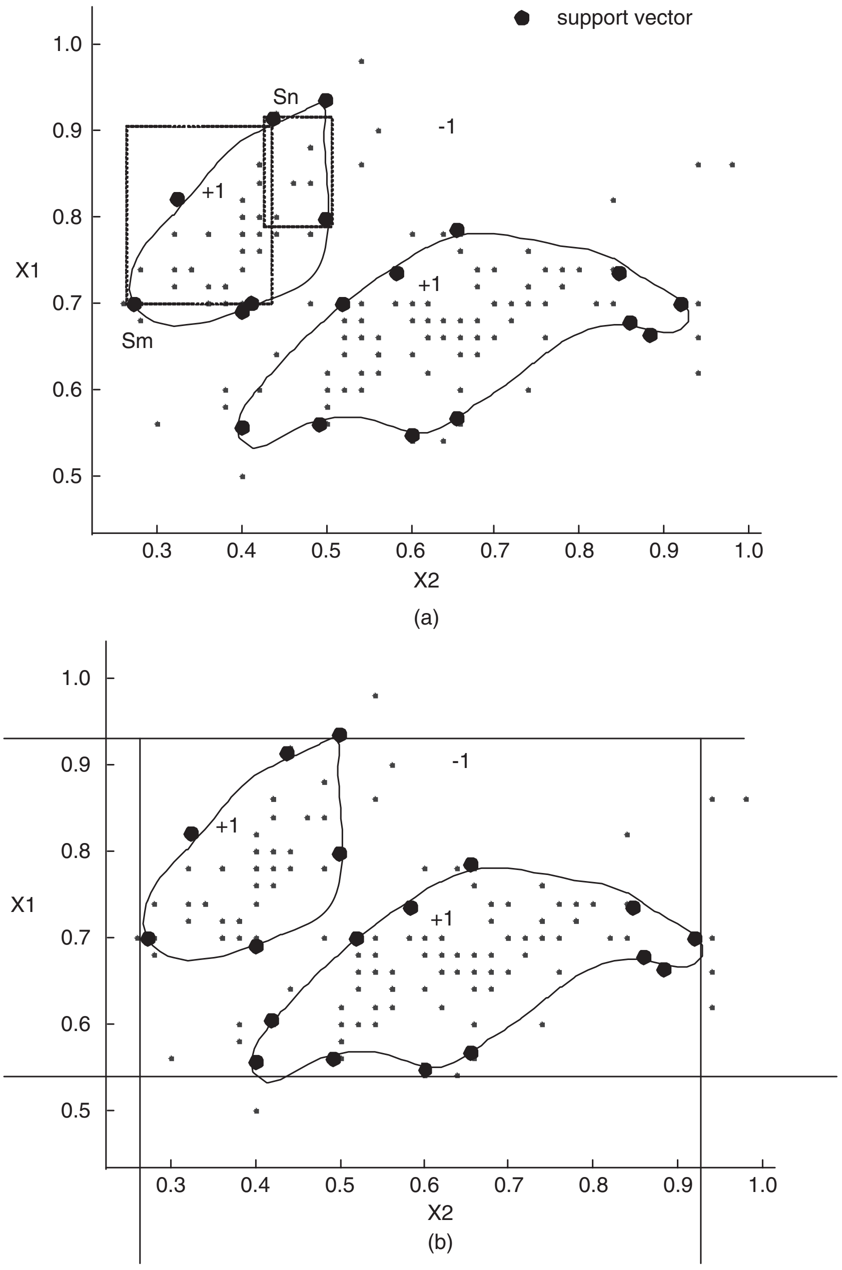


Fig. 3. Example of hyper-rectangular rule extraction from a trained one-class SVM. (**a**) Pairwise support vector rectangular rule extraction; (**b**) the maximum and minimum pair support vector rectangular rule extraction

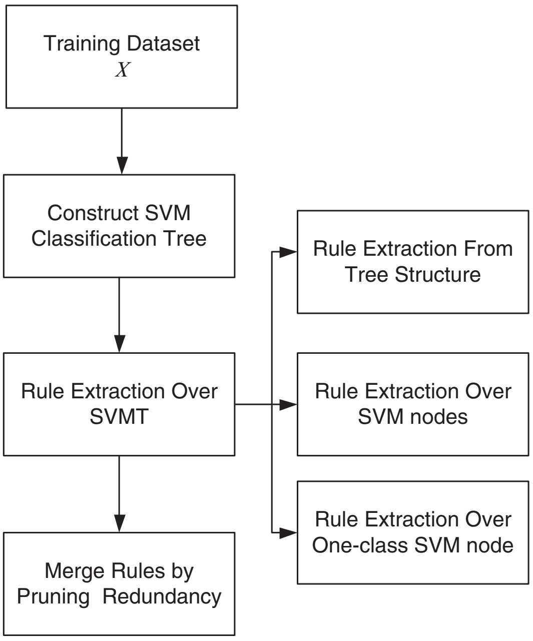

Training Dataset
X
Construct SVM Classification Tree
Rule Extraction Over SVMT
Merge Rules by Pruning Redundancy
Rule Extraction From Tree Structure
Rule Extraction Over SVM nodes
Rule Extraction Over One-class SVM node

Fig. 4. Block diagram of the proposed SVMT-Rule procedure

[Algorithm 7. Rule Extraction Over SVM Trees]

Inputs: $\mathcal{X}_{train}$ (Training dataset)
Output: R (tuned ruleset), C (Decision tree ruleset), S (Support vector ruleset)
Step 1: Apply the SVM classification tree algorithm (Algorithm 1 for DFS SVMT, and Algorithm 2 for BFS SVMT) to construct $\mathcal{T}$ over $\mathcal{X}_{train}$.
Step 2: For every path in $\mathcal{T}$ from the root node to a terminal node of SVMT, extract rules C by taking all test functions along the path as the conjunctive rule antecedent, and the class label held by the leaf as the rule consequence, $R = C$.
Step 3: For every terminal node (i.e., SVM node or one-class SVM node) in $\mathcal{T}$, do the following operations:

(a) If the terminal node that the leaf is attached is an SVM node, apply Algorithm 4 to the terminal node for the support vector rule R_{svm^2} extraction, $S = S \cup R_{svm^2}$ and $R = R \cup R_{svm^2}$.
(b) If the terminal node to which the leaf is attached is a one-class SVM node, apply Algorithm 5 to the terminal node for one-class support vector rule R_{svm^1} extraction, $S = S \cup R_{svm^2}$ and $R = R \cup R_{svm^1}$.

Step 4: Labeled samples in $\mathcal{X}_{train}$ to fall into a region of each of the above obtained rules in R.

Step 5: If the set of sample in a certain rule region is a subset of samples covered by another rule, this rule is removed.

Step 6: Repeat Step 3 until no rule is removed from R, output R, C, and S.

Note that, Algorithm 7 is run in the form of DFS SVMT-Rule and BFS SVMT-Rule, depending on the choice of SVMT in Step 1.

5 Experiments and Applications

In our experiments, algorithms were implemented in Matlab Version 6.5, run on Pentium 4 PC, 3.0 GHz 512 MB RAM, and comparison tests were performed in the environment of Neucom 2.0 [28]. For extracting rules over SVMT, one-class SVM is set with a RBF $\beta = 3.5$ kernel, and binary SVM with a standard linear kernel. All SVMs have the same penalty coefficient of 0.1. $\wp$ is assigned as a K-means clustering partitioning function, and all 2-SVMTs use the same default parameters of $\rho_0 = 2$ and $\xi = 0.02$.

Because Polynomial SVMs perform better than linear SVM on most classification tasks [34], for a simple validity verification we merely compare the SVMT-Rule with the SVM-Rule, and the rule extraction from a trained second-order polynomial single SVM [6]. Note that all SVMs are set with the same penalty coefficient.

To setup the class-imbalance criterion for comparing the input bias (class data distribution) with the output bias (class classification accuracy) of the system, we define the class bias ratio (CBR) as,

$$1 - min(class\,1, class\,2)/max(class\,1, class\,2), \tag{27}$$

where a larger CBR value means a more imbalanced of the class data distribution, or the class classification accuracy.

5.1 Synthetic Dataset

We first experimented the SVMT-Rule with a synthetic dataset for classification. Figure 5 shows the distribution of the dataset, where class 1 and class 2 data are both in a 2D Gaussian distribution, and the number of class 2 samples is approximately 1/10 of class 1, which follows that the CBR of this dataset in terms of class data distribution is 0.9.

Constructing SVMT over the dataset of Fig. 5, Fig. 6 gives an example of DFS-SVMT and BFS-SVMT generated by Algorithm 1 and Algorithm 2, respectively. The root node $P1$ represents the data partitioning function at scale ρ_0, $P2$ for the data partitioning at scale ρ_1, so on so forth. Every one-class node here is denoted as a circle labeled with class "1" or "2," and SVM node denoted as a ellipse circled "SVM" label. Each SVM node has two sons, which means that each SVM node provides, class 1 and class 2, two decision

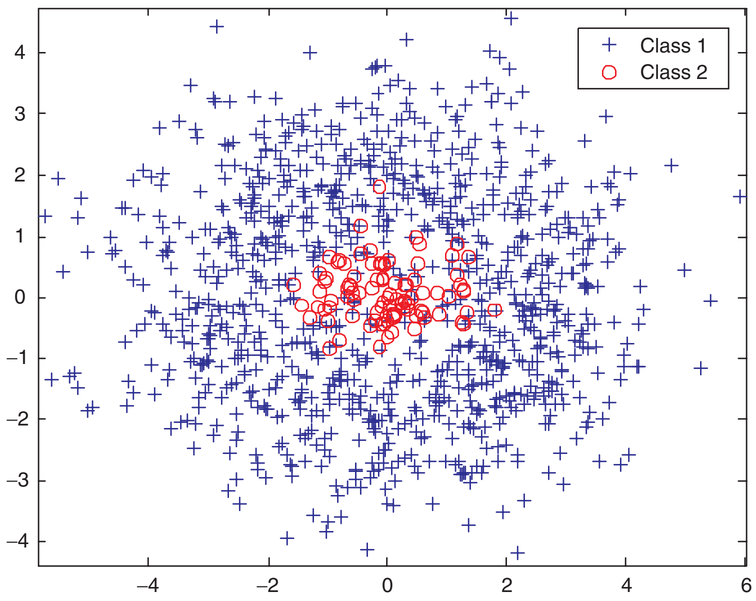


Fig. 5. Two-dimensional synthetic data for classification, CBR = 0.9

choices. They are symbolized the same as the one-class node in the figure, but they are not counted here as one-class node. The DFS-SVMT is seen to have 8 SVM nodes, 14 one-class nodes, and the BFS-SVMT has 8 SVM nodes, 24 one-class nodes. The number of one-class nodes is several times of SVM nodes, suggesting that SVMT makes decision depending on the decision tree more than the SVM.

In the construction of SVMT, the data is zoomed in recursively by a supervised data partitioning. Figure 7b shows the resulting hyper-plane from Algorithm 2 as the data partitioning step goes to the second iteration. Compared to the single SVM hyper-plane in Fig. 7a, the SVMT hyper-plane in Fig. 7b is a combination of the part of data partitioning boundary estimated in Fig. 7c and two pieces of short SVM hyper-planes. It turns out that SVMT conducts classification using less support vectors from SVM, but more decision boundaries from data partitioning. In other words, SVM is not used unless it is absolutely necessary. By Algorithm 7, a set of SVMT rules are encoded over the constructed SVMTs, example rules are given below, where the first two rules are encoded directly from the obtained SVM tree structure, and the remaining rules are from a reginal SVM.

if $1.89 <= x1 <= 4.95$ and $-5.0 <= x2 <= -2.3$ then [x1 x2] in cls.1
if $-5.00 <= x1 <= -2.00$ and $2.42 <= x1 <= 5.00$ then [x1 x2] in cls.1
if $-5.00 <= x1 <= -2.00$ and $-2.42 <= x1 <= 2.30$ and $x1^2 * 0.34 + x2 * 1.87 + 0.987 > 0$ then [x1 x2] in cls. 2.
if $-5.00 <= x1 <= -2.00$ and $-2.42 <= x1 <= 2.30$ and $x1^2 * 0.34 + x2 * 1.87 + 0.987 < 0$ then [x1 x2] in cls. 1.

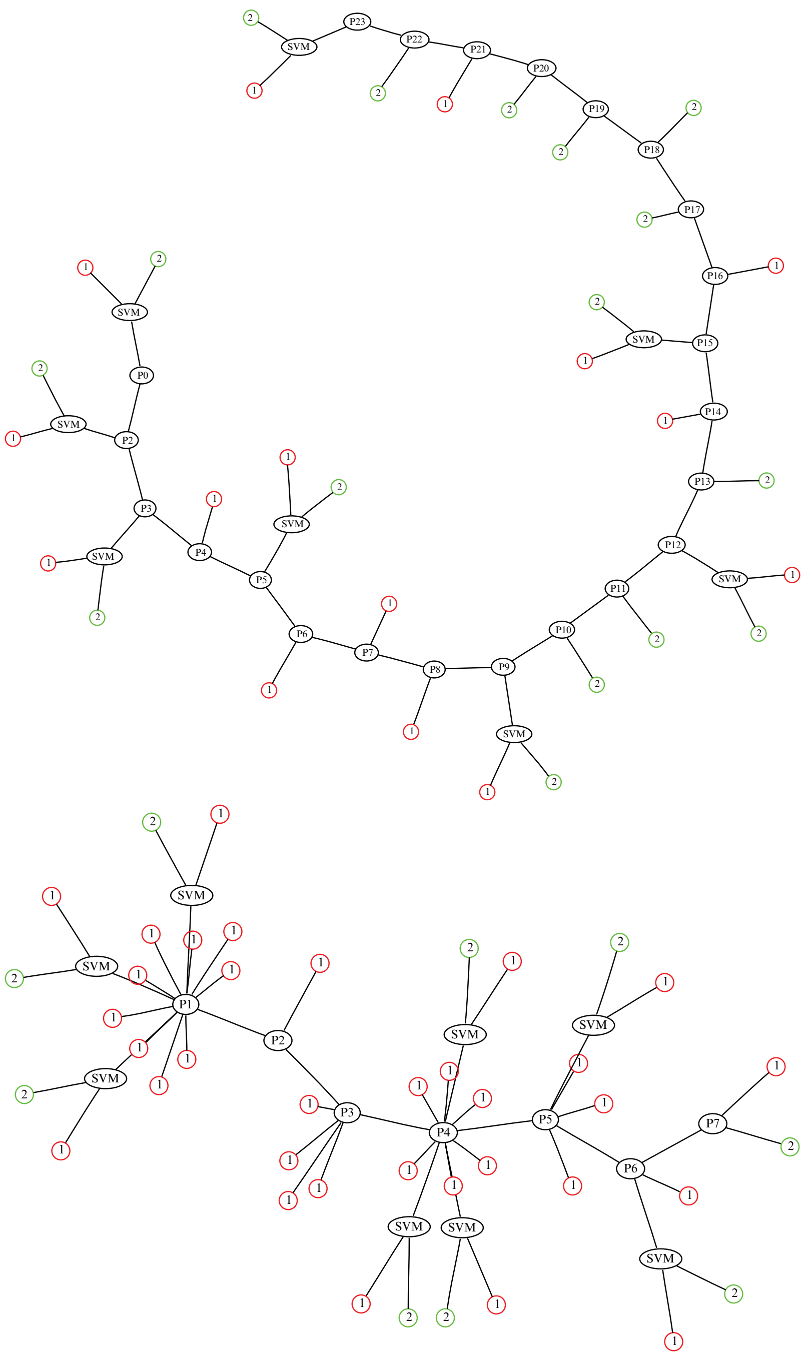


Fig. 6. Example of DFS-SVMT (*top*) and BFS-SVMT (*bottom*) from SVMT rule extraction over Fig. 5 dataset

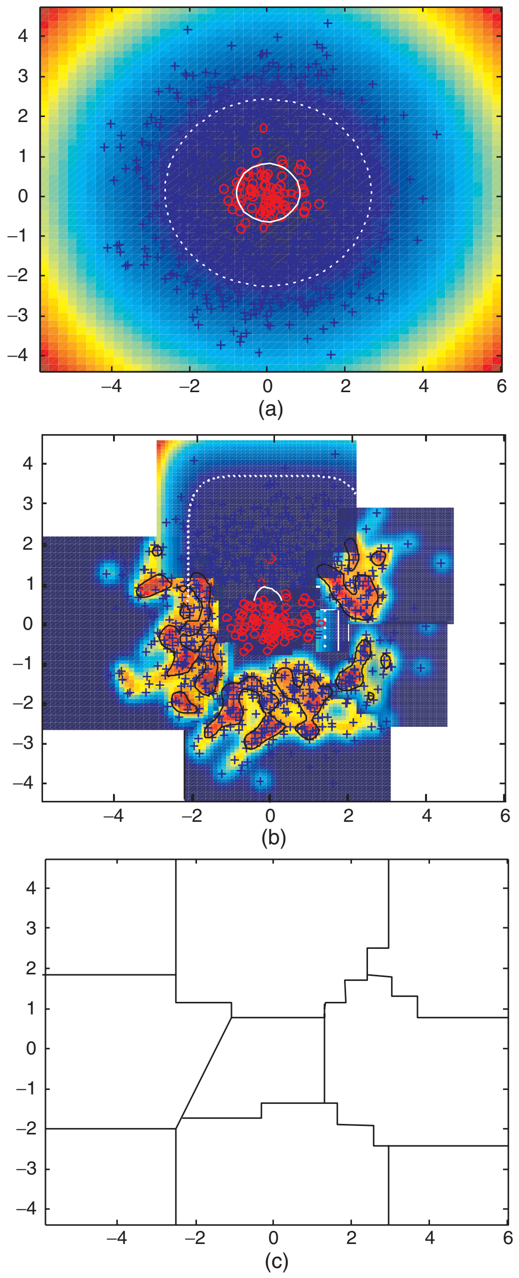


Fig. 7. (**a**) Single SVM hyper-plane, (**b**) SVMT hyper-plane, (**c**) the data partitioning boundary part of SVMT hyper-plane

Table 1. The results of comparison between SVM-Rule and SVMT-Rule on the classification of Fig. 5 dataset

Measurements	SVM-Rule	DFS SVMT-Rule	DFS-SVMT	BFS SVMT-Rule	BFS-SVMT
No. of rules	58	23 (8/15)	–	33(9/24)	–
No. of SVs	32	16	16	18	18
Ave. Acc.	91.3%	92.4%	92.7%	90.0%	90.1
Cls1 Acc.	100%	94.9%	94.8%	91.2%	92.0
Cls2 Acc.	0.0%	66.3%	70.5%	76.9%	79.0
CBR	1.0	0.30	0.25	0.16	0.14

Table 1 compares the SVMT-Rule with SVM-Rule and the original SVMT on the classification of the above synthetic dataset, based on the following six measurements:

(1) The number obtained rules (N. of Rules)
(2) The number of support vectors (N. of SVs)
(3) The average general classification accuracy (Ave. Acc.)
(4) The classification accuracy of class 1 (Cls1 Acc.)
(5) The classification accuracy of class 2 (Cls2 Acc.)
(6) The class bias ratio in terms of class classification accuracy (CBR)

For the CBR 0.9 dataset in Fig. 5, SVMT-Rules come up with a CBR ≤ 0.3 class classification accuracy while the SVM-Rule gives a completely imbalanced output with $CBR = 1.0$. It is distinct that SVMT-Rules have profited from the outstanding generalization ability of SVMTs as both statistics are shown to be quite similar in Table 1. The two BFS type SVMT methods (i.e., BFS SVMT-Rule and BFS-SVMT) perform even better than the DFS type SVMT methods (i.e., DFS SVMT-Rule and DFS-SVMT), this suggests that BFS is able to fit the data more appropriated than DFS for constructing SVMT.

5.2 Cancer Diagnosis

We have implemented the BFS SVMT-Rule technique in cancer diagnosis and decision support. Table 2 gives the seven well-known cancer microarray datasets with a two-class classification problem. As seen in the table, most datasets were biased in the sense of having an unbalanced number of patients in the two classes, e.g., normal vs. tumor.

For the three datasets that had an independent validation dataset available, we selected 100 genes from the training dataset by a typical t-test gene selection [26], and used them on the validation dataset. For the remaining four datasets, we verify the method using tenfold cross-validation, removing randomly one tenth of the data and then using the remainder of the data as

Table 2. Cancer datasets used for testing the algorithm

Cancer	Class 1 vs. class 2	Genes	Train data	Test data	Ref.
Lymphoma(1)	DLBCL vs. FL	7129	(58/19)77	–	[19]
Leukaemia	ALL vs. AML	7129	(27/11)38	34	[20]
CNS cancer	Survivor vs. failure	7129	(21/39)60	–	[21]
Colon cancer	Normal vs. tumor	2000	(22/40)62	–	[22]
Ovarian cancer	Cancer vs. normal	15154	(91/162)253	–	[23]
Breast cancer	Relapse vs. non-relapse	24482	(34/44)78	19	[24]
Lung cancer	MPM vs. ADCA	12533	(16/16)32	149	[25]

Columns for training and validation data show the total number of patients. The numbers in brackets are the ratios of the patients in the two classes

Table 3. The results of comparison between SVM-Rule and BFS SVMT-Rule on seven cancers diagnosis

Dataset	SVM-Rule			BFS SVMT-Rule			BFS SVMT
	Acc. (%)	No. of rules	No. of SVs	Acc. (%)	No. of rules	No. of SVs	Acc. (%)
Lymphoma(1)	89.6	233	60	80.8	221	38	89.6
Leukaemia	73.5	245	29	94.1	108	20	94.1
CNS cancer	61.7	234	37	62.9	278	26	65.0
Colon cancer	71.3	205	48	75.3	202	32	77.3
Ovarian cancer	96.5	178	34	98.4	198	21	97.3
Breast cancer	52.6	269	62	78.9	206	33	78.9
Lung cancer	94.6	171	23	98.5	188	17	99.3

Table 4. The statistics of rule extraction for fraud detection

Method	No. of Rules	No. of SVs	Rule/per SV	Accuracy (%)
SVM-Rule	269	35	7.6	78.5
DFS SVMT-Rule	98	17	5.7	89.3
BFS SVMT-Rule	123	19	6.4	92.3
C4.5-Rule	176	–	–	80.8

a training set. For each fold, we similarly selected 100 genes over the training set, then applied the obtained 100 genes to the corresponding testing set. The comparison of SVMT-Rule to single SVM-Rule and the original SVMT on seven cancer diagnosis is shown in Table 3, where rule extraction is measured in terms of diagnosis accuracy, the number of rules, and the number of involved support vectors, respectively. For comparison, Table 4 also shows the predictive accuracy of SVMT.

Although the SVMT-Rule does not perform better than the SVMT, it is still impressive that Table 3 indicates the prior generalization ability of SVMT-Rule to SVM-Rule is about 10% $(((0.808 - 0.896)/(1 - 0.808) + (0.941 - 0.735)/(1 - 0.735) + \cdots + (0.985 - 0.946)/(1 - 0.946))/7 = 0.1096)$.

More importantly, the comprehensibility of BFS SVMT-Rule is clearly improved as compared to SVM-Rule because the average number of support vectors for SVMT-Rule is only about 60% that of SVM-Rule. For CNS and Lung Cancer, BFS SVMT-Rule is found with less support vectors, but even more rules than SVM-Rule. This inconsistency owes to the reason that these two cancers are more difficult than the other type of cancers for a correct knowledge representation, where the BFS SVMT has allocated more efforts on decision tree spanning.

5.3 Fraud Detection

The cell phone fraud often occurs in the telecommunication industry. We have investigated a fraud detection using SVMT-Rule rule extraction. The database was obtained from a mobile telecom company. It recorded 1 years action data of 53,696 customers. The historical data on fraud includes 15 tables. Among them, three core tables for the topic are IBS_USERINFO (customer profile), IBS_USRBILL (customer payment information), and IBS_USERPHONE (customer action) and all others are the additional tables.

Fraud detection basically is a binary classification problem, where customers are divided into two classes: fraud or nonfraud. The aim of fraud detection is to distinguish the fraud customers from the remaining honest customers. The method of fraud detection is to analyze the historical data, which is a typical data mining procedure as follows:

Step 1: Data cleaning, to purge redundant data for a certain fraud action analysis
Step 2: Feature selection and extraction, to discover indicators corresponding to changes in behavioral indicatives of fraud
Step 3: Modeling, to determine fraud pattern by classifier and predictor
Step 4: Fraud action monitoring and prediction, to issue the alarm

Step 2 extracts the eight salient features to measure the customer payment behaviors using PROC DATA of SAS/BASE and combine the customer profile (IBS USERINFO) and customer action (IBS BILL) to one table (IBS_USERPERFORMANCE). Table IBS_USERPERFORMANCE has 53,696 records and nine attribute items.

To discover the fraud patterns, BS_USERPERFORMANCE data is divided into a training set and a test set, in which the training set contains 70% of 53,696 records, and the test dataset is the remaining 30% of 53,696 records. Table 4 lists the statistical results of rule extraction using SVMT-Rule, SVM-Rule, and C4.5-Rule for the fraud detection. As expected, the SVMT-Rule shows a more accurate fraud detection than SVM-Rule, as well as C4.5-Rule, where BFS SVMT-Rule performs slightly better than DFS SVMT-Rule at the price of that more support vectors and more rules are used. It is surprising that the number of rules from SVMT-Rule is only about half of the number from SVM-Rule. This could be explained that the SVMT-Rule suits better a large database, where the decision tree part of SVMT plays an absolutely more important role in encapsulating data for rule extraction.

6 Discussions and Conclusions

The SVMT aggregates the decision tree and the support vector machine methods in an SVM classification tree, facilitates the two-side rule extraction by selecting the simple part of the problem for a comprehensible decision tree representation, and leaving the remaining difficult part with support vector machine approximation. For extracting rules, SVMT-Rule is introduced in the form of DFS SVMT-Rule and BFS SVMT-Rule, which encodes the knowledge of rules by an induction over the decision tree of SVM, and the interpolation of the support vectors from local SVMs.

The experimental results and applications have shown that the SVMT-Rule, compared to single SVM rule extraction, has the following desirable properties:

(1) SVMT-Rule is more accurate, and especially more robust for the classification of data with an class-imbalanced data distribution.
(2) SVMT-Rule has a better comprehensibility for rule extraction because only about half the number of single SVM support vectors are kept in SVMT rules.
(3) The BFS SVMT-Rule commonly outperforms DFS SVMT-Rule.

SVMT-Rule is also able to apply to a multiclass case by extending the two-class SVMT to a m-class ($m \geq 2$) SVMT. One straightforward way is to do “one-to-one” or “one-to-all” SVMT integration. However, this approach creates a new problems of decision forests [27].

In the principle of SVM aggregation, the construction of a m-SVMT is to decompose an m-class task into a certain number of $1 - m$ classes regional tasks as,

$$\mathcal{T}_{m-SVMT}(\wp^{<\rho>}, f_{Svm^{<2>}}, \ldots, fSvm^{<m>}, f_{Sgn^{<1>}}, \ldots, f_{Sgn^{<m>}}, \boldsymbol{x}) \quad (28)$$

where $\wp^{<\rho>}, \rho \geq 2$ represents here a multi-split data partitioning function. Similarly, $f_{Svm^{<i>}}, 2 \leq i \leq m$ represents a i-class SVM classification function, and $f_{Sgn^{<j>}}, 1 \leq i \leq m$ represents a jth-class one-class SVM decision function.

Accordingly, the classification function of a m-SVMT can be written as,

$$\hat{f}(\boldsymbol{x}) = \begin{cases} f_{Sgn^{<i>}} & if \qquad \wp(\boldsymbol{x}) \in class\ i, \\ f_{Svm^{<j>}} & otherwise, \end{cases} \quad (29)$$

where $1 \leq i \leq m$ and $2 \leq i \leq m$.

The proposed SVMT-Rule method has improved the comprehensibility of SVM rule extraction by reducing the number of support vectors effectively, however there are still a few support vectors left in the rule of the SVMT-Rule. To further improve the comprehensibility of the SVMT-Rule, it is suggested that a symbolic rule interpolation [32] could be developed for SVMT-Rule in the future.

7 Acknowledgements

The research presented in the chapter was partially funded by the New Zealand Foundation for Research, Science and Technology under the grant: NERF/AUTX02-01, and by the knowledge Engineering and Discovery Research Institute (KEDRI) of Auckland University of Technology (`http://www.kedri.info`).

References

1. H. Nunez, C. Angulo, and A. Catala (2002) Rule-extraction from Support vector Machines. The European Symposiumon Aritificial Neural Netorks, Burges, pp. 107–112.
2. Y. Zhang, H. Y. Su, T. Jia, and J. Chu (2005) Rule Extraction from Trained Support Vector Machines, PAKDD 2005, LANI3518, pp. 61–70, Springer-Verlag Berling Heidelberg.
3. Lipo Wang and Xiuju Fu (2005) Rule Extraction from Support Vector Machine. In: Data Mining with Computational Intelligence, nced Information and Knowlegde Processing, Springer Berlin Heidelberg.
4. N. Barakat and Andrew P. Bradley (2006) Rule Extraction from Support Vector Machines: Measuring the Explanation Capability Using the Area under the ROC Curve. In The 18th International Conference on Pattern Recognition (ICPR'06), August, 2006, Hong Kong.
5. Glenn Fung, Sathyakama Sandilya, and Bharat Rao (2005) Rule Extraction for Linear Support Vector Machines, KDD2005, August 2124, 2005, Chicago, Illinois, USA.
6. Xueju Fu, C. Ong, S. Keerthi, G. G. Huang, and L. Goh (2004) Proceedings of IEEE International Joint Conference on Neural Networks, Vol. 1, no. 25–29 July 2004, pp. 291–296.
7. V. Vapnik, Estimation of dependences based on empirical data. Springer-Verlag, 1982.
8. V. Vapnik, The nature of statistical learning theory, New York: Spinger-Verlag, 1995.
9. C. Cortes and V. Vapnik, "Support vector network," Machine learning, vol. 20, pp. 273–297, 1995.
10. Shoning Pang, D. Kim, and S. Y. Bang, "Membership authentication in the dynamic group by face classification using SVM ensemble," Pattern Recognition Letters, Vol. 24, pp. 215–225, 2003.
11. Shaoning Pang, SVM Aggregation: SVM, SVM Ensemble, SVM Classification Tree, IEEE SMC eNewsletter Dec. 2005. http://www.ieeesmc.org/Newsletter/Dec2005/R11Pang.php

12. Shaoning Pang, D. Kim, and S. Y. Bang, “Face Membership Authentication Using SVM Classification Tree Generated by Membership-based LLE Data Partition,” IEEE Trans. on Neural Network Vol. 16, no. 2, pp. 436–446 Mar. 2005.
13. J.C. Schölkopf, J.C. Platt, J. Shawe-Taylor, A.J. Smola, and R.C. Williamson, “Estimating the support of a high-dimensional distribution,” Technical report, Microsoft Research, MSR-TR-99-87, 1999.
14. Tax D.M.J. (2001) One-class Classification, concept-learning in the absence of counter-examples. Ph.D. Thesis.
15. Tax D.M.J. and Duin R.P.W. (2001) Combining one-class classifiers. LNCS 2096: 299–308.
16. Xu Y. and Brereton R. G. (2005) Diagnostic pattern recognition on gene expression profile data by using one-class classifiers. J. Chem. Inf. Model. 45: 1392–1401.
17. J. R. Quinlan, C4.5: Programs for Machine Learning. San Mateo, CA: Morgan Kaufmann, 1993.
18. Hyun-Chul Kim, Shaoning Pang, Hong-Mo Je, Daijin Kim,and Sung Yang Bang, “Constructing support vector machine ensemble,” Pattern Recognition, vol. 36, no. 12, pp. 2757–2767.
19. M. A. Shipp, K. N. Ross, et al., (2002) “Supplementary Information for Diffuse large B-cell lymphoma outcome prediction by gene-expression profiling and supervised machine learning,” *Nature Medicine*, 8(1):68–74, (2002).
20. T. R. Golub, “Toward a functional taxonomy of cancer,” *Cancer Cell*, 6(2):107–8, (2004).
21. S. Pomeroy, P. Tamayo, et al., “Prediction of Central Nervous System Embryonal Tumour Outcome Based on Gene Expression,” *Nature*, 415(6870), 436–442, 2002.
22. U. Alon, N. Barkai, et al., *Broad Patterns of Gene Expression Revealed by Clustering Analysis of Tumor and Normal Colon Tissues Probed by Oligonucleotide Arrays*, Proc Natl Acad Sci, USA, (1999).
23. E. F. Petricoin, A. M. Ardekani, et al., “Use of Proteomic Patterns in Serum to Identify Ovarian Cancer,” *Lancet*, 359, 572–577, (2002).
24. Van’t Veer L. J. et al., “Gene expression profiling predicts clinical outcome of breast cancer,” *Nature* 415:530–536, (2002).
25. G. J. Gordon, R. Jensen, et al., Translation of Microarray Data into Clinically Relevant Cancer Diagnostic Tests Using Gene Expression Ratios in Lung Cancer And Mesothelioma,” *Cancer Research*, 62, 4963–4967, (2002).
26. Dudoit, S., Yang, Y. H., Callow, M. J., and Speed, T. P. (2002) Statistical methods for identifying differentially expressed genes in replicated cDNA microarray experiments. Stat. Sinica, 12, 111–139.
27. Tin Kam Ho, “The random subspace method for constructing decision forests Tin Kam Ho,” IEEE Transactions on Pattern Analysis and Machine Intelligence, 20(8), pp. 832–844, Aug. 1998.
28. NeuCom - A Neuro-computing Decision Support Enviroment, Knowledge Engineering and Discovery Research Institute, Auckland University of Technology, www.theneucom.com.
29. H. Nez, C. Angulo and Andreu Catal, “Hybrid Architecture Based on Support Vector Machines,” Lecture Notes in Computer Science Volume 2686, in Book Computational Methods in Neural Modeling, pp. 646–653, 2003.

30. Z. H. Zhou and Y. Jiang, “Medical Diagnosis with C4.5 Rule Preceded by Artificial Neural Netowrk Ensemble,” IEEE Trans. on Information Technology in Biomedicine, 7(1):37–42, 2003.
31. Yixin Chen and J. Z. Wang, J. Z., “Support vector learning for fuzzy rule-based classification systems, IEEE Transactions on Fuzzy Systems 11(6), pp. 716–728, 2003.
32. H. Nunez, C. Angulo, and A. Catala, “Support vector machines with symbolic interpretation,” Proceedings. VII Brazilian Symposium on Neural Networks, pp. 142–147, 11–14 Nov. 2002.
33. W. Duch, R. Setiono, and J. M. Zurada, “Computational intelligence methods for rule-based data understanding,” Proc. of the IEEE, 92(5), pp. 771–805, May 2004.
34. Shaoning Pang, Dajin Kim, S. Y. Bang (2001) Fraud Detection Using Support Vector Machine Ensemble. ICONIP2001, Shanghai, China.
35. Y. S. Chen and T. H. Chu, “A neural network classification tree,” Proc. IEEE Int. Conf. Neural Networks, Vol. 1 , Nov. 27- Dec. 1, 1995, pp. 409–413.

Prototype Rules from SVM

Marcin Blachnik[1] and Włodzisław Duch[2]

[1] Division of Computer Methods, Department of Electrotechnology, The Silesian University of Technology, ul. Krasińskiego 8, 40-019 Katowice, Poland
marcin.blachnik@polsl.pl

[2] Department of Informatics, Nicolaus Copernicus University, Grudziądzka 5, Toruń, Poland Google: W. Duch

Summary. Prototype based rules (P-rules) are an alternative to crisp and fuzzy rules, moreover they can be seen as a generalization of different forms of knowledge representation. In P-rules knowledge is represented as set of reference vectors, that may be derived from the SVM model.

The number of support vectors (SV) should be reduced to a minimal number that still preserves SVM generalization abilities. Several state-of-the-art methods that reduce the number of support vectors are compared with a new approach, taking into consideration possible interpretation of retained support vectors as the basis for P-rules.

1 Why Prototype-Based Rules?

Propositional logical rules may not be the best way to understand the class structure of data describing some objects or states of nature. The best explanation may differ depending on the problem, the type of questions and the type of explanations that are commonly accepted in a given field. Although most research has focused on propositional logical rules [14, 19] their expressive powers have serious limitations. For example, a simple majority voting can be expressed using the "majority is for it" concept that is easy to formulate using M-of-N threshold rules. Given n binary $x_i = 0, 1$ answers the rule $\sum_{i=1}^{n} x_i > 0.5n$ is an elegant expression of such concept and is impossible to state directly in propositional form, leading to $\binom{n}{n/2}$ terms. This type of rules may be regarded as a particular form of similarity or prototype-based rules. In the voting example the similarity to the "all for it" prototype $\mathbf{A}$, that is a vector with all $a_i = 1$, has to be greater than $n/2$ in the Hamming distance sense, $||\mathbf{A} - \mathbf{X}|| < n/2$. Cognitive psychology experiments proved that human categorization of natural objects and states of nature is based on memorization of numerous examples and creation of prototypes that are abstractions of these examples [34]. Propositional logical rules are prevalent in abstract sciences but in real life they are rarely useful, their use being restricted to enumeration of small number of nominal values, or one or two continuous features with corresponding thresholds. In real life "intuitive understanding" is used more often, reflecting experience, i.e. memorized examples of patterns combined with various similarity measures that allow for their comparison and evaluation.

Decision borders between different categories produced by propositional rules are simple hyperboxes. Univariate decision trees provide even simpler borders based on hierarchical reduction of decision regions to half-spaces and hyperboxes. Using similarity to prototypes quite complex decision regions may be created, including hyperboxes and fuzzy decision regions. Some of these decisions may be difficult to describe using linguistic statements and thus may server as a model of intuition. One may argue that comprehensibility of rules is lost in this way, but if similarity functions are sufficiently simple interpretation may in fact be quite easy. For example, interpretation of the $||\mathbf{A} - \mathbf{X}|| < n/2$ rule is quite obvious. Other voting rules may easily be expressed in the same way, including polarization of opinions around several different issues. Weighting evidence before decision is made requires non-trivial aggregation function to combine all available evidence, and similarity or dissimilarity functions are the most natural way to do it. Despite these arguments the study of prototype-based rules has been much less popular than of the other forms of rules.

Similarity-Based Methods (SBM) [8,13] are quite popular in pattern recognition and data mining. The framework for construction of such methods enables integration of many methods for data analysis, including neural networks [12], probabilistic and fuzzy methods [15], kernel approaches and many other methods [32]. One of the most exciting possibility that such framework offers is to build the simplest accurate method on demand, in a meta-learning scheme, searching for the best model in the space of all similarity-based methods [18]. This family of methods includes also prototype-based rules (P-rules) [17] that are more general than fuzzy rules (F-rules), crisp propositional rules (C-rules) and M-of-N rules, including them as special cases. All methods covered by the SBM framework represent knowledge as a set of prototypes or reference vectors, adding appropriate similarity metrics and the aggregation procedures that combine information from different prototypes giving the final output. Several similarity-based transformations may be done in succession, creating higher-order SBM models. Prototype based rules are based on the SBM framework, but their aim is to represent the knowledge hidden in the data in the most comprehensible way. This goal is obtained by reducing the number of prototype vectors (prototype selection), minimizing the number of features used to create final model and using simple similarity metrics.

One of the most important advantages of P-rules is their universality. They enable integration of different type of rules, depending on the similarity function associated with each prototype: classical crisp rules result form Chebychev distance, fuzzy rules (F-rules) from any separable similarity metrics [16]. P-rules can also represent M-of-N rules in a natural way using prototype threshold rules [2, 21], adding the distance to a prototype as one of the coordinates. Such rules often give very simple interpretation of data, for example a single prototype threshold rule gives over 97.5% accuracy on a well known Wisconsin Breast Cancer dataset [21]. Thus P-rules provide most general form of knowledge representation.

Two general types of P-rules are possible, the Nearest Neighbor Rules (PN-rules), and the prototype threshold rules (PT-rules), introduced in the next section. In the same section the use of support vectors as prototypes is discussed. Reduction of the number of support vectors (SVs) and methods of searching for informative prototypes are described in Sects. 2.3 and 3, while numerical examples are presented in Sect. 3.4. Perspectives on the use of support vector machines for P-rule extraction conclude this paper.

2 P-Rules and Their Interpretation

Prototype rules are based on analysis of similarity between objects and prototypes that are used as a reference. In its most general form [13, 32] objects (cases) $\{\mathbf{O^i}\}$, $i = 1..n$ do not need to be represented by numerical features, a kernel (or a set of different kernels that provide "receptive fields" that stress different perspectives) estimating (dis)similarity is sufficient $K_{ij} = K(\mathbf{O^i}, \mathbf{O^j})$ to characterize such objects. Selecting some of these objects as prototypes an object $\mathbf{O}$ is represented by n-dimensional vector $\mathbf{p}(\mathbf{O}) = \mathbf{K}^p$. Alternatively, each object is represented by N feature values. In the first case features come from evaluation of similarity and may be created for quite complex and diverse objects (such as proteins or whole organisms), for which a common set of features is hard to define. Below it is assumed that all objects are described by vectors in some feature space.

A single prototype $\mathbf{p}$ with associated similarity function $S(\cdot, \mathbf{p})$ defines for a given threshold θ a subspace $\mathcal{S}_p$ of vectors $\mathbf{x}$ for which $S(\mathbf{x}, \mathbf{p}) < \theta$. This subspace is centered at the position of the prototype $\mathbf{p}$ and may have different shapes, depending on the similarity function. Such interpretation defines a crisp logical rule for the new feature $x_p = S(\mathbf{x}, \mathbf{p})$. In this case the antecedent part of a P-rule uses similarity to a single prototype and the class label of that prototype (in classification tasks) is the consequence part.

$$\text{If } S(\mathbf{x}; \mathbf{p}) > \theta \text{ Then } C(\mathbf{x}) = C(\mathbf{p}) \tag{1}$$

The similarity value may be used to estimate confidence factor for such rule. The rescaled difference $\mu_p(\mathbf{x}) = S(\mathbf{x}, \mathbf{p}) - \theta$ may obviously be interpreted as a fuzzy membership function defining the degree to which vector $\mathbf{x}$ belongs to the fuzzy subspace $\mathcal{S}_p$. Many similarity functions are separable in respect to all features:

$$S(\mathbf{x}; \mathbf{p}, \sigma) = \prod_i S(x_i, p_i; \sigma_i) \tag{2}$$

where $\mathbf{x} = [x_1, x_2, \ldots, x_n]^T$ and $\mathbf{p} = [p_1, p_2, \ldots, p_n]^T$ are n-dimensional vectors, and $S(\cdot)$ is similarity function.

P-rules with separable similarity functions can be interpreted as fuzzy rules (F-rules) with a product as a fuzzy *and* aggregation operator. Linguistic interpretation of F-rules relies on semantics of linguistic values assigned to each linguistic variable as adjectives describing the membership functions. Such representation is sensitive to context. Good example of this context dependence is an adjective *high* that may describe objects of different types, for example *a person*, but even in this case different kinds of people: kids, women or basketball players will require different membership function representing variable "high". Thus indirectly fuzzy rules have to rely on prototypes of objects or concepts to define the context, but since in fuzzy rules this context is not explicitly represented confusion is quite likely. P-rules make this reliance explicit always pointing to prototypes of particular concepts, allowing each concept to be decomposed into independent features that may be treated as linguistic values in the fuzzy sense.

2.1 Types of P-Rules

Two distinct types of P-rules are:

- Prototype Threshold Rules (PT-rules), where each prototype $\mathbf{p}_i$ has an associated threshold θ_i value and i-th rule is written as:

$$\text{If } S(\mathbf{x}, \mathbf{p}_i) > \theta_i \text{ Then } C(\mathbf{x}) = C(\mathbf{p}_i) \tag{3}$$

  where $C(\cdot)$ is a function returning class labels or some other information associated with the prototype.
- Nearest Neighbor Rule (PN-rules), where the most similar prototype is selected:

$$\text{If } k = \arg\max_i S(\mathbf{x}, \mathbf{p}_i) \text{ Then } C(\mathbf{x}) = C(\mathbf{p}_k) \tag{4}$$

  so the output value depends on the internal relations between prototypes.

More general form of PN-rules is used by the Generalized Nearest Prototype Classifier [25]. From the rule-based perspective it is defined as: If $\mathbf{x}$ is similar to $\mathbf{p}_i$ then it is of the same class with some support w_i:

$$\text{If } w_i = S(\mathbf{x}, \mathbf{p}_i) \text{ Then } C(\mathbf{x}) = C(\mathbf{p}_i) \text{ with support } w_i \tag{5}$$

where $\mathbf{P} = [\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_v]^T$ is set of v prototype vectors, and w_i is support for the conclusion of the i-th rule. The final decision of the set of such rules is obtained as:

$$C(\mathbf{x}) = A(w_i, C(\mathbf{p}_i)) \tag{6}$$

where $A(\cdot)$ is an aggregation operator, which joins conclusions of individual P-rules.

2.2 Support Vectors as Prototypes

The SVM model defines a hyperplane that can be used for linear discrimination in the feature space:

$$\mathbf{\Psi} = \sum_{i=1}^{m} \gamma_i \phi(\mathbf{x}_i) \tag{7}$$

where $\phi(\mathbf{x})$ is function that maps vectors from n dimensional input space to some feature space $\mathcal{F}$. Since scalar products are sufficient to define linear models in the ϕ-transformed space kernels are used to represent these products in the original feature space. Decision function is in this case defined as:

$$f(\mathbf{x}) = \sum_{i=1}^{m} \gamma_i y_i K(\mathbf{x}, \mathbf{x}_i) + b \tag{8}$$

where m is the number of support vectors $\mathbf{x}_i$ with non-zero γ_i coefficients (Lagrangian multipliers), $K(\mathbf{x}, \mathbf{x}_i)$ is the kernel function, and $y_i = C(\mathbf{x_i}) = \pm 1$ are the class labels.

This model may be expressed as a set of PN-rules with weighted aggregation $A(\cdot)$ (5) as a sum from $i = 1$ to m, replacing the kernel with a similarity function $S(\cdot, \cdot)$ and defining support for a rule as $w_i = \alpha_i S(\mathbf{x}, \mathbf{p_i}; \alpha)$. Similar ideas have also been considered from the fuzzy perspective by Chen and Wang [5] who interpret SVM model as a fuzzy rule based system. In their paper they introduced Positive-Definite Fuzzy Classifiers using the Takagi Sugeno (TS) fuzzy inference system [37], adopting this model to extract fuzzy rules from support vector machines. However, in their solution comprehensibility and model transparency, the most important properties of any rule bases system, are lost. As stated in [19], logical rules are useful only if they are simple and accurate, otherwise there is no point in extracting rules from black box systems that works well because no additional understanding is gained by creation of many complex rules. The goal of comprehensibility and transparency can be achieved only when small number of support vectors (SV) can be defined, or when SVM decisions can be replicated with another simpler rule-based model. These two strategies have been studied by many research groups. The first leads to methods aimed at reduction of the number of support vectors through removing approximately linearly dependent kernels in the SV set. The second one leads to the "Reduced Set" methods aimed at reconstruction of the SVM hyperplane defined in the kernel feature space with smaller number of kernel functions.

The initial idea behind the kernel reduction methods was to speed up the decision process, but these methods can obviously be also used to understand data using small number of P-rules. These two approaches differ in the way support vectors are used. SVs are vectors that define or lie within the margin, that is they are close to the decision hyperplane. Those SVs that are outside of the margin on the wrong side should be removed, as they are cases that cannot be correctly classified and should not be used to create rules. Reducing linear dependencies removes some of the original SVs but will leave other SVs intact. In the "Reduced Set" approach new support vectors are not selected from the training examples but may be defined anywhere in the input space.

2.3 Removing Linear Dependencies Among Support Vectors

Many numerical methods of removing linear dependencies from the kernel matrix $K(\mathbf{x}_i, \mathbf{x}_j)$ may be defined. For smooth kernels the problem may also be analyzed in the feature space, because it is created by vectors that are too similar to each other. Therefore clusterization techniques may be used to select representative vectors that are sufficiently distinct to avoid problems with linear dependencies. In some applications cluster centers may replace original vectors.

SVM approach based on quadratic programming has a unique solution, avoiding the problem of local minima and initialization of parameters that neural network algorithms have to face. Still there are some differences in SVM implementations that use different quadratic programming solvers. Solutions obtained with SMO [33], SVM Light [24], SVMTorch [6] or other SVM methods slightly differ from each other. On the other hand even if identical decision function are obtained different support vectors may be selected during the optimization procedure. Such situation is bound to happen when SVs are linearly dependent. This observation leads to a reduction of the number of support vectors, as studied by Downs et al. in [7] in the algorithm referred below using RLSV acronym (removed linearly-dependent support vectors). Linear dependence in the kernel space Φ can be written as:

$$\phi(\mathbf{x}_k) = \sum_{\substack{i=1 \\ i \neq k}}^{m} q_i \phi(\mathbf{x}_i) \tag{9}$$

where q_i are scalar coefficients. If such vector $\mathbf{x}_k$ exist (up to predefined precision) the hyperplane (7) can be rewritten as:

$$\boldsymbol{\Psi} = \sum_{\substack{i=1 \\ i \neq k}}^{m} \gamma_i \phi(\mathbf{x}_i) + \gamma_k \sum_{\substack{i=1 \\ i \neq k}}^{m} q_i \phi(\mathbf{x}_i) \tag{10}$$

Using the kernel equation (10) is written as:

$$f(\mathbf{x}) = \sum_{\substack{i=1 \\ i \neq k}}^{m} \gamma_i y_i K(\mathbf{x}, \mathbf{x}_i) + \gamma_k y_k \sum_{\substack{i=1 \\ i \neq k}}^{m} q_i K(\mathbf{x}, \mathbf{x}_i) + b \tag{11}$$

This may be finally rewritten as:

$$f(\mathbf{x}) = \sum_{\substack{i=1 \\ i \neq k}}^{m} \gamma_i' y_i K(\mathbf{x}, \mathbf{x}_i) + b \tag{12}$$

where

$$\gamma_i' = \gamma_i + \gamma_k q_i y_k / y_i \tag{13}$$

The form of the decision function is thus unchanged, but the coefficients are redefined to account for the removed component.

2.4 Reducing the Number of Support Vectors

The methodology of reduced set methods was proposed by Burges in [4]. When support vectors are removed the dimensionality of the transformed space is decreased and this is reflected in the change of the original decision hyperplane $\boldsymbol{\Psi}$ in the input space to $\boldsymbol{\Psi}'$ plane. The distance between the two hyperplanes

$$d = \min ||\boldsymbol{\Psi} - \boldsymbol{\Psi}'||^2 \tag{14}$$

should be as small as possible, and the approximation $\boldsymbol{\Psi}'$

$$\boldsymbol{\Psi}' = \sum_{i=1}^{m'} \beta_i \phi(z_i) \tag{15}$$

for the P-rules should satisfy $m' \ll m$, with scalar coefficients β_i. The inequality $m' \ll m$ should be considered very carefully because the number of SV cannot be too small [27].

Now there are two possible solution to the problem stated in this way. First, both the coefficients β_i and the position of vectors $\mathbf{x}_i$ in the input space may be optimized, and second, only the coefficients are optimized while support vectors are kept selected from the input vectors z_i. Both approaches has some advantages and disadvantages. Optimization of SV positions allows for better approximation and thus stronger reduction of the number of support vectors, but may lead to vectors that are difficult to interpret from the P-rule perspective. For example, in medical applications unrestricted optimization of support vector positions may create cases that are quite different from real patient's data, including intermediate values of binary features (such as sex). A compromise in which optimization of SVs is performed only in selected dimensions may be the best solution from both accuracy and comprehensibility point of view.

Optimizing support vectors $\mathbf{z}_i$ requires minimization of (14) over β and $\mathbf{z}$:

$$\min_{\beta,\boldsymbol{z}}(d) = \sum_{i,j=1}^{m} \gamma_i\gamma_j K(\boldsymbol{x}_i, \boldsymbol{x}_j) + \sum_{i,j=1}^{m'} \beta_i\beta_j K(\boldsymbol{z}_i, \boldsymbol{z}_j) \\ -2 \sum_{i=1}^{m} \sum_{j=1}^{m'} \beta_j \gamma_i K(\boldsymbol{x}_i, \boldsymbol{z}_j) \tag{16}$$

Directly minimization [4] requires evaluation of derivatives:

$$\frac{\partial}{\partial \beta_a} \left\| \boldsymbol{\Psi} - \sum_{i=1}^{m'} \beta_i \phi(\mathbf{z}_i) \right\|^2 = 2\phi(\mathbf{z}_a) \left(\boldsymbol{\Psi} - \sum_{i=1}^{m'} \beta_i \phi(\mathbf{z}_i) \right) \tag{17}$$

Setting this derivative to zero and replacing $\boldsymbol{\Psi}$ with (7) one obtains:

$$\sum_{j=1}^{m} \gamma_j \phi(\mathbf{x}_j) = \sum_{i=1}^{m'} \beta_i \phi(\mathbf{z}_i) \tag{18}$$

In the kernel matrix notation $K^{zx}\gamma = K^{zz}\beta$ where $\gamma = [\gamma_1, \gamma_2, \dots, \gamma_m]^T$, $\beta = [\beta_1, \beta_2, \dots, \beta_{m'}]^T$, and K^{zx} is matrix of the $m' \times m$ dimensions containing $K(\mathbf{z}_i, \mathbf{x}_j)$ values. The solution may be written in a number of ways, for example:

$$\beta = \left(K^{zz}\right)^{-1} K^{zx}\gamma \tag{19}$$

or using pseudoinverse matrices etc. Selection of the support vectors $\mathbf{z}_i$ from the initial pool of SVM-selected input vectors can be done using systematic search techniques, or using some stochastic selection procedures.

An interesting procedure for approximation $\mathbf{\Psi}$ have been proposed in [35], where the problem has been analyzed as clustering in the feature space. First notice that instead of direct distance (17) minimization the distance between $\mathbf{\Psi}$ and orthogonal projection of $\mathbf{\Psi}$ on the space generated by $Span(\phi(\mathbf{z}))$ may be used. Considering a single vector $\mathbf{z}$ and (16) the value of β is calculated from (19) as:

$$\beta = \sum_{i=1}^{m} \gamma_i K(\mathbf{x}_i, \mathbf{z}) \Big/ K(\mathbf{z}, \mathbf{z}) \tag{20}$$

and then $\mathbf{z}$ may be optimized minimizing:

$$\min_{\mathbf{z}} \left\| \frac{\mathbf{\Psi} \cdot \phi(\mathbf{z})}{\phi(z)\phi(\mathbf{z})} \phi(\mathbf{z}) - \mathbf{\Psi} \right\|^2 = \|\mathbf{\Psi}\|^2 - \frac{\left(\mathbf{\Psi} \cdot \phi(\mathbf{z})\right)^2}{\phi(\mathbf{z})\phi(\mathbf{z})} \tag{21}$$

This is equivalent to maximization of:

$$\max_{\mathbf{z}} \left(\frac{\left(\mathbf{\Psi} \cdot \phi(\mathbf{z})\right)^2}{\phi(\mathbf{z})\phi(\mathbf{z})} \right) \tag{22}$$

In case of similarity-based kernels $K(\mathbf{z}, \mathbf{z}) = 1$ and maximization in (22) can be simplified just to maximization of the numerator using fixed-point iterative methods. Calculating derivatives it is not hard to show that first approximation to $\mathbf{z}_1$ is calculated as [35]:

$$\mathbf{z}_1 = \frac{\sum_{i=1}^{m} \gamma_i K(||\mathbf{x}_i - \mathbf{z}||^2)\mathbf{x}_i}{\sum_{i=1}^{m} \gamma_i K(||\mathbf{x}_i - \mathbf{z}||^2)} \tag{23}$$

and iterations improve this estimation:

$$\mathbf{z}_{n+1} = \frac{\sum_{i=1}^{m} \gamma_i K(||\mathbf{x}_i - \mathbf{z}_n||^2)\mathbf{x}_i}{\sum_{i=1}^{m} \gamma_i K(||\mathbf{x}_i - \mathbf{z}_n||^2)} \tag{24}$$

Stability of this process is not guaranteed, and results strongly depend on the initialization of $\mathbf{z}$ and may require multiple restarts to find good solution.

This is one of many possible approaches. Another interesting method has been proposed by Kwok and Tsang [26], using Multidimensional Scaling (MDS) algorithm to represent images of the feature space vectors back in the input space.

2.5 Finding Optimal Number of Support Vectors

Analysis of numerical experiments performed by Downs et al. [7] shows that RLSV method is not sufficient for rule generation. Elimination of linear dependencies among SVs leads to a small reduction of their number, although quality of results is usually quite good. One exception is reduction of over 80% of the original number of SVs for the Heberman dataset [7], where quadratic kernel with SMO optimization found 87 SVs, while RLSV algorithm reduced it to just 10 vectors. Stronger reduction may be achieved relaxing numerical accuracy for linear dependency tests, but this will probably degrade also the quality of results. The effects of such reduction remains to be investigated.

Quality of this method depends on the type of kernel function, the C-value and the complexity of the decision border created by the SVM algorithm. Parameter C defining the size of SVM margins has important influence on the number of SVs. In soft margin SVM softer margins (lower C value) leads to a higher number of SVs that have more linear dependencies and thus higher reduction rate is obtained. Generally best results of RLSV algorithm are obtained for linear kernel, as in principle two support vectors are sufficient to define a decision hyperplane.

RS-SVM approach allows for significant reduction of the number of SVs, leading to more comprehensible models. To find optimal number of SVs any search method can be used with typical cost function driven by minimization of the distance between separating hyperplanes ((14)):

$$E_1(m') = \|\boldsymbol{\Psi} - \boldsymbol{\Psi}'\| =$$
$$\left(\sum_{i,j=1}^{m} \gamma_i\gamma_j K(\mathbf{x}_i, \mathbf{x}_j) + \sum_{i,j=1}^{m'} \beta_i\beta_j K(\mathbf{z}_i, \mathbf{z}_j) - 2\sum_{i=1}^{m}\sum_{j=1}^{m'} \beta_j\gamma_i K(\mathbf{x}_i, \mathbf{z}_j)\right)^2 \tag{25}$$

An additional term $\alpha m'/m$ defining model complexity as a fraction of reduced number of SVs (m') to the original number of SVs (m) multiplied by some constant α may be added to the difference of distances between hyperplanes. Because the distance $\|\boldsymbol{\Psi} - \boldsymbol{\Psi}'\|$ may take very high values α may be rescaled by $1/\|\boldsymbol{\Psi} - \boldsymbol{\Psi}'_1\|$, where $\boldsymbol{\Psi}'_1$ is $\boldsymbol{\Psi}'$ defined with just one SV.

An alternative function that measures changes in classification accuracy may be defined as:

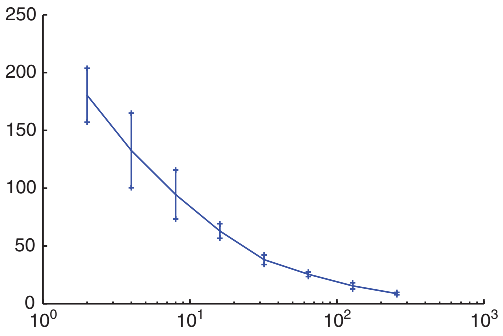
$$E_2(m') = \mathrm{acc(SVM)} - \mathrm{acc(RSSVM(m'))} \tag{26}$$

where acc() is classification accuracy measured using some loss function; in this case also the penalty for complexity may be added. Because $acc(SVM)$

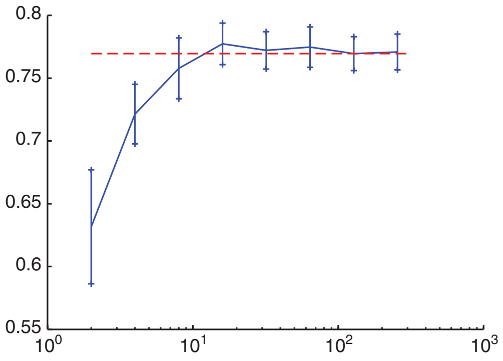
doesn't change during optimization the number of SV, we can simplify the function (26) omitting the first component, optimizing:

$$E_3(m') = \text{acc}(\text{RSSVM(m')}) \tag{27}$$

To compare the approach based on minimization of distance and accuracy few tests have been done using Gaussian SVM on two datasets, Pima Indians Diabetes, and Cleveland Heart Disease [28]. In the first step all datasets were normalized to the $[0, 1]$ range. The best C value for the SVM was selected using five-fold cross validation (CV) greedy search procedure in the $C = 2^1$ to $C = 2^8$ range, while $\sigma = 1$, and Alpha cutoff $= 10^{-2}$ were fixed. Finally the process of five-fold CV was used to test different cost functions using Fixed Point Iteration algorithm (Fig. 1 for the first, and Fig. 2 for the second

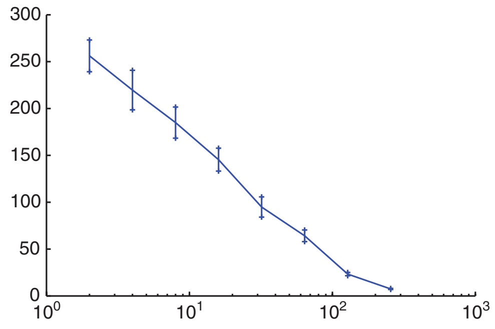


(a) Dependence of the cost function E_1 on the logarithm of the number of SVs

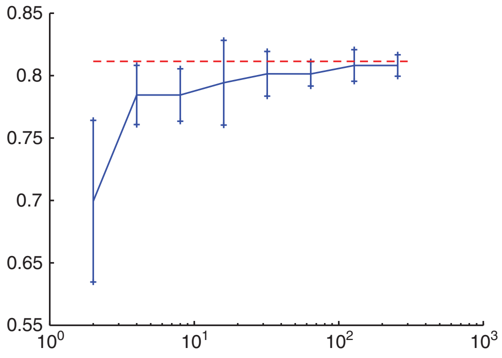


(b) Dependence of the mean accuracy (cost function E_3) on the logarithm of the number of SVs. *Dashed line* represents mean accuracy of the original SVM

Fig. 1. Comparison of the distance (25) and accuracy (27) based cost functions for Pima Indians diabetes data



(a) Dependence of the cost function E_1 on the logarithm of the number of SVs



(b) Dependence of the mean accuracy (cost function E_3) on the logarithm of the number of SVs. Dashed line represents mean accuracy of the original SVM

Fig. 2. Comparison of the distance (25) and accuracy (27) based cost functions for Cleveland Heart disease data

dataset). The distance between hyperplanes plotted in the top subfigure is decreasing in approximately linear way with the logarithm of the number of SVs. On the other hand the classification accuracy (27) grows rapidly reaching the accuracy of SVM with just a few SVs.

Although increasing the number of SVs leads to decision border that are equivalent to the one found by SVM algorithm without restrictions on the number of SVs results are not correlated with increasing accuracy of the models. Large differences between hyperplanes in the region far from data are not important, but the distance-based approach does not distinguish between different regions, trying to decrease the overall distance. This problem will be especially acute for Gaussian or other non-linear kernels that place SV far from decision borders in the feature space. For two overlapping distributions

SVM with Gaussian kernels will use support vectors that are all around both distributions, even though only those that are close to the support vectors from the opposite class are really useful. It should be possible to use the distance between closest support vectors from the opposite classes to rank candidates for removal in the SV selection process. This can simplify the search in the accuracy-based approach.

2.6 Problems with Interpretation

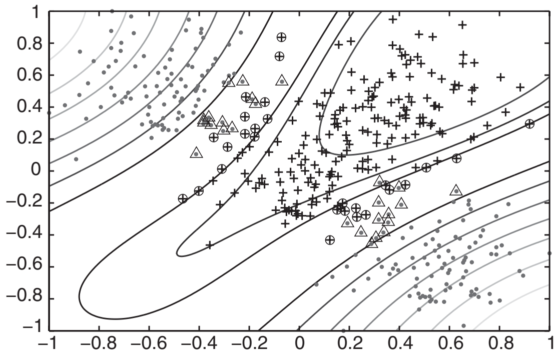
Even if a simple and transparent model that mimics SVM's decision borders could be created the question "what can be learned from it" still remains. Similar problems face most rule extraction approaches, including fuzzy and rough rule based systems, with the exception of simple crisp rule sets that sometimes have straightforward interpretation [14, 19]. Prototype-based rules demand not only a small number of prototypes but also a meaningful position of these prototypes among other input vectors.

None of the support vector reduction methods considered here gives prototypes which have simple interpretation, as they are never placed at the centers of clusters (as in the RBF networks). This problem is illustrated in Fig. 3a. Four prototypes selected by the Schölkopf algorithm are somewhere near the decision border and in the "flattened" image space are sufficient to define good border, but in the feature spaces they make little sense. More intuitive solution is obtained with the Burges algorithm where position of prototypes looks more "natural", however also here the knowledge which can be inferred from these positions is not clear.

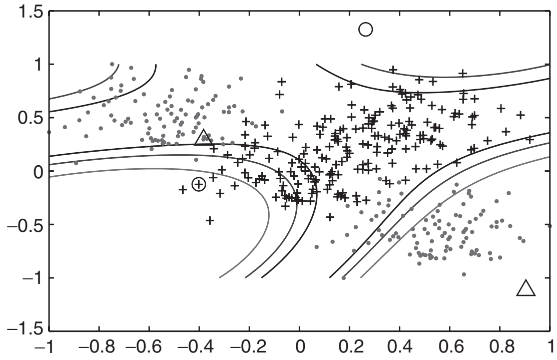
If the goal is to understand the data the problem of prototype selections should be solved in some other way. In the next section algorithms driven by prototype selection methods used in the k-nearest neighbor (kNN) classifiers are used to search for informative prototypes.

3 Searching for Informative Prototypes

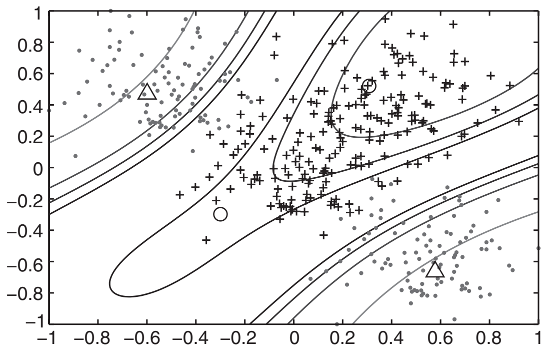
SVM decision borders should be approximated in such a way that uses informative prototypes to understand data structure. These prototypes do not have to be selected from support vectors, but may be placed in optimized positions. Possible solutions can be taken from kNN learning algorithms where many prototype selection methods that reduce the number of reference vectors exist. Good comparison of existing prototype selection algorithms can be found in papers by Jankowski and Grochowski [22, 23] and Wilson and Martinez [39]. The general algorithm proposed here starts from training SVM model, then selecting prototypes using one of the algorithms developed for kNN methods, and then assigning to each prototype weight value to reproduce the SVM decision border. The weights are calculated using (19). To facilitate better interpretation of results



(a) Contour plot of SVM classifier decision borders



(b) Contour plot of Schölkopf's RS-SVM with marked positions of prototypes



(c) Contour plot of Burges RS-SVM with marked positions of prototypes

Fig. 3. An example of decision borders generated by (**a**) SVM classifier, (**b**) with RS-SVM reduction according to Schölkopf and (**c**) Burges algorithm

weights can be normalized without any loss of generalization using softmax procedure:

$$\beta' = \frac{\beta}{\sum \beta} \tag{28}$$

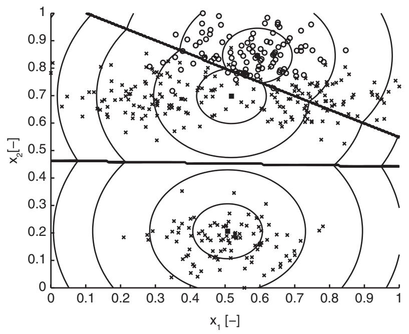
The weight value after normalization indicates how strong is the influence of each prototype on the final decision function. Generally the higher β'_i is, the more important associated i'th prototype is. The algorithm is schematically written below.

Algorithm 1

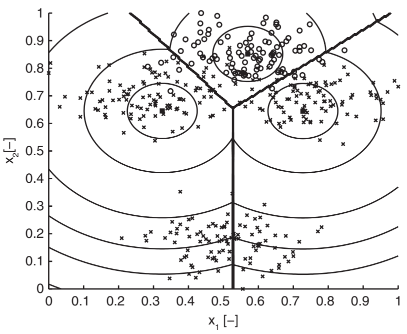
```
1: train SVM;
2: select prototypes with one of the kNN-based algorithms;
3: optimize prototype weights using formula (19);
4: normalize weights to [0,1] range.
```

3.1 Prototype Selection Using Context Dependent Clustering

One of the most popular methods for prototype selection in kNN and RBF classifiers is to use clustering of the training vectors. However, unsupervised clustering algorithms do not use any knowledge about class structure, leading to unnecessarily large number of prototypes. Such situation is presented in Fig. 4, where one of the prototypes is useless because it does not participate directly in construction of the decision border. This problem may be solved with semi-supervised clustering. A clustering approach which uses additional knowledge to reduce the number of prototypes was proposed by Blachnik



(a) Prototype selection using classical clustering method (FCM)



(b) Prototype selection using context clustering (CFCM)

Fig. 4. Comparison of prototype selection methods using two types of clustering methods, FCM and CFCM

et al. [3]. In this approach context dependent clustering was used to train the kNN prototypes. Context dependent clustering is a family of grouping algorithms which use external, user defined variable (for each input vector) describing the strengths of association between the input vector and external parameter. Context clustering was studied by Pedrycz [29, 31], Łęski [20, 36] and others, and has been applied with very good results in training of the RBF networks [1, 30].

3.2 The Conditional Fuzzy Clustering Algorithm

One of methods that belong to the context dependent clustering family of algorithms is Conditional Fuzzy C-Means (CFCM). It is based on minimizing cost function defined as:

$$J_m(\mathbf{U},\mathbf{P}) = \sum_{i=1}^{c}\sum_{k=1}^{m} (u_{ik})^{\delta} \left\|\mathbf{x}_k - \mathbf{p}_i\right\|_A^2 \tag{29}$$

where c is the number of clusters centered at $\mathbf{p}_i$, m is the number of vectors, $\delta > 1$ is a parameter describing fuzziness, and $\mathbf{U} = (u_{ik})$ is a $c \times m$ dimensional membership matrix with elements $u_{ik} \in [0,1]$ defining the degree of membership of the k-th vector in the i-th cluster. The matrix $\mathbf{U}$ has to fulfill three conditions:
1^o each vector x_k belongs to the i-th cluster to some degree:

$$\underset{1\le i\le c}{\forall}\ \underset{1\le k\le m}{\forall}\ u_{ik} \in [0,1] \tag{30}$$

2^o sum of the membership values of k-th vector x_k in all clusters is equal to f_k

$$\underset{1\le k\le m}{\forall}\ \sum_{i=1}^{c} u_{ik} = f_k \tag{31}$$

3^o no clusters are empty.

$$\underset{1\le i\le c}{\forall}\ 0 < \sum_{k=1}^{m} u_{ik} < m \tag{32}$$

Under these conditions cost function (29) reaches minimum for [29],

$$\underset{1\le i\le c}{\forall}\ \mathbf{p}_i = \sum_{k=1}^{m} (u_{ik})^{\delta}\mathbf{x}_k \left[\sum_{k=1}^{m} (u_{ik})^{\delta}\right]^{-1} \tag{33}$$

$$\underset{\substack{1\le i\le c\\ 1\le k\le m}}{\forall}\ u_{ik} = f_k \left[\sum_{j=1}^{c} \left(\frac{\|\mathbf{x}_k - \mathbf{p}_i\|}{\|\mathbf{x}_k - \mathbf{p}_j\|}\right)^{2/(\delta-1)}\right]^{-1} \tag{34}$$

3.3 Determining the Context

In classification problems the goal is to find a small number of prototypes that define classification border. In simple cases when linear solution is sufficient one prototype far from decision border implements approximately linear threshold P-rule. In more complex situations prototypes that are close to the decision border are needed, and they are also close to vectors from the opposite classes. This leads to a conclusion that grouping algorithms should be focused on clusters found close to the decision border and not on the whole space. For the context dependent clustering appropriate coefficients $f(k)$ taking this into account should be defined. Such a coefficient can be introduced in various ways, with one possible approach [3] based on the ratio of distances:

$$w_k = \sum_{j,C(\mathbf{x}_j)=C(\mathbf{x}_k)} \|\mathbf{x}_k - \mathbf{x}_j\|^2 \left[\sum_{l,C(\mathbf{x}_l)\neq C(\mathbf{x}_k)} \|\mathbf{x}_k - \mathbf{x}_l\|^2 \right]^{-1} \tag{35}$$

These coefficients are renormalized to fit the [0,1] range:

$$w_k \longleftarrow \left(w_k - \min_i w_i\right)\left(\max_i w_i - \min_i w_i\right)^{-1} \tag{36}$$

Normalized w_k coefficients reach values close to 0 for vectors inside large homogeneous clusters, and close to 1 if the vector x_k is near the vectors of the opposite classes and far from other vectors from the same class (for example if it is an outlier). These normalized weights determine the external variable which then is used to assign appropriate context or condition in the CFCM clustering process.

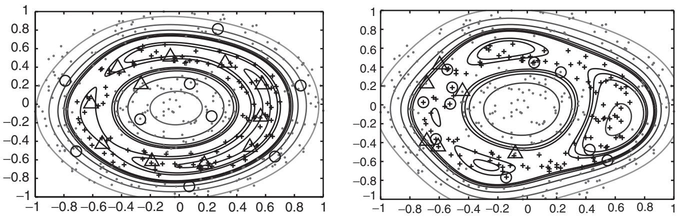
$$f_k = \exp\left(-\eta(w_k - \mu)^2\right) \tag{37}$$

with the best parameters in the range of $\mu = 0.6-0.8$ and $\eta = 1-3$, determined empirically for a wide range of datasets. The μ parameter controls where the prototypes will be placed; for small μ they are closer to the center of the cluster and for larger μ closer to the decision borders. The range in which they are sought is determined by the η parameter.

3.4 Numerical Illustration of the CFCM Approach

Conditional clustering proposed above does not use SVM to place prototypes directly, but the adjustment of weights is based on the SVM decision function. To verify this approach some simple numerical experiments were performed. Because in the CFCM method the number of prototypes for each class has to be determined independently the total number of desired SVs has been divided equally among the classes.

An artificial dataset example with a ring of data from one class between inner circle and outer data from another class was considered first, generated



(a) SVs selected using CFCM clustering (b) SVs selected using Schölkopf approach

Fig. 5. Comparison of the CFCM and Fixed Point algorithms on artificial data, showing support vector positions

using the Spider toolbox subroutines [38]. Results from the Fixed Point Iteration calculations (Schölkopf algorithm) and from the CFCM-based algorithm described are presented in Fig. 5. The number of SVs was set to 20 (in CFCM 10 SVs per class), and Gaussian SVM parameters $C = 10000$ and $\sigma = 1$ have been used for both methods.

Five SVs found by the Fixed Point algorithm could not be plotted because they lie outside of the figure. Mean accuracy of the original SVM was $92.0 \pm 1.7\%$, for the Fixed Point algorithm 87.5 ± 1.4 and 91.5 ± 2.3 for the CFCM algorithm.

This example shows that our CFCM algorithm finds prototypes that are informative and represent the shape of the decision border with high accuracy. More knowledge can be derived from CFCM prototypes if the number of prototypes per class is optimized. This can be done using for example the racing algorithm described in [3].

Three well known benchmark datasets from the UCI repository [28] were used to verify quality of the proposed solution on real data. The Pima Indians Diabetes, Cleveland Heart Disease, and Ionosphere datasets have been selected. All calculations were performed with the Spider toolbox [38] using the 5-fold crossvalidation, extended by our own subroutines for CFCM prototype selection algorithm. In all cases the number of SVs were fixed to 4 (in CFCM two per class), the C value for SVM was optimized using the greedy search approach, and the Gaussian kernel parameter was fixed $\sigma = 1$. Classification results are presented in Table 1.

These results show that also on the real-word problems CFCM clustering combined with SVM gives quite good results. For such a small number of SVs Schölkopf and Burges RS-SVM algorithms give rather poor results, while RS-SVM based on CFCM clustering on the Cleveland Heart dataset obtained even better results then the original SVM classifier.

Table 1. 5×CV classification error on the three datasets; the number of SVs from the Gaussian SVM given in the second column has been reduced in all cases to four

	#SV	SVM	Schölkopf RS-SVM	Burges RS-SVM	CFCM
Pima	305	23.4 ± 2.3	37.9 ± 7.6	38.3 ± 8.1	25.9 ± 1.3
Cleveland	99	20.9 ± 1.9	44.5 ± 8.8	31.6 ± 8.2	18.9 ± 1.8
Ionosphere	65	6.3 ± 1.3	18.8 ± 3.3	16.5 ± 2.7	13.1 ± 1.9

4 Conclusions

Although SVMs is a very powerful black box data classification tool it cannot be used directly for problems where decisions should be comprehensible. The hyperplane found in the feature space cannot be easily translated into the knowledge useful for data understanding in the original input space. Therefore various ways of expressing this knowledge should be studied. In this paper prototype-based rules are advocated as a natural extension of most of the rule based systems, well suited to the form of knowledge that may be derived from the SVM algorithm.

To represent knowledge contained in the SVM model in a comprehensible way as P-rules reduction of the number of SVs is necessary. This topic has been studied by many experts and a few approaches have been discussed in this chapter. Minimization of the distance between original SVM hyperplane and the one obtained after reduction of the number of SVs does not seem to be correlated with the accuracy of the system obtained in this way. More comprehensible results are obtained using cost functions that are based directly on the classification accuracy.

Another problem that is facing P-rules based on typical RS-SVM algorithms is the interpretation of obtained prototypes. A solution proposed here is to use algorithms developed for optimization of the classical kNN classifiers. As an example conditional clustering algorithm (CFCM) was adopted to learn prototypes (SV) from the original dataset, with SVM hyperplane used to fit appropriate weights to selected prototypes. Results of such a combination on the artificial and real data used in this paper appear to be quite good, although it should be tested on much wider range of data and actual knowledge in form of P-rules should be carefully analyzed. This approach should be combined with feature selection to simplify further the interpretation of the rules.

Visualization techniques offer an interesting alternative to the rule-based understanding of the SVM function, as it has been done for MLP [10] and RBF neural networks [9, 11].

References

1. M. Blachnik. Warunkowe metody rozmytego grupowania w zastosowaniu do uczenia radialnych sieci neuronowych. Master's thesis, Silesian University of Technology, Gliwice, Poland, 2002.
2. M. Blachnik and W. Duch. Prototype-based threshold rules. *Springer Lecture Notes in Computer Science*, 4234, 2006.
3. M. Blachnik, W. Duch, and T. Wieczorek. Selection of prototypes rules - context searching via clustering. *Lecture Notes in Artificial Intelligence*, 4029:573–582, 2006.
4. C. Burges. Simplified support vector decision rules. In *International Conference on Machine Learning*, pages 71–77, 1996.
5. Y. Chen and J.Z. Wang. Support vector learning for fuzzy rule-based classification systems. *IEEE Transactions on Fuzzy Systems*, 11(6):716–728, 2003.
6. R. Collobert and S Bengio. SVMTorch: Support vector machines for large-scale regression problems. *Journal of Machine Learning Research*, 1:143–160, 2001.
7. T. Downs, K. Gates, and A. Masters. Exact simplification of support vector solutions. *The Journal of Machine Learning Research*, 2:293–297, 2001.
8. W. Duch. Similarity based methods: a general framework for classification, approximation and association. *Control and Cybernetics*, 29:937–968, 2000.
9. W. Duch. Coloring black boxes: visualization of neural network decisions. In *Int. Joint Conf. on Neural Networks, Portland, Oregon*, volume I, pages 1735–1740. IEEE Press, 2003.
10. W. Duch. Visualization of hidden node activity in neural networks: I. visualization methods. In L. Rutkowski, J. Siekemann, R. Tadeusiewicz, and L. Zadeh, editors, *Lecture Notes in Artificial Intelligence*, volume 3070, pages 38–43. Physica Verlag, Springer, Berlin, Heidelberg, New York, 2004.
11. W. Duch. Visualization of hidden node activity in neural networks: Ii. application to rbf networks. In L. Rutkowski, J. Siekemann, R. Tadeusiewicz, and L. Zadeh, editors, *Lecture Notes in Artificial Intelligence*, volume 3070, pages 44–49. Physica Verlag, Springer, Berlin, Heidelberg, New York, 2004.
12. W. Duch, R. Adamczak, and G.H.F. Diercksen. Distance-based multilayer perceptrons. In M. Mohammadian, editor, *International Conference on Computational Intelligence for Modelling Control and Automation*, pages 75–80, Amsterdam, The Netherlands, 1999. IOS Press.
13. W. Duch, R. Adamczak, and G.H.F. Diercksen. Classification, association and pattern completion using neural similarity based methods. *Applied Mathematics and Computer Science*, 10:101–120, 2000.
14. W. Duch, R. Adamczak, and K. Grąbczewski. A new methodology of extraction, optimization and application of crisp and fuzzy logical rules. *IEEE Transactions on Neural Networks*, 12:277–306, 2001.
15. W. Duch and M. Blachnik. Fuzzy rule-based systems derived from similarity to prototypes. In N.R. Pal, N. Kasabov, R.K. Mudi, S. Pal, and S.K. Parui, editors, *Lecture Notes in Computer Science*, volume 3316, pages 912–917. Physica Verlag, Springer, New York, 2004.
16. W. Duch and G.H.F. Diercksen. Feature space mapping as a universal adaptive system. *Computer Physics Communications*, 87:341–371, 1995.
17. W. Duch and K. Grudziński. Prototype based rules - new way to understand the data. In *IEEE International Joint Conference on Neural Networks*, pages 1858–1863, Washington D.C, 2001. IEEE Press.

18. W. Duch and K. Grudziński. Meta-learning via search combined with parameter optimization. In L. Rutkowski and J. Kacprzyk, editors, *Advances in Soft Computing*, pages 13–22. Physica Verlag, Springer, New York, 2002.
19. W. Duch, R. Setiono, and J. Zurada. Computational intelligence methods for understanding of data. *Proceedings of the IEEE*, 92(5):771–805, 2004.
20. J. Łęski. Ordered weighted generalized conditional possibilistic clustering. In J. Chojcan and J. Łęskiki, editors, *Zbiory rozmyte i ich zastosowania*, pages 469–479. Wydawnictwa Politechniki lskiej, Gliwice, 2001.
21. K. Grąbczewski and W. Duch. Heterogeneous forests of decision trees. *Springer Lecture Notes in Computer Science*, 2415:504–509, 2002.
22. M. Grochowski and N. Jankowski. Comparison of instance selection algorithms. ii. results and comments. *Lecture Notes in Computer Science*, 3070:580–585, 2004.
23. N. Jankowski and M. Grochowski. Comparison of instance selection algorithms. i. algorithms survey. *Lecture Notes in Computer Science*, 3070:598–603, 2004.
24. T. Joachims. *Learning to Classify Text Using Support Vector Machines.* Kluwer Academic Publisher, 2002.
25. L.I. Kuncheva and J.C. Bezdek. An integrated framework for generalized nearest prototype classifier design. *International Journal of Uncertainty*, 6(5):437–457, 1998.
26. J.T. Kwok and I.W. Tsang. The pre-image problem in kernel methods. *IEEE Transactions on Neural Networks*, 15:408–415, 2003.
27. K. Lin and C. Lin. A study on reduced support vector machines. *IEEE Transactions on Neural Networks*, 14(6):1449–1459, 2003.
28. C.J. Merz and P.M. Murphy. UCI repository of machine learning databases, 1998–2004. http://www.ics.uci.edu/∼mlearn/MLRepository.html.
29. W. Pedrycz. Conditional fuzzy c-means. *Pattern Recognition Letters*, 17:625–632, 1996.
30. W. Pedrycz. Conditional fuzzy clustering in the design of radial basis function neural networks. *IEEE Transactions on Neural Networks*, 9(4), 1998.
31. W. Pedrycz. Fuzzy set technology in knowledge discover. *Fuzzy Sets and Systems*, 98(3):279–290, 1998.
32. E. Pękalska and R.P.W. Duin. *The dissimilarity representation for pattern recognition: foundations and applications.* New Jersey; London: World Scientific, 2005.
33. J. Platt. Using sparseness and analytic qp to speed training of support vector machines. *Advances in Neural Information Processing Systems*, 11, 1999.
34. I. Roth and V. Bruce. *Perception and Representation.* Open University Press, 1995. 2nd ed.
35. B. Schölkopf, P. Knirsch, A. Smola, and C. Burges. Fast approximation of support vector kernel expansions. *Informatik Aktuell, Mustererkennung*, 1998.
36. J. Łęski. A new generalized weighted conditional fuzzy clustering. *BUSEFAL*, 81:8–16, 2000.
37. T. Takagi and M. Sugeno. Fuzzy identification of systems and its applications to model ing and control. *IEEE Transactions on Systems, Man, Cybernetics*, 15:116–132, 1985.
38. J. Weston, A. Elisseeff, G. BakIr, and F. Sinz. The spider. http://www.kyb.tuebingen.mpg.de/bs/people/spider/.
39. D.R. Wilson and T.R. Martinez. Reduction techniques for instance-based learning algorithms. *Machine Learning*, 38:257–268, 2000.

Part III

Applications

Prediction of First-Day Returns of Initial Public Offering in the US Stock Market Using Rule Extraction from Support Vector Machines

Rolf Mitsdorffer[1] and Joachim Diederich[2]

[1] Queensland Education, Brisbane, Australia
[2] American University of Sharjah, UAE and University of Queensland, Australia

Summary. Artificial neural networks (ANNs) and support vector machines have successfully improved the quality of predicting share movements in relation to statistically based counterparts. However, it has not been feasible to gain insight into the reasons why a certain prediction is made. Due to this limitation, the use of machine learning techniques in the capital market has met a critical hurdle. This chapter outlines a method based on pedagogical learning for extracting rules from support vector machines. To the best of our knowledge, the experiments reported here are the first attempt to utilize learning based rule extraction from support vector machines for financial data mining.

The experiments use predictions from support vector machines for extracting rules associated with the first-day returns of "initial public offerings" (IPOs) in the US stock market. A novel feature of the experiments is the simultaneous application of *fundamental* and *technical analysis* in the context of predicting the success of IPOs. Cross-industry IPOs covering the period from 1974 to 1984 and software and services IPOs launched between 1996 and 2000 are utilized.

1 Motivation

Predictions of share prices in the capital market are said to be inconsistent with the theory underlying the "capital asset pricing method" (CAPM). CAPM is based on the random walk hypothesis which assumes linearity of the data. Unfortunately, the statistically based linearity assumption is frequently invalid. Marginal improvements in the prediction of share prices have been obtained based on non-linear models. Forecasting limitations are also imposed by the difficulty of understanding the time dependent dynamics of the share market. Models based on symbolic manipulation never quite capture the dynamic essence of the market. This problem results in a "knowledge acquisition bottleneck" and limits human understanding of the multi-dimensional problem.

The experiments reported here are aimed at addressing the current forecasting limitations at the birth of the capital market: initial public offerings.

2 Introduction

2.1 Financial Data Mining

Machine learning techniques are increasingly being adopted by capital market analysts and have been used in numerous studies. It is likely that only a minority of projects have been published due to the commercial nature of the applications.

A number of authors have used ANNs to model stock returns. Refenes and Zapranis (1995) describe an experiment using a set of unspecified factors extracted from the balance sheets of companies in the universe of UK stocks. Mitsdorffer et al. (2001, 2002) employed ANNs and other machine learning techniques for the predictions of first-day returns of initial public offerings and found significant market inefficiencies.

Comparing results from classical statistical techniques with simple neural learning procedures, Mitsdorffer et al. (2001, 2002) concluded that predictions derived from ANNs outperform current best practice and have a significantly better generalisation capability.

Similar to artificial neural networks, the knowledge embedded in support vector machines is opaque in that it cannot easily be made comprehensible to a human user. While rule extraction from ANNs is now established (Andrews et al. 1995), there have not been any attempts to extract rules from support vector machines prior to 2002. Mitsdorffer et al. (2001, 2002) report first experiments including learning based rule extraction from support vector machines.

2.2 IPOs as a Case Study

This research explores the application of machine learning techniques in the field of initial public offerings, a subset of the capital market. In this section, the purpose of IPOs and processes leading up to them are outlined.

A private company can be converted to a public corporation by raising funds for expansion, product development or the restructuring of debt. IPOs reduce the dependence of companies on bank credit, a notoriously unstable way to allocate capital. IPOs are often used by venture capitalists as an "exit route".

The most important and time consuming task facing the IPO preparation team is the development of the prospectus, a business document that serves as a brochure for the company. Since the US Securities and Exchange Commission (SEC) imposes a "quiet period" on companies once they file for an

IPO until 25 days after the stock starts trading, the prospectus is the primary source of information for the investor.

At the heart of a prospectus is the company's financial position and past financial performance. Financial reports are essential for informed decision making. Relevant financial information may be found in several sections of a prospectus – including the balance sheet, the profit and loss statement, cash flow analysis and the accountant's report and notes.

2.3 The Valuation of IPOS

According to Ritter (1991), IPOs are not different from other stocks, where discounted cash flow (DCF) analysis and the comparable firm analysis are used. Numerous studies have investigated the "short run under-pricing" of IPOs and the "hot issue market" phenomenon (Ibbotson & Jaffe 1975; Ritter (1984)). Under pricing of IPOs is an internationally widespread phenomenon. Ritter (1991, p. 3) observed that "investors are periodically overoptimistic about the earning potential of young growth companies and firms take advantage of these windows of opportunity".

There is a presumption that many young firms issuing new shares have growth potential, which is difficult to forecast using one-year-ahead earnings projections. Moonchul and Ritter (1999) tested this idea by using a sample of young and older firms. Consistent with the assumption that younger firms are more difficult to value; the authors determined that the valuation errors were noticeably smaller for older firms.

3 Overview of the Chapter

The rule extraction experiments described in this chapter focus on two different datasets: "cross-industry" and "single-industry" IPOs. Following trials to investigate the overall ability of SVMs to predict first-day IPO returns, support vector machine predictions are obtained from test datasets. The predictions associated with these test datasets are then used to extract rules representing what the SVM has learned using pedagogical rule extraction techniques. Finally, statistical tests are utilized to establish that the extracted rules represent what the SVM has learned.

4 Methodology

The project includes the collection of data from company reports and stock market indices. Following a pre-processing phase to make the data amenable to support vector and decision tree learning, the data serves as training input to machine learning techniques.

For each of the cross-industry and single-industry datasets:

1. SVM training and prediction using the *leave-one-out cross-validation method* is conducted by use of SVMlight (Joachims, 1999). Different parameters are used to establish the effect on prediction and generalisation quality.
2. SVM predictions are transcribed into the test dataset. This dataset represents what the SVM has learned.
3. Rules are extracted from the transcribed dataset using See5, C4.5, Ripper and Rulex.
4. An overall analysis of the prediction, generalisation and rule extraction procedure is performed.

The performance of individual machine learning techniques is evaluated in terms of precision, and recall as well as the f-value:

- Precision = true positive/(true positive + false positive)
- Recall = true positive/(true positive + false negative)
- F-value = (2*precision*recall)/(precision + recall)

The experiments result in several competing models of the stock market dynamics governing returns of IPOs on the first trading day based on the extracted rules.

4.1 Statistical Tests

In order to establish that rules represent what the SVM has learned, the McNemar Test (Gardner & Altman 1989) is used to test whether combinations between two dichotomous variables are equally likely. The exact p-value is determined using the binomial distribution as described in Gardner and Altman (1989) and implemented in the statistical analysis tool *Analyse-it*. Based on the p-value (p-value < critical value) the null hypothesis of inequality is accepted. The rejection of the null hypothesis leads to the conclusion that rules from a given technique significantly represent what the SVM has learned at a level of confidence of 95%.

4.2 Data

Cross-Industry IPOs

An extensive search was conducted to locate data in the public domain suitable for this research. The search established IPO data sources in the public domain of R. J. Ritter, University of Florida that are freely available for academic research.[1] Ritter (1991) has used this data to investigate the long-term performance of IPOs.

[1] `http://bear.cba.ufl.edu/ritter/ipodata.htm`.

The dataset includes 2,609 firms with common stock initial public offerings in the period between 1974 and 1984. Companies included in the dataset have used S1 or S18 registration statements. The primary source of information is the direct inspection of the prospectuses.

For each IPO, the following aspects are relevant:

- First-day trading data, including the date the company went public, open and closing prices
- Fundamental data to enable the valuation of companies, such as assets and liabilities as well as shareholder equity and dilution
- Past performance data, including sales, the cost of sales, expenditure for R&D, etc.
- Proxies for market sentiment

In order to reduce extreme outliers, the following selection criteria are used to exclude IPOs with the following attribute values:

- Minimum number of shares $< 2,000$
- Market capitalisation $< \$1\,\text{m} > \$1,000\,\text{m}$
- Offer price $> \$30$
- First-day variation $< -80\% > 200\%$
- Offer fraction < 0.05.

The cross-industry attributes are shown in Table 1.

Market sentiment data is represented by wins and losses of the NASDAQ computer index in the 100 trading days immediately preceding the IPO. The period of 100 trading days is condensed into five separate 20-day periods.

Input values are scaled in the range from -1 to $+1$. Outliers are replaced by maximum and minimum values established according to the table of bin attribute values.

Next, the target values are established. The analysis of first-day gains identified about 24% of the 1,841 IPOs with gains of over 18.1%. Using binary classification, IPOs in the above 18.1% bracket are classified as positive while the remainders are classified as negative.

Computer Software and Services IPOs

This section describes the data used for predicting first-day returns of single industry IPOs. The rationale for selecting a single industry is to ascertain if financial ratios representing underlying company fundamentals within a single industry are more comparable than cross-industries data and thus provide more plausible explanations of first-day IPO returns.

IPO data related to the computer software and services categories were retrieved from the Hoover service. Companies in this sector are involved in the design and marketing of all types of software and the provision of computer services, such as mainframe and system integration.

Table 1. Attribute summary of cross-industry IPOs

Attribute name	Attribute description
BOOK.CAPZTN	Book value/capitalization ratio
BOOK.VALUE	Book value (absolute)
CAPITALIZTN	Capitalization (absolute)
EXPENSES	Expenses (absolute)
GAIN.LOSS.1	First-day IPO gain/loss 1–20 days prior to first trading day
GAIN.LOSS.2	First-day IPO gain/loss 21–40 days prior to first trading day
GAIN.LOSS.3	First-day IPO gain/loss 41–60 days prior to first trading day
GAIN.LOSS.4	First-day IPO gain/loss 61–80 days prior to first trading day
GAIN.LOSS.5	First-day IPO gain/loss 81–100 days prior to first trading day
NO.IPOS.1	Number of IPOs released 1–20 days prior to the first trading day
NO.IPOS.2	Number of IPOs released 21–40 days prior to the first trading day
NO.IPOS.3	Number of IPOs released 41–60 days prior to the first trading day
NO.IPOS.4	Number of IPOs released 61–80 days prior to the first trading day
NO.IPOS.5	Number of IPOs released 81–100 days prior to the first trading day
OFFER.FRACT	Offer fraction
OFFER.PR	Offer price (absolute)
REV.CAP	Revenue vs. capitalization ratio
REVENUE	Revenue absolute
RISKS	Number of risk factors
SEL.DAYS	Number of days selling days
SP.DIFF.1	Absolute difference of the S&P Index 1–20 days prior to the first trading day
SP.DIFF.2	Absolute difference of the S&P Index 21–40 days prior to the first trading day
SP.DIFF.3	Absolute difference of the S&P Index 41–60 days prior to the first trading day
SP.DIFF.4	Absolute difference of the S&P Index 61–80 days prior to the first trading day
SP.DIFF.5	Absolute difference of the S&P Index 81–100 days prior to the first trading day
UW.DISCOUNT.CAP	Underwriter discount in relation to the market capitalization
YR.FOUND	Year the company was founded
BOOK.CAPZTN	Book value/capitalization ratio
BOOK.VALUE	Book value (absolute)
CAPITALIZTN	Capitalization (absolute)
EXPENSES	Expenses (absolute)

IPOs satisfying certain criteria were selected. These criteria include a minimum of \$10 million in sales and 80 employees or more. At the time of data collection, nearly all IPOs between 1996 and 1999 were considered.

The aim of the next stage of the data collection is the acquisition of balance sheet and income data from company IPO prospectuses launched with the SEC.

The following aspects are included in the model:

- IPO specific data, such as the date of listing, the date the company went public, offer price, offer fraction, post offering shares
- First-day closing prices
- Fundamental data to enable the valuation of companies, such as assets and liabilities as well as shareholder equity and dilution
- Past performance data, including sales, the cost of sales, expenditure for R&D, etc.
- Daily NASDAQ computer index values for 100 days preceding each IPO

The single-industry attributes considered for the analysis are shown in Table 2.

Each IPO is represented by a vector with 27 input features and one output or predictor attribute. Input features include attributes calculated from balance sheets and income data and those constructed from the NASDAQ computer index. Attributes are scaled which requires knowledge of the statistical properties of attributes to eliminate the effects of outliers.

The decision was made to consider the influence of the NASDAQ computer index on 100 trading days preceding the IPO. Considering the small dataset of 182 IPOs, the number of attributes representing the index was reduced by dividing the 100 index values into 10 time periods and forming the absolute difference of the index for each 10-day period. Generating attributes based on the absolute difference also has the effect of removing the time dependency of index values which in turn enables the formation of time independent rules.

The output attribute represents first-day gains or losses of an IPO. Since the aim of the research is the prediction of first-day gains and the extraction of rules, a binary model was built where IPOs exceeding a certain percentage of first-day gains are classified as positive and others as negative.

4.3 Machine Learning Techniques Used in This Study

Support Vector Machines

Support vector machines are an alternative to neural networks as tools for solving pattern recognition problems. SVMs have a major advantage over neural networks in that they formulate the learning problem as a quadratic optimization problem whose error surface is free of local minima and has a unique global optimum.

Table 2. Attribute summary of single-industry IPOs

Attribute name	Attribute description
ActualOffer	Actual offer price (absolute)
CashLiab	Cash vs. liabilities (%)
CurAssLiab	Current assets vs. liabilities (%)
EquityShare	Equity per share (%)
GrMarginShare	Gross margin per share (%)
GrossMargin	Gross margin (%)
IncGrowth	Income growth (%)
MarketCap	Market capitalization (absolute)
NASDAQ.Period.1	NASDAQ computer index 10 days prior to first trading day
NASDAQ.Period.2	NASDAQ computer index 20 days prior to first trading day
NASDAQ.Period.3	NASDAQ computer index 30 days prior to first trading day
NASDAQ.Period.4	NASDAQ computer index 40 days prior to first trading day
NASDAQ.Period.5	NASDAQ computer index 50 days prior to first trading day
NASDAQ.Period.6	NASDAQ computer index 60 days prior to first trading day
NASDAQ.Period.7	NASDAQ computer index 70 days prior to first trading day
NASDAQ.Period.8	NASDAQ computer index 80 days prior to first trading day
NASDAQ.Period.9	NASDAQ computer index 90 days prior to first trading day
NASDAQ.Period.10	NASDAQ computer index 100 days prior to first trading day
NetIncShare	Net income per share (%)
OfferOutst	Offer vs. outstanding shares (%)
PropActOffer	Proposed vs. actual offer price (%)
RDRev	Research & development vs. revenue (%)
RegDays	Registration days (absolute)
Revenue	Revenue (absolute)
RevGrowth	Revenue growth (%)
RevShare	Revenue per share (%)
SGRev	Sales and general expenses vs. revenue (%)

SVMs are based on some simple ideas and provide a clear intuition of what learning from examples is all about. More importantly, they also show high performance in practical applications. SVMs correspond to a linear method in a very high dimensional feature space that is non-linearly related to the input space. Even though SVMs implement a linear algorithm in a high dimensional feature space, in practice they do not involve any computations in that high dimensional space. By use of kernels, all necessary computations are performed directly in input space. Data vectors nearest to the separating hyperplane in the transformed space are called support vectors. Classification as well as regression can be learned by SVMs.

Joachims (1998) reported that SVMs are well suited to learn in high dimensional spaces (>10,000 inputs). They achieve substantial improvements over currently best performing methods, reducing the need for feature selection.

Rule Extraction from Neural Networks (Rapid Backpropagation)

Rule extraction from neural networks is used for benchmark purposes in this context. The extraction of symbolic knowledge from ANNs and the direct encoding of partial knowledge into ANNs before training are important issues. They allow the exchange of information between symbolic and neural network knowledge representation. ANNs store knowledge in a completely numerical form, which is not open to explanation, a situation similar to SVMs.

Rule extraction from local function networks employs decompositional algorithms that directly decompile weights to generate rules. The underlying network is formed by "Rapid Back Propagation" (RBP), a three layer architecture similar to radial base function networks. The network consists of an input layer, a hidden layer of locally responsive basis function nodes, and an output node. The network is suitable for binary classification tasks as well as function approximation.

Rulex, a tool used in this project and described by Andrews and Geva (1996), is a program that converts the numeric weights of RBP networks into symbolic IF THEN rules that explain the decisions made by the network.

Other Machine Learning Techniques

Classification techniques such as decision trees play a major role in machine learning and knowledge based systems. These learning methods have been successfully applied to a large range of tasks, from learning medical diagnostics to credit risks assessment and are used in this research.

See5/C5 is a system commercialized by Rulequest Research (1997) for analysing data and generating classifiers in the form of decision trees and/or rule sets. C4.5 is the program used in our experiments. Quinlan's work (1986,1993,2001) on C4.5 is widely acknowledged as a major contribution to the development of classifier systems. Examples include a mixture of nominal and numeric properties that are analysed to allow the discrimination of classes. The patterns are expressed in the form of a decision tree or a set of IF THEN rules. The rules can be used to classify new cases.

The Ripper rule learner is a system for inducing classification rules from a set of pre-classified examples and has been used in this project for benchmark purposes. Ripper (Repeated Incremental Pruning to Produce Error Reduction) is an efficient, noise tolerant propositional rule learning algorithm based on the separate and conquer strategy. The basic strategy used by Ripper is to find an initial model and then to iteratively improve that model using an optimisation procedure described in Cohen (1995).

5 Results

For each of the datasets, cross-industry and single industry IPOs, the following results are reported:

- SVM training and prediction results using the leave-one-out method built-in to SVMlight (Joachims, 1999).
- SVM prediction results using explicit test sets. SVM prediction results are then transcribed into the test dataset. This dataset in essence represents what the SVM has learned.
- Rules extracted from the transcribed dataset using See5, C4.5, Ripper and Rulex.

5.1 Results of Rule Extraction from SVM for Cross-Industry IPOs

Leave-one-out Cross-Validation Results

At first glance, the learning and generalisation ability of support vector machines is sufficient as indicated by an error rate of 22.38% (Table 3).

SVM Prediction Using a Test Set

Results from a randomly created test set are shown below.

As is evident from Table 4, the quality of the SVM prediction is not satisfactory with a precision of 0.33, a recall 0.35 and an f-value 0.34. The problem is obviously a confusion of the positive class. This may be due to lack of data.

Table 3. Leave-one-out training results of cross-industry IPOs (rbf kernel)

Trade-off between training error and margin (c)	Cost factor (j)	Parameter in gamma rbf kernel (g)	Test error %
100	0.12	0.5	24.12
50	0.30	0.2	22.87
Default	0.1	0.1	23.85
Default	0.90	0.1	**22.38**

Table 4. Confusion matrix for SVM predictions

(a)	(b)	<-classified as
38	76	(a): class positive
72	275	(b): class negative

Rules Extracted by Use of See5, C4.5, Ripper and Rulex

This part of the experiment is based on transcribing the SVM prediction results into the training sets for rule learners. The results of training the four rule learners with the SVM test predictions as target output are shown below.

In order to establish if See5 rules represented what the SVM has learned the McNemar test (Gardner & Altman 1989) was used to investigate whether combinations between two dichotomous variables shown in the confusion matrix in Table 5 are equally likely.

The exact p-value was computed using the binomial distribution as described by Gardner and Altman (1989) and implemented in the statistical analysis tool *Analyse-it*. Based on the p-value (p-value < critical value) the null hypothesis of inequality is accepted. The acceptance of the null hypothesis leads to the conclusion that See5 rules fail to represent what the SVM has learned at a level of confidence of 95%. Consequently the resulting rules are not discussed here.

The results of extracting rules by use of C4.5 are shown in Table 6.

Similar to See5, C4.5 did not represent what the SVM has learned at a level of confidence of 95%.

Ripper extracted *one* rule from the dataset representing what the SVM has learned (Table 7).

Table 5. See5 confusion matrix for cross-industry IPOs

Evaluation on training data (460 cases):		
(a)	(b)	<-classified as
44	30	(a): class positive
3	383	(b): class negative

Table 6. Decision tree (C4.5) results

Classification	C4.5	
SVM	Positive	Negative
Positive	48	26
Negative	2	384

Table 7. Ripper rules of what the SVM has learned for cross-industry IPOs

```
Final hypothesis is:

positive :-
SP_DIFF_1>= 1, SP_DIFF_2>= −0.422777, YR_FOUND>= 0.317073 (40/30)
Default negative (328/62)
Train error rate: 20.00% 1.87% (460 data points) ≪
Hypothesis size: 1 rule, 4 conditions
```

Table 8. RBP/Rulex results of what the SVM has learned for cross-industry IPOs

Classification	Rulex	
	Positive	Negative
Positive	23	51
Negative	12	374

The precision for Ripper is 0.57, recall 0.38 and the f-value 0.46.

The null hypothesis of inequality (McNemar test) was rejected (p-value > critical value) and the alternative hypothesis of equality of what the SVM and Ripper have learned was accepted at a level of confidence of 95%. The success of this test is the extraction of a minimal rule set representing SVM learning results.

The RBP/Rulex results are shown in Table 8.

The McNeamar Test leads to the conclusion that Rulex failed to significantly represent what the SVM has learned at a level of confidence of 95% and the resulting rules are not discussed here.

5.2 Rule Extraction from SVM Results for Single-Industry IPOs

SVM Training and Prediction Using the Leave-one-out Method

Support vector machines are used to explore to what extent the upper 25% of first-day returns of "Software and Services IPOs" can be predicted and what are the rules governing these predictions are.

Results from SVM leave-one-out predictions (Table 9) including an error rate as low as 18.13% are an indication that market inefficiencies exist.

SVM Training and Prediction Using Test Sets

Similar to the earlier approach, a learning-based method for rule extraction from support vector machines is used. Software and Services IPOs are randomly split into 122 training and 60 test cases. Repeated random selection is performed until the test set contains about 25% positive cases (22), the same proportion as in the total dataset.

As is evident from Table 10, the quality of the SVM predictions is insufficient, with a precision of 0.3, recall of 0.5 and an f-value of 0.38.

To determine why SVM learning has failed, SVM prediction results are transcribed into the training sets for rule learners to establish what the SVM learned or failed to learn.

Rules Extracted by Use of See5, C4.5, Ripper and Rulex

The results of training the four rule learners with the SVM test predictions as target output are shown below.

Table 9. Leave-one-out results for single-industry IPOs (Linear SVMs)

C value	Test error %	Test recall %	Precision %
Default	19.23	57.14	74.42
0.1	19.23	57.14	74.42
0.2	29.12	5.36	100
1	18.68	60.71	73.91
2	18.13	62.50	74.47
4	21.98	54.17	66.67

Table 10. SVM prediction using a test set for single-industry IPOs

Classed as positive	Classed as negative	
7	16	Positive
7	32	Negative

The rejection of the null hypothesis in the McNemar Test leads to the conclusion that the See5 rules significantly represented what the SVM has learned at a level of confidence of 97.5%. Rule precision was established as 0.86, recall as 1 and the f-value as 0.92.

The rules extracted from C4.5 are shown in Table 12. Evaluation of the rules yields the following results: Since the confusion matrix in Table 13 is identical to the See5 learning results (Table 11) the conclusions are identical. The results of what Ripper has learned are shown in Table 14: Ripper established just one rule with a precision of 0.8, recall 0.57 and f-value 0.67. The McNemar Test established that Ripper significantly represents what the SVM has learned. The results of using the local functions network RBP and the rule extraction technique Rulex are shown in Table 15. The confusion matrix based on RBP and Rulex is shown in Table 16. RBP/Rulex precision is 0.93, recall 0.93 and f-value 0.93. This concludes the rule extraction from SVM experiments, leading to the interpretation of results. The McNemar Test established that RBP/Rulex significantly represents what the SVM has learned.

6 Discussion of Results

The knowledge stored in support vector machines is opaque and cannot easily be extracted. The aim of this research is the extraction of rules from support vector machines in the context of initial public offerings in the US stock market as well as the evaluation of the quality of the rules.

The results from these experiments show how pedagogical techniques using cross-industry and single-industry IPO datasets successfully extract rules from support vector machines.

Table 11. See5 rules of what the SVM has learned for single-industry IPOs

```
Extracted rules:
Rule 1: (cover 8)
        CurAssLiab > 0.6863084
        RevGrowth > −0.4512843
        − > class positive (0.900)
Rule 2: (cover 4)
        CurAssLiab <= −0.1916766
        MarketCap > 0.3253333
        − > class positive (0.833)
Rule 3: (cover 38)
        CurAssLiab <= 0.6863084
        MarketCap <= 0.3253333
        − > class negative (0.950)
Rule 4: (cover 16)
        RevGrowth <= −0.4512843
        − > class negative (0.944)
Rule 5: (cover 25)
        CurAssLiab > −0.1916766
        CurAssLiab <= 0.6863084
        − > class negative (0.926)
Default class: negative
          (a)      (b)      <-classified as
          12        2       (a): class positive
           0       48       (b): class negative
```

Table 12. C4.5 rules

```
Rule 1:
        CurAssLiab > 0.686308
        RevGrowth > −0.407389
        − > class positive (84.1%)
Rule 2:
        CurAssLiab <= −0.191677
        OfferOutst <= −0.761158
        − > class positive (70.7%)
Rule 3:
        CurAssLiab <= 0.686308
        OfferOutst > −0.761158
        − > class negative (93.2%)
Rule 4:
        RevGrowth <= −0.407389
        − > class negative (92.6%)
Rule 5:
        CurAssLiab > −0.191677
        CurAssLiab <= 0.686308
        − > class negative (89.8%)
Default class: negative
```

Table 13. C4.5 rule evaluation

Rule	Size	Error	Used	Wrong	Advantage	
1	2	15.9%	8	0 (0.0%)	8 (8|0)	positive
2	2	29.3%	4	0 (0.0%)	4 (4|0)	positive
3	2	6.8%	38	1 (2.6%)	0 (0|0)	negative
4	1	7.4%	6	0 (0.0%)	0 (0|0)	negative
5	2	10.2%	6	1 (16.7%)	0 (0|0)	negative
		(a)	(b)	<-classified as		
		12	2	(a): class positive		
		0	48	(b): class negative		

Table 14. Ripper results

Final hypothesis is:
Positive: – CurAssLiab >= 1 (8/2)
Default negative (46/6)
Train error rate: 12.90% 4.29% (62 data points) ≪
Hypothesis size: 1 rules, 2 conditions

The conclusions drawn from the cross-industry experiments are:

- There is an indication of market inefficiencies, however, the learning results are insufficient.
- SVM learning from randomly selected data points to a "hard to learn" dataset. Precision is established as 0.33, while recall is 0.5 and the f-value 0.38.
- The subsequently extracted rules from the SVM using the rule learners See5, C4.5 and RBP/Rulex did not significantly represent what the SVM has learned.
- The one rule extracted from the SVM using Ripper significantly represents what the SVM has learned. Rule precision was established as 0.57, recall as 0.38 and the f-value as 0.46.

The Ripper rule uses technical and company specific attributes: a steep increase in the S&P index in the last two periods combined with more established companies (year founded). The rule is economically plausible since it points to a "hot issue market" phenomenon (Ibbotson & Jaffe 1975; Ritter 1984) and the age of the firms is found to be significant by Moonchul and Ritter (1999). The issue that accounting ratios did not feature in the one Ripper rule may point to the difficulty of comparing companies across different industries.

Table 15. RBP/Rulex results

Number of rules = 2	Number of antecedents = 24
RULE 1	
IF CashLiab	IS BETWEEN −0.0357694 AND 1
AND ActualOffer	IS BETWEEN −1 AND 0.484134
AND PropActOffer	IS BETWEEN −1 AND 0.603759
AND RegDays	IS BETWEEN −1 AND 0.0335572
AND GrossMargin	IS BETWEEN −1 AND 0.598339
AND GrMarginShare	IS BETWEEN −1 AND −0.26638
AND RevGrowth	IS BETWEEN −0.605131 AND 1
AND Revenue	IS BETWEEN −1 AND 0.381687
AND NASDAQ_Period_5	IS BETWEEN −1 AND 0.286907
AND NASDAQ_Period_4	IS BETWEEN −0.67481 AND 1
AND NASDAQ_Period_3	IS BETWEEN −0.183016 AND 1
AND NASDAQ_Period_1	IS BETWEEN −1 AND 0.0780277
THEN >50%	
RULE 2	
IF MarketCap	IS BETWEEN 0.131663 AND 1
AND PropActOffer	IS BETWEEN −1 AND 0.563242
AND OfferOutst	IS BETWEEN −1 AND 0.780495
AND RegDays	IS BETWEEN −1 AND 0.655597
AND GrossMargin	IS BETWEEN −0.380277 AND 1
AND RevGrowth	IS BETWEEN −0.418135 AND 1
AND NASDAQ_Period_10	IS BETWEEN −0.502027 AND 1
AND NASDAQ_Period_9	IS BETWEEN −1 AND 0.342823
AND NASDAQ_Period_7	IS BETWEEN −0.446386 AND 1
AND NASDAQ_Period_5	IS BETWEEN −1 AND 0.334158
AND NASDAQ_Period_3	IS BETWEEN −1 AND 0.770492
AND NASDAQ_Period_1	IS BETWEEN −0.906461 AND 1
THEN >50%	

The conclusions drawn from the single-industry experiments are:

- The overall error rate of 18.13% achieved by SVM learning using cross-validation points to market inefficiencies, but not to the specific factors responsible for these inefficiencies.
- SVM learning from randomly selected data points to a "hard to learn" dataset. Precision was established as 0.3, recall as 0.5 and the f-value as 0.46.
- The subsequently extracted rules from the SVM by all rule learners (See5, C4.5, Ripper and RBP/Rulex) significantly represent what the SVM has learned.

Table 16. Performance summary of RBP/Rulex

Evaluation on test data (62 items):					
Rule	Size	Used	Correct	Wrong	Certainty
1	12	10	10	0	1.00
3	12	6	5	1	0.83
		Performance summary			
		predicted			
		0	1	No classification	
Class	0	47	1	0	
	1	1	13	0	

- The attributes of See5 rules exclusively focus on accounting ratios. Higher asset to liabilities ratios and higher market capitalisation feature as making a positive contribution to higher first-day IPO returns. See5 rule accuracy is very high and rules are economically plausible. Precision was established as 0.86, recall as 1 and the f-value as 0.92.
- Similarly C4.5 focuses exclusively on accounting ratios, but adds the rule element of offered vs. outstanding shares. C4.5 rule accuracy is also very high and rules are economically plausible. Precision was established as 0.86, recall as 1 and the f-value as 0.92.
- The one rule Ripper generates, including the asset vs. liabilities ratio, is precise and economically plausible. Precision was established as 0.8, recall as 0.57 and the f-value as 0.67.
- In contrast RBP/Rulex yield two rules consisting of a mix of company specific and market sentiment elements. RBP/Rulex rules are very precise. Again RBP/Rulex rules are economically plausible, adding the “hot issues market” theme. Precision was established as 0.93, recall as 0.93 and the f-value as 0.93. The RBP/Rulex rules are therefore by a slim margin superior to See5 and C4.5 rules.

In summary, all rule learners found it substantially easier to extract rules from the dataset representing what the SVM has learned in comparison to the original data. This points to a filtering or smoothing effect as a result of SVM learning. Two major competing rule models emerge, one that exclusively focuses on accounting ratios and one that combines accounting ratios with market sentiment.

7 Conclusions

The experiments have shown how the ability of SVMs to solve problems can be combined with the benefits of extracting the symbolic representation of the knowledge contained in SVMs.

The experiments have shown that small pockets of predictability exist in the IPO market. The economic plausibility of the rule attributes associated with the predictions has been confirmed. These results are not only relevant for investment decisions in the capital market, but may be of benefit to other applications, such as software verification and safety applications.

References

Andrews R, Diederich J, Tickle AB (1995) A Survey and Critique of Techniques For Extracting Rules From Trained Artificial Neural Networks, Knowledge Based Systems, 8, pp. 373–389

Andrews R, Geva S (1996) Rules and local function networks, In Rules and Networks, R. Andrews, R. & J. Diederich (eds), Queensland University of Technology, Neurocomputing Research Centre

Cohen WW (1995) Fast effective rule induction (Ripper), AT&T Laboratories, New York: Proceedings of 12th International Conference of Machine Learning, Lake Tahoe, California, July 9–12

Gardner MJ, Altman DG (1989) Statistics with confidence. London BMG Books

Ibbotson RG, Jaffe JF (1975) Hot issue markets, Journal of Finance, vol. 30, pp. 1027–1032

Joachims T (1998) Categorization with support vector machines, learning with many relevant features', European Conference of Machine Learning. Chemnitz, Germany, April 21–23, Proceedings: Springer (1998)

Joachims T (1999) Making large-Scale SVM learning practical. In: Advances in kernel methods – Support Vector Learning. Schölkopf B, Burges C, Smola A (eds), MIT Press, Cambridge, Massachusetts

Mitsdorffer R, Diederich J, Tan TW (2001) Predicting first-day returns of cross-industry initial public offerings in the US stock market', ANNES 2001, the biennial International New Zealand Conference on Artificial Neural Networks and Expert Systems, University of Otago, Dunedin, New Zealand

Mitsdorffer R, Diederich J, Tan TW (2002) Rule-extraction from technology IPOs in the US Stock Market: The Ninth International Conference on Neural Information Processing (ANNES 02), November 2002, Singapore

Quinlan JR (1986) Induction of decision trees, Machine Learning, Vol. 1(1), pp. 81–106

Quinlan JR (1993) C4.5: Programs for machine learning. San Mateo: CA, Morgan Kaufmann

Quinlan JR (2001) Bagging and boosting, and C4.5, University of Sydney, Sydney

Moonchul K, Ritter JR (1999) Valuing IPOs, Journal of Financial Economics, Vol. 53, pp. 409–437

Refenes AN, Zapranis AD (1995) Modeling stock returns in the framework of APT: A comparative study with regression models, Neural networks in the capital market. pp. 138–161. John Wiley & Sons, Chichester

Ritter JR (1984) The "Hot Issue" market of 1980, Journal of Business, Vol. 32, pp. 215–240

Ritter JR (1991) The long-run performance of initial public offerings, Journal of Finance, vol. 46, pp. 3–27

Rulequest Research (1997) Data Mining Tools See5 and C5.0, Available at: http://www.rulequest.com/see5-info.html, viewed: 2003, October 28

Accent in Speech Samples: Support Vector Machines for Classification and Rule Extraction

Carol Pedersen[1] and Joachim Diederich[1,2]

[1] School of Information Technology and Electrical Engineering, The University of Queensland, St Lucia, Australia
[2] American University of Sharjah, Sharjah, UAE

1 Introduction

1.1 Motivation and Significance

Accent is the pattern of pronunciation which can identify a person's linguistic, social or cultural background. It is an important source of inter-speaker variability and a particular problem for automated speech recognition. This study aims to investigate the effectiveness of rule extraction from support vector machines for speech accent classification. The presence of a speaker's accent in the speech signal has significant implications for the accuracy of speech recognition because the effectiveness of an Automatic Speech Recognition System (ASR) is greatly reduced when the particular accent or dialect in the speech samples on which it is trained differs from the accent or dialect of the end-user [4] [14]. The correct identification of a speaker's accent, and the subsequent use of the appropriately trained system, can be used to improve the efficiency and accuracy of the ASR application. If used in automated telephone helplines, analysing accent and then directing callers to the appropriately-accented response system may improve customer comfort and understanding. The increasing use of speech recognition technology in modern applications by people with a wide variety of linguistic and cultural backgrounds, means that addressing accent-related variability in speech is an important area of ongoing research. Rule extraction in this context can aid in the refinement of the design of a successful classifier, by discovering the contribution of the various input features, as well as by facilitating the comparison of the results with other machine learning methods.

1.2 Overview

Current approaches to the identification of speaker accent usually require specialized linguistic knowledge or analysis of the particular speech contrasts between the accents, and often extensive pre-processing of large amounts of data. In contrast, this chapter presents an accent classification system using time-based segments consisting of Mel Frequency Cepstrum Coefficients as features and utilizing rule extraction from SVMs (support vector machines). It is applied to a small corpus of two accents of English. Its performance is compared to two other machine learning techniques, and rule extraction is performed using a combination of SVMs and a rule-based learner.

2 Accent Recognition

2.1 Accent

Each person speaks in his or her own idiosyncratic way, but groups of people of a similar geographical or sociological background can be considered to share various common patterns in their speech. These include, but are not limited to:

- Pronunciation or acoustic features: the use of particular vowel and consonant sounds and how these change when they are combined in words and groups of words, as well as stress, tempo, rhythmic, and intonational factors [30]
- Grammar and vocabulary: morphology and syntax, vocabulary and idiom [30]

The combination of these patterns contributes to a sense of “accent” or “dialect” and can vary according to geographical origin, sex, social class, age, education, and whether the language being spoken is one’s first or has been subsequently learned. Accent is usually considered to include only the effects of pronunciation or acoustic features, whereas dialect includes accent as well as grammar and vocabulary differences.

Accents are systematic and repeatable [27] [30]. Accents occur in most if not all languages with a sufficiently large number of speakers. However, accents are not set, “standard” entities, and considerable variation between people occurs. Accent is usually established early in life [30] but may be altered by, for example, living in another country for significant lengths of time, or significant speech training. A person’s accent may also change in the short term, depending on to whom he or she is talking [30]. That is, humans are able to adjust their accent (usually along cultural or social lines rather than regional/geographic lines) in order to improve understanding and/or social acceptance [30].

Despite the variability in the realization of the spoken message due to different accents, and also differences in listener characteristics (such as listener attention and familiarity with the accent and speaker), both utterance understanding and accent identity are usually preserved [27]. Tatham and Morton [27, p114] note that “Speakers and listeners recognise the utterance 'behind' the accent. That is, they can identify... the same utterance spoken in different accents, and can readily disassociate the utterance from the accent."

Accent recognition by listeners can even occur in the absence of an identifiable message, that is, in unfamiliar languages [3] and in artificially manipulated speech (e.g. playing recordings backwards, removing features [22] [23] [29]).

Modern English is an important world language, serving as an international lingua franca in business, education, international relations and the media [7]. Many, if not most, speakers of English worldwide have English as their second or even third language, so advances in computerized speech technology will increasingly have to deal with a wide variety of accents if they are to be successful on the world stage. Similarly, speech technology based on other languages will benefit from research into accent and other sources of variation in human speech.

2.2 Automatic Speech Recognition

ASR is the recognition of human speech by use of computer analysis. An input speech signal is compared to a stored model of the various elements of spoken language (usually phonemes and their combinations) and the most likely sequence of words is produced. It is used in many applications, such as data entry, voice dialing, caller routing and translation assistance. Accuracy is highly dependant on the application domain and the training data used to build the stored acoustic and language models.

The effectiveness of an Automatic Speech Recognition System is greatly reduced when the particular accent or dialect in the speech samples on which it is trained differs from the accent or dialect of the end-user [14]. Increasing the accuracy of ASRs on accented speech can be done in a number of ways. At its simplest, the accented speech can be passed through a number of ASRs trained on different accents, and the output is then evaluated for the most likely utterance. If there are a large number of accents, this "multiple processing" can be time consuming and expensive.

Alternatively, the recognizer itself can be made to identify multiple accented "versions" of the various phonemes and words. This leads to very large numbers of alternative representations within the recognizer, of words, phonemes or other units relevant to the operation of the ASR. This can actually reduce decoding accuracy [14] because of greater numbers of confusions and overlapping entities.

A third approach is to classify speech into its appropriate accent, and then to pass it through the appropriately accent-trained recognizer. The classifier would not necessarily need to fully decode the speech sample, but would only need to classify speech into the appropriate accent group.

If the aim is simply to identify an utterance as having one accent or another in, for example, a telephone call or recording, the usual approach involves training several ASRs on different varieties of accented speech, and choosing the best performer as the indicator of the accent [17] [28] [1]. The training of ASRs is labor-intensive and requires specialized phonetic knowledge to transcribe and label the data. Training ASRs also requires very large amounts of data, which is generally not available for accented speech, especially for some less studied or less populous accents. Other methods usually involve some prior knowledge or training on specific linguistic features [1]. The accuracy of such systems greatly depends on the method used, the accents investigated and the restrictions placed on the input speech samples, and ranges in the order of 65 to 98.5% [28] [9] [14].

For a simple classification task, the use of a fully developed ASR may not be required if the differences between particular accents can be learned by a supervised machine learning system. It may not even require knowledge of the specific linguistic differences between the accents of concern if the classifier can successfully learn from real speech examples. However, discovering the contribution of the various input features - e.g. through rule extraction - can aid in the refinement of the design of a successful classifier.

2.3 Mel Frequency Cepstrum Coefficients

Mel Frequency Cepstral Coefficients (MFCCs) provide an efficient means of representing the frequency characteristics of the speech waveform, and are the most widely used feature in state-of-the-art speech recognition systems.

The standard speech recognizer front end includes calculation of the 13 absolute MFCCs and their first- and second-order derivatives (a total of 39 MFCCs). MFCC extraction is carried out on a speech sample using the following steps [19] [8]:

1. Compensation for the unequal sensitivity of human hearing across frequency.
2. Spectral analysis using a Fourier transform on 20–30 ms Hamming-shaped windows (frames) of speech every 10 ms.
3. Mel-scale filtering using a bank of triangular windows which become more compact at lower frequencies, in accordance with the sensitivity of human hearing.
4. Log compression of the mel-filterbank channels to model the relationship between the intensity of sound and its perceived loudness.
5. Discrete cosine transforms to produce the cepstral coefficients.
6. Cepstral mean subtraction and energy normalisation to reduce channel effects.
7. Extraction of derivatives (first- and second-order).

Following their widespread use in the speech domain, MFCCs have also been used successfully in the classification of music samples [13].

The first MFCC (MFCC(0)) shows a close correlation to the geometric energy in the (mel filtered) speech signal. MFCC(1) represents spectral slope, but beyond that it is less clear what the individual MFCCs are representing in terms of how they relate to perceived aspects of speech and sound. Simple inversion of the MFCC extraction process does not generate a speech signal [20], since phase and fundamental frequency information are discarded. Relating individual MFCC values to individual aspects of human perception (such as pitch or particular phonemes) appears extremely difficult. This limits the use of rule extraction because rule sets may be useful during the experimental stage only.

MFCCs on a single time frame basis may not be particularly useful for accent classification, since they may encode a time too short to represent meaningful information. For use in an ASR, the MFCCs are usually combined into phoneme units; however this requires further segmentation of the speech sample and identification of phoneme units using a pre-trained system. Pure phoneme recognition rate is rather low [18] so further processing is required to produce a sequence of recognized speech, using scoring probabilities for phoneme combinations in the particular language and domain.

Since phonemes in continuous speech are approximately 60–70 ms in average duration [12] and the actual identity of the units is not of concern for simple accent classification, it may be possible to use time-based segments rather than phoneme-based segments for the simple classification task. The optimum duration of these segments would be an important part of the investigation.

3 Rule Extraction from Support Vector Machines for Accent

3.1 Support Vector Machines

Support vector machines (SVMs) are a class of algorithms which are well-suited to learning classification and regression tasks. They have been used successfully on a wide variety of tasks, including text and image classification [15] [5] as well as bio-medical applications [11]. SVMs utilize kernels to work in a high-dimensional feature space, since only inner products of data points are used rather than the input features themselves. In classification tasks, the margin between the two classes is maximized in order to find the best possible separator, and is further optimized in the presence of noisy data by the introduction of slack variables.

SVMs have been designed for high-dimensional input spaces. Speech provides the opportunity for working with a very large number of features. Very large numbers of samples of accented speech are not generally available, and the numbers of samples from the different accent groups may be imbalanced, hence investigating the performance of SVMs is an important task in these contexts. A small number of samples increases the chance of overfitting, and as a result, the performance of the SVM has to be tightly controlled [10].

3.2 Rule Extraction

There are three classes of users of speech recognition systems: (1) the engineer who explores features sets and designs the system, (2) the application expert who installs and maintains a speech recognition system (e.g. for directory assistance), and (3) the end user. The application expert is interested in the performance of the system but not necessarily in speech features while the end user is interested in a fast, reliable and accurate service and does not require detailed knowledge of the system. In this application, it is the engineer who employs machine learning and tests its performance who is interested in explanation. In this context, it is important to know *why certain input features lead to acceptable results* and while other feature sets fail.

As Craven and Shavlik (1994) [6] observe, "a (learning) system may discover salient features in the input data whose importance was not previously recognised." If a support vector machine has learned interesting relationships, these are encoded incomprehensibly as alphas and support vectors and hence cannot easily serve the generation of scientific theories. Rule extraction algorithms significantly enhance the capabilities of SVMs to explore data to the benefit of the user.

Support vector machines do not easily lend themselves to the discovery of explanations or rules that represent classification decisions. Unlike rule-based or decision tree systems, the output of the SVM is a numeric value and does not include additional information such as pattern elements or their combinations, which could be useful in explaining the knowledge obtained in the training process. Such explanations are important for the acceptance of SVM results by researchers and developers performing machine learning experimentation, and for the contribution SVMs can make to the knowledge in the domain in which they are operating. Rule extraction from SVMs is, therefore, an important advance for both the usefulness and verification of SVMs.

As indicated earlier, the relevant user in this case is the engineer/researcher who applies support vector machines for speech recognition and not the end user. That is, rule extraction from SVMs supports *experimentation* and *testing*, in particular the identification of features and feature sets that contribute to classification. A range of alternative methods are available, for instance sensitivity analysis. However, rule extraction from SVMs represents a convenient way to capture *the totality of knowledge learned by the SVM* (at least, this is the objective). The rule-based representation facilitates the comparison of SVM learning results with other machine learning techniques. Hence, the experimenter can select the best machine learning method for deployment.

Rule extraction is easily realized if SVMs are used in combination with other symbolic learning systems such as decision trees, when pattern labels predicted by an SVM (the "black box") are used as input labels for the second system (the "white box") in order to represent what the SVM has learned. The rules extracted may be few and simple, with high accuracy and fidelity [2], however, this method faces severe limitations if the "white box" learner cannot accept high dimensional input patterns (see Chap. 1). In addition, small rule sets may not fully explain the decision-making process of the SVMs, i.e. the totality of knowledge learned by the SVM. Going from the high dimensional SVM to a lower dimensional learning system represents a loss of information and as a result, reduced fidelity and explanatory capability.

3.3 Objectives

If it is accepted that the intention in the utterance "behind" the accent is unaltered despite differences in realization due to accent differences - and this would be especially true in the case of "read" speech - then by comparing the manifest speech patterns for particular utterances, an appropriately trained classifier may be able to accurately identify the accent of the speaker from features derived from the speech signal, *even without explicit decoding of the intended message itself.* Knowledge of the particular acoustic or phonetic contrasts between various accents may not be necessary if the classifier is able to "learn" from examples rather than operate using the coding of known accent-related differences. This would make the extension of the system to previously unstudied accents a simpler and less time-consuming task. In addition, if accent-related differences were already known or were discovered, they could later be used to further enhance the effectiveness of the system. Support Vector Machines provide an ideal example of a classifier which is able to work with the high-dimensional inputs provided by speech. Rule extraction from SVMs in this context is useful in advancing further classifier design and for an explanation of the knowledge obtained

This chapter presents an analysis of an accent classification system using SVMs with MFCC features in time-based segments as inputs. The length of speech sample required for good performance, as well as the duration of the temporal segments is investigated for three samples of differently accented speech. Rule extraction is undertaken in order to identify the features which contribute to classification.

4 Methodology

4.1 Speech Data and Feature Extraction

A corpus of accented speech was collected from 40 male and female subjects in two groups, Arabic (n = 27) and Indian (n = 13) accents of English. Subjects were aged between 20 and 56 years (mean 27.8 y) and had a high to very high level of spoken English proficiency. Fully informed consent was obtained. Subjects read a single page of English text on each of three topics. Read speech was chosen to provide a uniform sample space, and because it is easier to elicit than spontaneous or conversational speech. The speech samples were recorded using a unidirectional dynamic close-talk head-mounted microphone, via a mixer and USB audio interface onto computer as mono WAV signed 16 bit PCM (uncompressed) files at 16 KHz sample rate. All recordings were made in the same location under identical conditions in order to minimize channel effects.

Three sections of speech samples were chosen for initial analysis, one from each topic, and each 10 s long. Samples were trimmed to 50 ms before the start of the relevant section in order to minimize the effect of potential edge-related effects on parameters. Analysis was conducted on samples of between 1 and 10 s in duration, in 1-s steps, all starting at the same "zero" point.

The samples were processed to obtain energy and 12 basic MFCCs, their velocity and acceleration parameters (first and second order derivatives). The method included cepstral mean subtraction and energy normalization in order to minimize any recording differences. A frame shift of 10 ms was used (that is, MFCCs were calculated every 10 ms) and a Hamming window of width 25 ms was used. There was therefore some overlap between adjacent frames, however the shape of the window means that most of the energy was in the center of the segment. The processing resulted in 39 features for each frame for the duration of the speech sample, giving 3,900 features for each second of sample duration.

Because a shift of 10 ms is a very short time relative to the length of many phonemes, each feature was averaged across a number of frames in order to obtain values for larger time segments. The procedure was repeated for segment "lengths" of 10 ms (that is, no averaging) to 150 ms.

4.2 Machine Learning Experiments

The sequence of averaged MFCCs for a particular sample was used as the input feature vector for the particular subject for SVM training and testing. The samples were not divided into separate training and testing groups due to the small number of samples; instead, leave-one (speaker)-out cross-validation (LOO; built-in to SVM^{light}, one of the tools use in this study [16]) was used for performance evaluation, focusing on accuracy, precision and recall parameters. In addition, ROC curve analysis was used.

Recall and precision are defined as follows:

$$\text{Recall} = \frac{TP}{TP + FN} \tag{1}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{2}$$

where TP, FP and FN are the number of true positives, false positives and false negatives respectively.

The three topics were initially analyzed separately. Experiments were repeated for each of the sample duration and segment size combinations.

Various SVM kernel designs (linear, polynomial, RBF) were investigated for the binary classification task.

The number of MFCCs per segment was also reduced from 39 to 13 (that is, excluding the first and second order MFCC derivatives) and the analysis repeated for all duration-segment combinations.

A series of LOOs was performed using training samples from one topic set and testing samples from each of the other two topics, in order to test the effect of sample content mismatch. This was repeated for each combination of topic, sample length and segment size.

A further series of tests was performed by adding extra "non-matching" samples to the training set of the third topic (1-s sample, 100 ms segment case), and conducting LOOs. The "non-matching" samples had the same duration and segment size but were not from the same part of the speech sample as the original training set.

A subset of the 13-MFCC duration-segment combinations from each of the three topics was analyzed using a Decision Tree Learner (J48) [31] [24] and a Rule-based classifier (JRip) [31] in order to provide a comparison with the SVM results.

4.3 Rule Extraction and Evaluation

A variation of the pedagogical rule extraction method [21] [2] was used for rule extraction from the SVMs for the same subset of the duration-segment combinations for each topic that was used with the non-SVM machine learning methods.

In each topic, sample length and segment size combination, the model produced by SVM analysis was used to reclassify the original input patterns. The predicted class labels were then applied to the patterns to create a synthetic dataset which was used to train a rule-based classifier (JRip). The rules produced by the rule-based classifier were then examined in terms of accuracy and ROC curve analysis, and were compared to the performance on the original data, both in relation to the SVM and the rule-based classifier. Individual rules were examined and the elements of the rules compared with both the original JRip analysis and the original J48 analysis.

5 Results

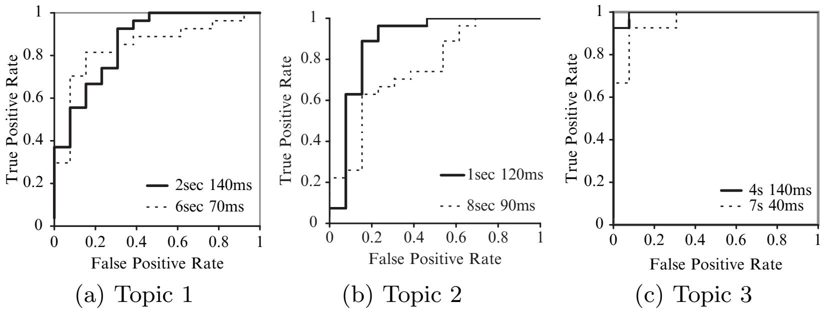
5.1 Machine Learning Experiments

Classification results varied by topic, sample length and segment size. The results for 13 and 39 features per segment were almost identical, therefore results for 13 MFCC features per segment will be presented.

Best results were obtained using a linear SVM, for the third topic and 4-s sample duration or less (Table 1). Best classification accuracy ranged from

Table 1. Performance of SVM – accuracy, duration, recall – for best cases

Topic number	Accuracy (%)	Recall (%)	Precision (%)	Sample duration (s)	Segment duration (ms)
1	75	92.59	75.76	2	140
2	87.5	96.3	86.67	1	30, 40, 60–80, 120–150
3	97.5	100	96.43	1	130
3	97.5	100	96.43	4	60, 80–110, 140



(a) Topic 1 (b) Topic 2 (c) Topic 3

Fig. 1. ROC curves

Table 2. Area under ROC curve

Topic number	Sample duration (s)	Segment duration (ms)	Area under ROC curve
1[a]	2	140	0.8604
1[b]	6	70	0.8348
2[a]	1	120	0.8832
2[b]	9	60	0.7778
3[a]	4	140	0.9943
3[b]	10	20	0.9516

[a] Best case
[b] Poor case

75% to 97.5%, with very high precision and recall. Accuracy, recall and precision fell as sample duration increased from these peak results. Accuracy was slightly higher (mean 2.5% points) for longer segment durations. Recall did not change with segment duration, and precision increased by an average of 1.1%, 3.2% and 5.6% points for the first, second and third topics respectively, as sample duration increased from 10 to 150 ms.

Selected ROC curves are presented in Fig. 1 as examples of typical best and worst cases (by accuracy, precision and recall) for each topic. Area under the ROC curve is shown in Table 2.

The effect of a mismatch between training and testing samples varied substantially between different training-testing combinations. Best results were

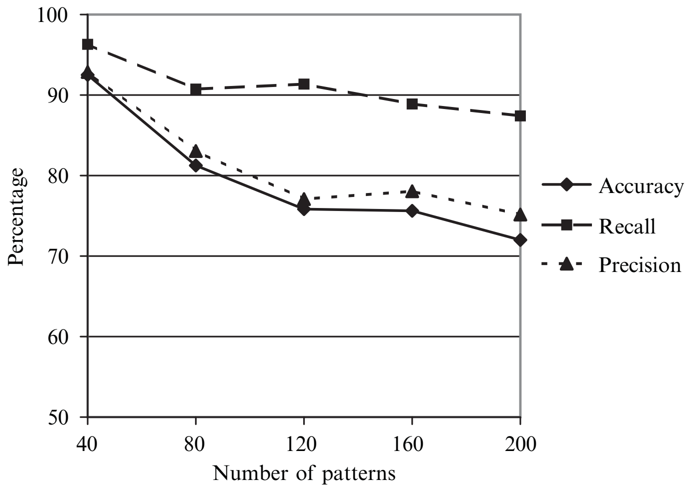


Fig. 2. Effect of adding non-matching samples from the same topic

achieved for SVM training on topic 3 and testing on topic 2, with up to 85% accuracy, 82% recall and 87% precision. In contrast, training on topic 1 and testing on topics 2 or 3 resulted in 50–70% accuracy, 97% recall and 67% precision, with almost all errors being misclassification of Indian samples as Arabic. These results varied little with increasing sample duration and segment size. Training on topic 2 and testing on topic 3 produced improved recall with longer samples and smaller frame sizes, but a drop in precision in both cases.

Adding more 1-s samples from topic 3 to the training set for topic 3 (1-s samples, 100 ms segments in all cases) had a negative effect on accuracy, precision and recall (Fig. 2). When all 1-s samples were included in the training set (a total of 400 patterns), accuracy was 68.25%, recall 86.3% and precision 72.14%. Area under the ROC curve in this case was 0.6728, compared with 0.9829 when only the first sample was used for training.

Decision Tree (J48) and rule-based (JRip) analyses were conducted on samples of 1, 2 and 4 s duration, with segments of 40 to 150 ms and 13 MFCCs per segment, in order to provide a comparison with SVM results, for most of the best cases in Table 1. Due to the small number of patterns, 40-fold cross-validation was done to most closely correlate to the SVM LOOs, and a comparison of the accuracy for each classifier (by topic) is shown in Table 3. The mean accuracy was calculated for each sample duration-text combination and these are shown in Fig. 3. J48 and JRip accuracy varied according to segment size much more than did SVM accuracy. Taking the topic 1, 2-s group as a typical example, SVM accuracy varied by up to 7.5% points across segment sizes from 40 to 150 ms, whereas J48 and JRip accuracy varied by up to 42.5 and 30% points respectively. J48 trees and JRip rules are shown in Tables 4 and 5 for the topic 1, 2-s cases.

Table 3. Comparison of LOO accuracy for various machine learning methods

Topic	JRip accuracy (%)		J48 accuracy (%)		SVM accuracy (%)	
number	Mean	SD	Mean	SD	Mean	SD
1	67.2	8.3	65.6	10.7	66.6	4.0
2	71.5	8.8	73.2	8.8	82.9	3.2
3	81.9	7.3	83.5	8.6	94.6	2.0

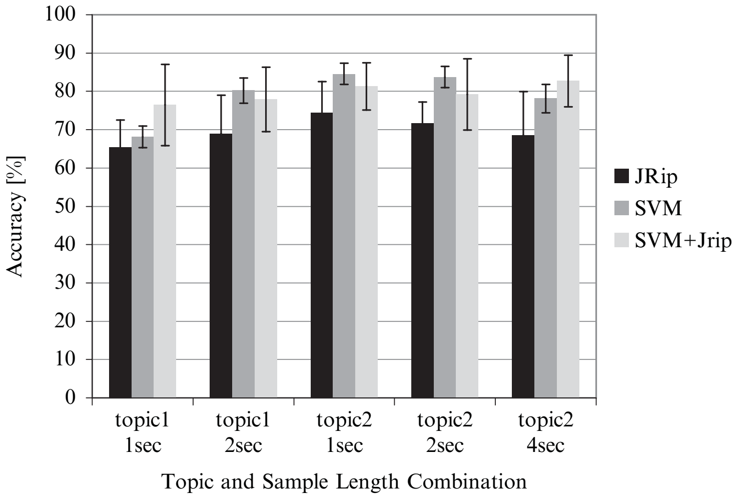


Fig. 3. Accuracy for various machine learning techniques

Table 4. Trees from J48 analysis on topic 1, 2-s samples

Segment size (ms)	Tree

40	`time6mfcc2 <= -4.595: Arabic`
	`time6mfcc2 > -4.595`
	`|    time3mfcc12 <= 1.622: Indian`
	`|    time3mfcc12 > 1.622: Arabic`
	`Number of Leaves :     3`
	`Size of the tree :     5`

50	`time5mfcc2 <= -3.177: Arabic`
	`time5mfcc2 > -3.177`
	`|    time3mfcc13 <= 0.005: Indian`
	`|    time3mfcc13 > 0.005: Arabic`
	`Number of Leaves :     3`
	`Size of the tree :     5`

Table 4. (Continued)

Segment size (ms)	Tree
60	time4mfcc2 <= -5.963: Arabic
	time4mfcc2 > -5.963
	| time2mfcc4 <= 3.721: Arabic
	| time2mfcc4 > 3.721: Indian
	Number of Leaves : 3
	Size of the tree : 5
70	time4mfcc13 <= -2.145: Indian
	time4mfcc13 > -2.145
	| time2mfcc2 <= 4.046: Arabic
	| time2mfcc2 > 4.046
	| | time24mfcc5 <= -0.078: Indian
	| | time24mfcc5 > -0.078: Arabic
	Number of Leaves : 4
	Size of the tree : 7
80	time2mfcc8 <= -4.328: Indian
	time2mfcc8 > -4.328
	| time3mfcc12 <= 4.537: Arabic
	| time3mfcc12 > 4.537: Indian
	Number of Leaves : 3
	Size of the tree : 5
90	time3mfcc2 <= -4.493: Arabic
	time3mfcc2 > -4.493
	| time2mfcc8 <= -0.409: Indian
	| time2mfcc8 > -0.409
	| | time3mfcc1 <= -1.387: Indian
	| | time3mfcc1 > -1.387: Arabic
	Number of Leaves : 4
	Size of the tree : 7
100	time3mfcc13 <= -1.73: Indian
	time3mfcc13 > -1.73
	| time17mfcc7 <= -1.242
	| | time1mfcc10 <= -2.199: Arabic
	| | time1mfcc10 > -2.199: Indian
	| time17mfcc7 > -1.242: Arabic
	Number of Leaves : 4
	Size of the tree : 7
110	time9mfcc9 <= -6.577: Indian
	time9mfcc9 > -6.577
	| time4mfcc2 <= -0.788: Arabic
	| time4mfcc2 > -0.788
	| | time1mfcc2 <= 1.778: Arabic
	| | time1mfcc2 > 1.778: Indian
	Number of Leaves : 4
	Size of the tree : 7

(*continued*)

Table 4. (Continued)

```
Segment size (ms)   Tree
120                 time1mfcc6 <= 7.001
                    |   time14mfcc7 <= -3.048
                    |   |   time4mfcc2 <= -2.548: Arabic
                    |   |   time4mfcc2 > -2.548: Indian
                    |   time14mfcc7 > -3.048: Arabic
                    time1mfcc6 > 7.001: Indian
                    Number of Leaves :   4
                    Size of the tree :   7
130                 time1mfcc6 <= 2.334
                    |   time1mfcc7 <= -2.372
                    |   |   time1mfcc4 <= 4.738: Arabic
                    |   |   time1mfcc4 > 4.738: Indian
                    |   time1mfcc7 > -2.372: Arabic
                    time1mfcc6 > 2.334
                    |   time10mfcc9 <= 3.71: Indian
                    |   time10mfcc9 > 3.71: Arabic
                    Number of Leaves :   5
                    Size of the tree :   9
140                 time1mfcc6 <= 2.285
                    |   time2mfcc9 <= 4.912: Arabic
                    |   time2mfcc9 > 4.912: Indian
                    time1mfcc6 > 2.285
                    |   time3mfcc9 <= -4.871: Arabic
                    |   time3mfcc9 > -4.871: Indian
                    Number of Leaves :   4
                    Size of the tree :   7
150                 time1mfcc6 <= 0.311: Arabic
                    time1mfcc6 > 0.311
                    |   time2mfcc2 <= -3.38
                    |   |   time6mfcc8 <= -2.558: Indian
                    |   |   time6mfcc8 > -2.558: Arabic
                    |   time2mfcc2 > -3.38: Indian
                    Number of Leaves :   4
                    Size of the tree :   7
```

5.2 Evaluation of the Rule Extraction Results

Analysis was conducted on samples of 1, 2 and 4 s duration, with segments of 40–150 ms and 13MFCCs per segment, for all three topics.

The SVM-predicted class labels were found to be the same as the original class labels for all segment durations of the topic 3, 2 and 4-s duration cases, and for over 50% of the cases in topic 1, 4-s and topic 3, 1-s duration. Therefore, further analyses of accuracy and AUC were only conducted on topic 1

Table 5. Rules extracted using JRip on original data, topic 1, 2-s samples

```
40 ms (2 rules)
(time6mfcc2 >= -2.312) => class=Indian
 => class=Arabic

50 ms (2 rules)
(time5mfcc2 >= -2.905) => class=Indian
 => class=Arabic

60 ms (3 rules)
(time28mfcc7 <= -1.046) and (time5mfcc2 <= -0.249) => class=Indian
(time4mfcc1 <= -1.587) => class=Indian
 => class=Arabic

70 ms (2 rules)
(time24mfcc7 <= -1.006) and (time4mfcc9 >= 2.148) => class=Indian
 => class=Arabic

80 ms (2 rules)
(time3mfcc2 >= -4.391) and (time11mfcc7 >= -1.175) => class=Indian
 => class=Arabic

90 ms (2 rules)
(time3mfcc2 >= -3.551) => class=Indian
 => class=Arabic

100 ms (3 rules)
(time17mfcc7 <= -1.242) and (time1mfcc10 >= -1.917) => class=Indian
(time3mfcc13 <= -1.73) => class=Indian
 => class=Arabic

110 ms (3 rules)
(time17mfcc9 >= 3.455) and (time1mfcc1 <= 1.029) => class=Indian
(time3mfcc13 <= -2.838) => class=Indian
 => class=Arabic

120 ms (3 rules)
(time14mfcc7 <= -3.057) => class=Indian
(time1mfcc6 >= 7.082) => class=Indian
 => class=Arabic

130 ms (2 rules)
(time1mfcc6 >= 2.346) => class=Indian
 => class=Arabic

140 ms (2 rules)
(time2mfcc2 >= -3.543) and (time1mfcc6 >= -0.125) => class=Indian
 => class=Arabic

150 ms (2 rules)
(time2mfcc2 >= -3.843) and (time1mfcc6 >= 0.426) => class=Indian
 => class=Arabic
```

(1 and 2-s samples), and topic 2, (1, 2 and 4-s samples). Rule content was analyzed for all data.

Accuracy and Area Under Roc Curve

Mean results are shown in Table 6. Paired samples t-tests were performed to compare the various conditions. Accuracy and AUC were significantly worse for the original JRip analysis compared to both the SVM and JRip on data labeled with SVM-predictions (hereafter called "JRip Improved") ($p < 0.001$). There was no significant difference in accuracy between the original SVM analysis and JRip Improved analysis. AUC was significantly better for original SVM analysis than JRip Improved analysis ($p = 0.001$), which was in turn significantly better than original JRip analysis (all $p < 0.001$).

Mean Accuracy and AUC for the five topic-segment duration combinations are shown in Figs. 4 and 5. Where SVM accuracy was poorest (topic 1, 1-s samples), JRip Improved analysis had significantly greater accuracy than both SVM and JRip (paired t-test, $p = 0.021$ and 0.024 respectively) but AUC was not significantly different (at around 0.6). Only in the topic 2, 4-s sample case was the AUC for JRip Improved analysis significantly different from AUC for

Table 6. Accuracy and AUC for different learning systems

Learning system	Mean accuracy (%)	Area under ROC curve
JRip on original labels	69.79	0.5965
SVM on original labels	78.95	0.7316
JRip improved	79.5	0.6778

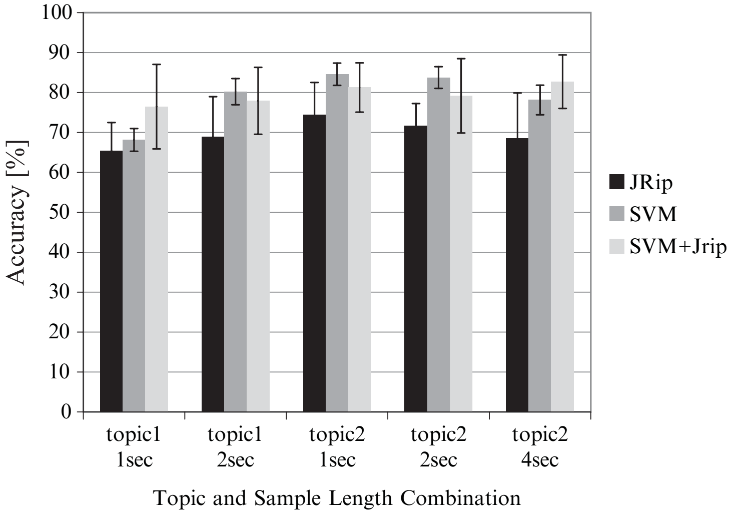


Fig. 4. Accuracy, various machine learning systems

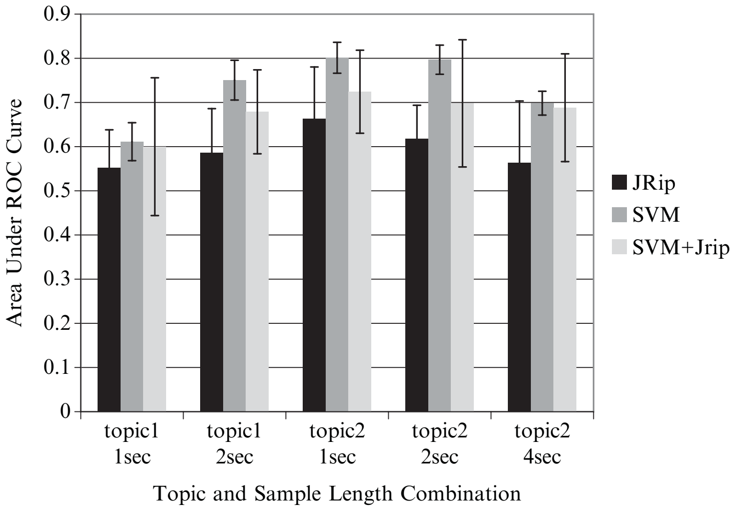


Fig. 5. Area under ROC curve, various machine learning systems

original JRip analysis (pairwise t-test, p = 0.008), whereas SVM AUC was significantly better than original JRip analysis for all but the topic 1, 1-s case.

Rules

Individual rules were examined for each topic, sample length and segment size, for J48, JRip and JRip Improved analysis.

MFCCs in the antecedents of the rules were identified by a segment time period and an MFCC number (e.g. time6mfcc2, signifying MFCC number 2 in the 6^{th} segment from the beginning). The MFCC number represents the same aspect of the speech signal regardless of the segment it is in, whereas each segment time period covered a different section of the speech signal (both in duration and location), depending on the segment size being analyzed.

In the JRip Improved analysis, there were 166 MFCCs mentioned in the antecedents of the 108 rule sets (all three topics and three sample lengths). MFCC number 3 was mentioned most often (15% of rules, in 25 rule sets), followed by MFCCs 1 and 2 (each 13.25% of rules) and MFCC 5 (12.65%), MFCC 7 (10.8%) and MFCC 6 (10.2%). The distribution of these MFCCs across the various topics varied greatly. Counting those MFCCs occurring in greater than 10% of the rule antecedents within a particular topic, MFCCs 1, 3 and 7 accounted for 83.3% of the antecedents for topic 3; MFCCs 2, 3, 5 and 6 accounted for 80.6% of the antecedents for topic 2, and MFCCs 2, 6, 7, 9 and 13 accounted for 76.8% of the antecedents for topic 1.

Sample rules are shown in Table 7 for JRip Improved analysis. In JRip Improved analysis, there were 108 rule sets (three topics, three sample lengths, 12 segment sizes). 68.5% of rule sets had only two rules, and 29.6% had three

Table 7. Rules for topic 2, 1-s samples extracted using Jrip on data labelled with SVM predictions (JRip improved)

```
40 ms segments (2 rules)
(time4mfcc5 >= 2.198) => class=Indian
 => class=Arabic

50 ms segments (2 rules)
(time3mfcc6 >= -0.076) and (time2mfcc2 >= 6.887) => class=Indian
 => class=Arabic

60 ms segments (3 rules)
(time4mfcc3 <= -5.189) => class=Indian
(time15mfcc11 >= 4.709) => class=Indian
 => class=Arabic

70 ms segments (2 rules)
(time2mfcc6 >= 1.696) => class=Indian
 => class=Arabic

80 ms segments (2 rules)
(time3mfcc3 <= -3.96) => class=Indian
 => class=Arabic

90 ms segments (2 rules)
(time2mfcc5 >= 2.514) => class=Indian
 => class=Arabic

100 ms segments (3 rules)
(time2mfcc5 >= 2.708) => class=Indian
(time1mfcc12 >= 2.972) => class=Indian
 => class=Arabic

110 ms segments (2 rules)
(time1mfcc6 >= 2.352) and (time1mfcc12 >= -2.99) => class=Indian
 => class=Arabic

120 ms segments (2 rules)
(time2mfcc6 >= -3.298) and (time1mfcc6 >= 1.41) => class=Indian
 => class=Arabic

130 ms segments (2 rules)
(time1mfcc6 >= 1.581) and (time2mfcc5 >= -3.744) => class=Indian
 => class=Arabic

140 ms segments (2 rules) 8
(time2mfcc3 <= -5.81) => class=Indian
 => class=Arabic

150 ms segments (3 rules)
(time2mfcc3 <= -4.848) => class=Indian
(time3mfcc5 >= 4.652) => class=Indian
 => class=Arabic
```

rules. Two rule sets had only one rule, that is, everything was classified as Arabic. Of the 246 rules, only 29 had more than one condition.

57% of the 60 topic-segment length combinations (topic 1, 1 and 2-s segments; topic 2, 1, 2 and 4-s segments) had at least one common MFCC mentioned in the antecedent of a rule (or in a tree) for both J48 and JRip analysis (on original data). The accuracy of JRip on these cases was significantly better than where there was no MFCC in common in the rule antecedent (73.3% vs. 65.2%, t-test $p = 0.001$). Exactly 50% of the combinations had at least one common MFCC mentioned in the antecedent in the rules of JRip and JRip Improved analysis. There was no significant difference in accuracy of JRip Improved analysis between these two groups. (77.99% vs. 77.01%). Exactly 50% of the combinations had at least one common MFCC in the antecedent of a rule in J48 and JRip Improved analysis. The accuracy of JRip Improved was significantly better if there was a common MFCC in the rule antecedent in both J48 and JRip Improved analysis (82.5% vs. 76.5%, $p = 0.0024$).

Out of the 30 MFCCs that were in common between J48 and JRip Improved analysis, only 19 were already in common between J48 and JRip (original). There was no significant difference in JRip Improved accuracy between these two groups. There were 34 rule sets where there was an MFCC in common between JRip and J48 analysis; 15 of these did not have that MFCC in common when the rules and trees for those cases in J48 and JRip Improved analysis were compared. In addition, 11 new common MFCCs had arisen between J48 and JRip Improved analysis.

6 Discussion

The performance of the SVM classifier using time-based segments of averaged MFCCs as features was very high, with up to 97.5% accuracy, with a sample length of up to only 4 s. This compares favorably with a human listener study [25] conducted using the same samples, which yielded accuracy of 92.5% (range 80–100%) after an average of 7.7 s. Interestingly, error analysis revealed that SVMs mostly made mistakes on the Indian-accented samples while humans made almost all their mistakes on the Arabic-accented samples.

SVM accuracy also compared favorably with JRip and J48 classifier accuracy. SVM accuracy was 7.3% and 7.9% points higher on average than JRip and J48 accuracy respectively. There was much less variability in SVM accuracy than for the other classifiers, across the various segment durations. This means that the choice of segment size was much less critical for the good performance of the SVM system. SVM accuracy was, overall, slightly higher for longer segment durations than shorter ones.

Classification accuracy with SVMs appears to be dependant on the content of the speech sample under investigation, as shown by the different results for the various topics. Also, when the content of a test sample is different from that on which the classifier is trained, accuracy can still be up to 85% but is often worse. Adding extra, non-matching samples in order to improve the feature-pattern ratio does not improve performance, and in fact may degrade performance further [26]. This is likely to be due to the diversity of sounds across the samples (due to diverse speech content), being greater than the difference in sound realization between the accent groups, as represented by MFCCs.

Many speech sounds are shared by different accents, and the nature of the variations that do occur can often be subtle and sparse. If strong contrasts in the speech sounds between the accents do actually occur in a short enough time (that is, over a few seconds, thereby avoiding excessive variation in content) the SVM-based classifier can be very effective in distinguishing between the accents, even without linguistic pre-processing or explicit identification of the individual contrasting speech sounds.

SVMs do not easily provide rules or explanations for the classifications that they make, but in this study the rules provided by the "white box" learner also do not easily translate to knowledge about accent differences in the speech stream. Apart from the first MFCC (usually termed MFCC(0), but here termed "mfcc1"), which shows a close correlation to log Energy of a speech signal, and the second MFCC which represents spectral slope, it becomes increasingly unclear as to what exactly the higher individual MFCCs represent, in terms of actual speech sounds, despite their demonstrated usefulness in speech recognition and accent classification. In addition, because of the small number of patterns and their high dimensionality, rules extracted by the "white box" learner can vary greatly from one fold to the next, depending on which patterns are excluded at learning time. Nevertheless, rule extraction gives some indication which features of the high dimensional input space contribute to classification.

As mentioned above, there was no significant difference in accuracy between the original SVM analysis and JRip Improved analysis. However, AUC was significantly better for the original SVM analysis than JRip Improved analysis, which was in turn significantly better than original JRip analysis. It is worth noting that rule extraction resulted in the JRip Improved analysis which is a success in itself. The more conservative AUC evaluation confirmed the overall superiority of the SVM.

Not all MFCC-based features may be important for classifier performance, as was shown by the redundancy of the first and second order derivatives. The prominence of certain MFCCs in the rulesets of the various "white box" learners is also an indication that not all MFCCs are equally important for accurate classification. However, the contribution of individual MFCCs may not be fully captured in propositional rules such as those presented in this study. Investigation of more expressive rule languages may capture relations between features that are not represented in the rules presented above, but which are nevertheless important for the good performance of the SVM learner.

Future work will focus on additional methods for feature selection, with the goal of minimizing the number of features required, and extending the ability of the classifier to handle miss-matched data. Testing on other corpora is also an important priority. Emphasis will also be on knowledge initialisation of the SVMs by the use of domain knowledge to create virtual data sets in order to enhance classifier accuracy.

References

1. Angkititrakul P, Hansen JLH (2003) Use of trajectory models for automatic accent classification. In: Proc INTERSPEECH-2003/Eurospeech-2003, Geneva, Switzerland, pp. 1353–1356, September 2003.
2. Barakat N, Diederich J (2005) Eclectic Rule-Extraction from Support Vector Machines. Int J Computational Intelligence 2(1):59–62.
3. Bond ZS, Stockmal V, Markus D, (2003) Sentence Durations and Accentedness Judgments. J Acoust Soc Am 113(4):2330–2331.
4. Caballero M, Moreno A, Nogueiras A (2006) Multidialectal Acoustic Modeling: a Comparative Study. In: Proc ITRW on Multilingual Speech and Language Processing, Stellenbosch, South Africa, paper 001, April 2006.
5. Chapelle O, Haffner P, Vapnik VN (1999) Support vector machines for histogram-based image classification: Vapnik-Chervonenkis (VC) learning theory and its applications. In: IEEE Transcactions on Neural Networks, vol. 10, no. 5, pp. 1055–1064, September 1999.
6. Craven MW, Shavlik JW (1994) Using Sampling and Queries to Extract Rules from Trained Neural Networks. In: Cohen WW, Hirsh H (eds) Machine Learning: Proceedings of the Eleventh International Conference. Morgan Kaufmann San Francisco pp. 37–45.
7. Crystal D (1997) English as a global language. Cambridge University Press, Cambridge New York.
8. Davis SB, Mermelstein P (1980) Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences. In: IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. ASSP-28, No. 4, August 1980.
9. Frid J (2002) Automatic classification of accent and dialect type: results from southern Swedish. In: Fonetic 2002 – TMH QPSR, vol. 43, pp. 89–92.
10. Furey TS, Cristianini N, Duffy N, Bednarski DW, Schummer M, Haussler D (2000) Support vector machine classification and validation of cancer tissue samples using microarray expression data. Bioinformatics 16(10): 906–914.
11. Golland P, Grimson WEL, Shenton ME, Kikinis R (2000) Small sample size learning for shape analysis of anatomical structures. In: Proc. MICCAI-00, Pittsburgh, PA, pp. 72–82, October 2000.
12. Gong Y, Treurniet WC (1993) Duration of Phones as Function of Utterance Length and its use in Automatic Speech Recognition. In: Proc Eurospeech-93, Berlin, Germany, pp. 315–318, September 1993.
13. Guo G, Li SZ (2003) Content-Based Audio Classification and Retrieval by Support Vector Machines. IEEE Transactions on Neural Networks 14(1):209–215.
14. Huang C, Chen T, Chang E (2004) Accent Issues in Large Vocabulary Continuous Speech Recognition. Int J Speech Technology 7:141–153.

15. Joachims T (1998) Text Categorization with Support Vector Machines: Learning with Many Relevant Features. In: ECML-98, 10th European Conference on Machine Learning, Heidelberg, Germany, pp. 137–142, April 1998.
16. Joachims T (1999) Making Large-Scale SVM Learning Practical. In: Schölkopf B, Burges C, Smola A (eds) Advances in Kernel Methods – Support Vector Learning, MIT Press.
17. Kumpf K, King RW (1996) Automatic accent classification of foreign accented. Australian English speech. In: Proc ICSLP 1996, Philadelphia, PA, pp. 1740–1743, October 1996.
18. Lin X, Simske S (2004) Phoneme-less heirarchichal accent classification. In: Matthews MB (ed) Signals, Systems and Computers 2004; Conference Record of the Thirty-Eighth Asilomar Conference on. vol. 2:1801–1804.
19. Milner B (2002) A Comparison of Front-End Configurations for Robust speech Recognition. In: Proc. ICASSP 2002, Orlando Florida May 2002.
20. Milner B, Shao X, (2007) Prediction of Fundamental Frequency and Voicing from Mel-Frequency Cepstral Coefficients for Unconstrained Speech Reconstruction. IEEE Transactions on Audio, Speech and Language Processing 15(1): 24–33.
21. Mitsdorffer R, Diederich J, Tan CNW (2002) Rule Extraction from Technology IPOs in the US Stock Market. In: 9th International Conference on Neural Information Processing. 4th Asia-Pacific Conference on Simulated Evolution And Learning. 2002 International Conference on Fuzzy Systems and Knowledge Discover. Orchid Country Club, Singapore, 18 November-22 November 2002.
22. Munro MJ (1995) Non-segmental factors in foreign accent: Ratings of filtered speech. Studies in Second Language Acquisition 17:17–34.
23. Munro MJ, Derwing TM, Burgess CS (2003) The Detection of Foreign Accent in Backwards Speech. In: Sole M-J, Recasens De, Romero J (eds) Proceedings of the 15th International Congress of Phonetic Sciences, (Barcelona). Causal Productions Australia. pp. 535–538.
24. Quinlan JR (2007) Data Mining Tools See5 and C5.0, Rulequest Research (2007) http://rulequest.com/see5-info.html.
25. Pedersen C, Diederich J (2006) Listener Discrimination of Accent. In: Proc Human and Machine Speech Workshop, HCSNet Summerfest '06, Sydney, Australia, p107, November–December 2006.
26. Pedersen C, Diederich J (2007) Accent Classification Using Support Vector Machines. In: Lee R, Chowdhury MU, Ray S, Lee T (eds) Proceedings 6th IEEE/ACIS International Conference on Computer and Information Science. Melbourne Australia, July 2007, pp. 444–449.
27. Tatham M, Morton K (2005) Developments in Speech Synthesis. Wiley, Chichester.
28. Teixeira C, Trancoso IM, Serralheiro A (1996) Accent Identification. In: Proc ICSLP 1996, Philadelphia, PA, pp. 1784–1787, October 1996.
29. van Els T, de Bot K (1987) The Role of Intonation in Foreign Accent. The Modern Language Journal 71(2):147–155.
30. Wells JC (1982) Accents of English: An Introduction. Cambridge University Press Cambridge New York.
31. Witten IH, Frank E (2005) "Data Mining: Practical machine learning tools and techniques, 2nd edn. Morgan Kaufmann, San Francisco.

Rule Extraction from SVM for Protein Structure Prediction

Jieyue He[1], Hae-jin Hu[2], Bernard Chen[2], Phang C Tai[3], Rob Harrison[2], and Yi Pan[2]

[1] School of Computer Science and Engineering, Southeast University, NJ 210096, China
[2] Department of Computer Science, Georgia State University, Atlanta, GA 30303 USA
[3] Department of Biology, Georgia State University, Atlanta, GA 30303-4110, USA

Summary. In recent years, many researches have focused on improving the accuracy of protein structure prediction, and many significant results have been achieved. However, the existing methods lack the ability to explain the process of how a learning result is reached and why a prediction decision is made. The explanation of a decision is important for the acceptance of machine learning technology in bioinformatics applications such as protein structure prediction. The support vector machines (SVMs) have shown better performance than most traditional machine learning approaches in a variety of application areas. However, the SVMs are still black box models. They do not produce comprehensible models that account for the predictions they make. To overcome this limitation, in this chapter, we present two new approaches of rule generation for understanding protein structure prediction. Based on the strong generalization ability of the SVM and the interpretation of the decision tree, one approach combines SVMs with decision trees into a new algorithm called SVM_DT. Another method combines SVMs with association rule (AR) based scheme called SVM_PCPAR. We also provide the method of rule aggregation for a large number of rules to produce the super rules by using conceptual clustering. The results of the experiments for protein structure prediction show that not only the comprehensibility of SVM_DT and SVM_PCPAR are much better than that of SVMs, but also that the test accuracy of these rules is comparable. We believe that SVM_DT and SVM_PCPAR can be used for protein structure prediction, and understanding the prediction as well. The prediction and its interpretation can be used for guiding biological experiments.

1 Introduction

For the past few decades, many studies have focused on the accuracy of protein structure prediction using machine learning technologies such as neural networks or support vector machine and have achieved good results [3,4,10,12,24]. In spite of this, these methods do not explain the process of how a learning result is reached and why a decision is made. It is important to be able to explain how a decision is made for the acceptance of the machine learning technology, especially for applications such as bioinformatics since the reasons for a decision is a useful guide for the "wet experiments". The extracted rules can also be used later as a basis for advanced approaches to deduce biological features.

In most of these cases, the performance of support vector machines (SVMs) is either similar to or better than that of traditional machine learning approaches, including neural networks. It is especially important for the field of computational biology because it is used for pattern recognition problems including protein remote homology detection, microarray gene expression analysis, recognition of translation start sites, protein structure prediction, functional classification of promoter regions, prediction of protein-protein interactions, and peptide identification from mass spectrometry data [17]. Nevertheless, like the neural networks, the SVMs are black box models. They do not have the ability to produce comprehensible models that account for their predictions.

Recent researches try to extract the embedded knowledge in trained neural networks in the form of symbolic rules in order to improve comprehensibility in the field of neural networks (NNs) [26–28]. These rule extraction methods serve for several purposes: to provide NNs with explanatory power, to acquire knowledge for symbolic AI systems, to explore data, to develop hybrid architectures and to improve adequacy for data mining applications [18].

With SVM, some researchers have started to address the issue of improving the comprehensibility. Rule-extraction from SVM [18] and learning-based rule-extraction from SVMs technique [1] are two examples. Some of the limitations of these two approaches are discussed in [25].

Although some researchers have started to apply SVMs and decision trees in bioinformatics areas, all of these have not integrated the merits of both SVMs and decision trees. For example, Krishnan et al. [29] have done a comparative study of SVMs and decision tree to predict the effects of single nucleotide polymorphisms on protein function. In his paper [14], Lin classified genes by names using decision trees and SVMs. The result showed that, although the prediction errors of both methods were acceptably low for production purpose, SVM outperforms decision trees. There is also some research using the decision tree to produce rules for bioinformatics, such as automatic rule generation for protein annotation with the C5.0 data mining algorithm [20] applied on SWISS-PROT [19].

In this chapter, two novel approaches of rule-extraction for understanding protein structure prediction are presented. One approach combines SVM with decision tree into a new algorithm called SVM_DT, which proceeds in four steps. This algorithm first trains SVMs. Next, a new training set is generated by careful selection from the result of SVMs. Third, this new training set is used to train a decision tree learning system and extract the corresponding rule sets. Finally, it decodes the rules into logical rules with biological meaning according to encoding schemes. Another method combines SVM with a new association rule based classifier, pattern based classification with predictive association rules (PCPAR), into an algorithm called SVM_PCPAR with the similar process applied to SVM_DT.

Since a large number of rules are difficult for researchers to interpret and analyze, we use conceptual clustering to cluster huge number of rules based on similarity, and then aggregate the rules in each cluster to generate new super-rules. These super-rules represent the consensus rule pattern and the essential underlying relationship of classification. Because the super-rules come from each clusters, the researchers can not only understand the general trend and ignore the noise, but also interactively focus on the key aspects of the domain by using super-rules and selectively view the original rules in the corresponding cluster.

Based on protein secondary structure prediction with the RS126 data sets and transmembrane segments prediction with the 165 low-resolution data sets [5], the results show that they have similar accuracy while SVM_DT and SVM_PCPAR are more comprehensible. Hence, SVM_DT and SVM_PCPAR can be used both for prediction and guiding biological experiments.

This chapter is organized as follows. Section 2 describes the method of SVM_DT and presents the experiments of protein secondary structure prediction on RS126 data sets and transmembrane segments prediction on 165 low-resolution data set. Section 3 presents the method of extracting rules from SVM based on association rule based method. Section 4 is about the rule clustering and super rules generation. Finally, Section 5 summarizes the main contribution of this chapter and discusses some issues of the methods that should be further investigated.

2 Rule Generation by Combing SVM and DT

SVM have shown strong generalization ability in many application areas, including protein structure prediction. However, it is a black box model. On the other hand, a decision tree has good comprehensibility. It motivates us to integrate merits of both support vector machine and decision tree to generate rules for understanding protein structure prediction. This approach combines SVM with decision tree into a new algorithm called SVM_DT.

2.1 SVM_DT

SVM represents novel learning techniques that have been introduced in the framework of structural risk minimization (SRM) inductive principle and in the theory of Vapnik Chervonenkis (VC) [22] bounds. SVM has a number of interesting properties, including effective avoidance of over fitting, the ability to handle large feature spaces, and information condensing of the given data set, etc.

The basic idea of applying SVM for solving classification problems can be stated briefly in two steps. First, SVM transforms the input space to a higher dimension feature space through a non-linear mapping function. Second, it constructs the separating hyperplane with maximum distance from the closest points of the training set [2].

Decision tree learning [15] is a means for approximating discrete-valued target functions, in which the learned function is represented by a decision tree. It is one of the most popular classification methods and has been used in many research areas, such as personalized recommender system based on web [6] and effective technology commercialization [7]. Learned trees can also be re-represented as sets of if–then rules to improve human readability. Let us suppose, in a set of records, each record has the same structure, consisting of a number of attribute/value pairs. One of these attributes represents the category of the record. The problem is to determine a decision tree that, on the basis of answers to question about the non-category attributes, predicts correctly the value of the category attribute. In the decision tree, each node corresponds to a non-categorical attribute and each arc to a possible value of that attribute. A leaf of the tree specifies the expected value of the categorical attribute for the records described by the path from the root to that leaf. There are many decision tree algorithms. The results of the experiment [13] show the C4.5 [20] tree-induction algorithm provides good classification accuracy.

C5.0 is a new version of C4.5. They use the gain ratio criterion, which is based on information theory and produces suboptimal trees heuristically [20]. At first, a decision tree is built using the training set. In a second step, the decision tree is pruned by replacing a whole subtree by a leaf node. If a decision rule establishes that the expected error rate in the subtree is greater than in the single leaf, the replacement takes place.

Decision trees can sometimes be quite difficult to understand. Thus, the rule sets that consist of simple if–then rules are derived from a decision tree: write a rule for each path in the decision tree from the root to a leaf. In that rule, the leaf-hand side is easily built from the label of the nodes and the labels of the arcs. Rules are ordered by class and sub-ordered by confidence, and a default rule is created for dealing with instances that are not covered by any of the generated rules. The default rule has no antecedent and its consequence is the class that contains the most training instances not covered by any rule. Each of the rule sets produced is then evaluated using the original training data and the test data.

SVM claims to guarantee generalization, i.e. its decision model reflects the regularities of the training data rather than the incapability of the learning machine. SVM reveals the classification by looking at the critical cases. On the other hand, the advantage of the Decision Trees algorithm is easily comprehensible; it describes what attributes are important for classification [14]. Thus, the motivation of combining SVM and decision tree to classify is the desire of combining the strong generalization ability of SVM and the strong comprehensibility of rule induction. Specifically, our new algorithm SVM_DT employs SVM as a pre-process of decision tree.

Suppose we are given a training data set $\mathrm{S} = \{(\mathrm{x_1}, \mathrm{y_1}), (\mathrm{x_2}, \mathrm{y_2}), \ldots, (\mathrm{x_m}, \mathrm{y_m})\}$, where $\mathrm{x_i}$ is the feature vector and $\mathrm{y_i}$ is the expected class label or target of the *i*th training instance. At first, SVMs are trained using N-fold cross validation. That is, for data set S, we divided it into N subsets with similar sizes (k) and similar distribution of classes. We perform the tests for the N runs, each with a different subset as the test set ($\mathrm{Te\_svm^i}, \mathrm{i} = 1 \ldots \mathrm{N}$) and with the union of the other N-1 subsets as the training set ($\mathrm{Tr\_svm^i}, \mathrm{i} = 1 \ldots \mathrm{N}$). Then, from each test set ($\mathrm{Te\_svm^i i}, \mathrm{i} = 1 \ldots \mathrm{N}$), based on the result of prediction $\mathrm{P^i\_svm}$, we select cases that are correctly predicted by SVMs into new data set ($\mathrm{S^i\_svm}, \mathrm{i} = 1 \ldots \mathrm{N}$). Finally, we use the original test data $\mathrm{Te\_svm^i}, \mathrm{i} = 1 \ldots \mathrm{N}$ as test data set ($\mathrm{Te\_dt^i}, \mathrm{i} = 1 \ldots \mathrm{N}$) and the union of the other N-1 subsets $\mathrm{S^i\_svm}$ as the training set ($\mathrm{Tr\_dt^i}, \mathrm{i} = 1 \ldots \mathrm{N}$) to train decision trees and induce the rule sets. In summary, the pseudo-code of SVM_DT algorithm is shown in [49]. Since support vector machine usually has strong generalization ability and we select the new data set from the correct result of SVMs as our inputs to DT, we believe that some bad ingredients of S, such as the noise, may be reduced by the process of SVMs, and some weak cases may be filtered by SVMs. It is indicated that new data set $\mathrm{S^i\_svm}$ data is better than the original training data set S for rule induction based on our experiment results that will be shown later. This is the reason why we use support vector machine as a pre-process of decision tree.

2.2 Protein Second Structure Prediction with SVM_DT

We apply the method of SVM_DT to the prediction of protein secondary structure. On one hand, the method is used to generate the rule sets for explaining how a secondary structure can be classified, and on the other hand, it is applied to evaluate the performance of the algorithm. We use RS126 [21] as a data set which was proposed by Rost and Sander. Based on their definition, it is a non-homologous set. This set was used in many researches on protein secondary structure prediction such as the experiments by Hua [10] and Kim [12]. The protein secondary structure prediction can be analyzed as a typical classification problem where the class (secondary structure) of a given instance is predicted based on its sequence features. The goal of secondary structure prediction is to classify a pattern of adjacent residues as helix (H), sheet (E) or coil (C, the remaining part) based on the idea that the segments of consecutive residues prefer certain secondary structure.

In this study, firstly, we combined orthogonal matrix and BLOSUM62 matrix [8] as encoding schemes [9]. The orthogonal encoding scheme is the simplest profile which assigns a unique binary vector to each residue, such as (1, 0, 0. . .), (0, 1, 0. . .), (0, 0, 1. . .) and so on. The BLOSUM62 matrix is a measure of difference between two distantly related proteins. Namely, the values in the BLOSUM62 matrix mean "log-odds" scores for the possibility that a given amino acid pair will interchange with each other and it contains the general evolutionary information among the protein families. This BLOSUM62 matrix was applied as an encoding scheme by converting its data range to [0,1]. In the encoding schemes, the information about the local interactions among neighboring residues can be embedded as a feature value, because the feature values of each amino acid residue in a window mean the weight of each residue in a pattern. Therefore, the optimal window length 13 was adopted by testing different window lengths from 5 to 19. We construct three one vs. one binary classifiers (H/$\sim$H, E/$\sim$E, and C/$\sim$C).

Secondly, to train the SVM, we selected the kernel function $K(x, y) = e^{-\lambda\|x-y\|^2}$ based on the previous studies [10, 12], and the parameter of the kernel function λ and the regularization parameter C were optimized based on tests [9]. With the data set, we ran sevenfold cross validation in the experiments. That is, we divided the data set into seven subsets with similar sizes and similar distribution of classes. Then, we performed the tests for the seven runs, each with a different subset as the test set and with the union of the other six subsets as the training set. In this experiment, we used SVM$^{\text{light}}$ [11] software. In each run, we fed the training data into SVM$^{\text{light}}$ to get the model and used test data as validation.

Thirdly, in order to compare the prediction result from SVM on test data to the original data set, and to see if they were consistent, we selected the instance into a new data set which was used later for building rules. We repeat the process until seven sets of new data have been finished. Then, combining six of them as a training data and original test data as test data to train decision tree of C4.5 and C4.5 rules, we get seven group rule sets. For comparison, we also applied the original train data and test data directly into C4.5 and C4.5 rules. All the average accuracy of binary classifier by three methods is shown as Fig. 1, respectively. From Fig. 1 we can see that the accuracy of binary classifier by SVM (SVM$^{\text{light}}$) is better than that of binary classifier by the decision tree (C4.5), but the accuracy of binary classifier by SVM_DT is better than that of binary classifier by the decision tree. We believe that this is a benefit from the generalization ability of SVM.

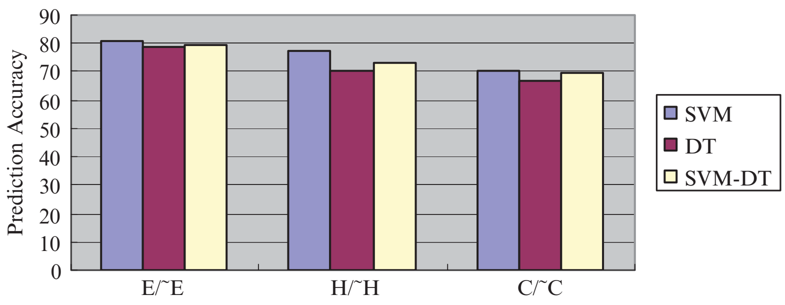


Fig. 1. Comparison of accuracy of E/$\sim$E, H/$\sim$H and C/$\sim$C with three methods

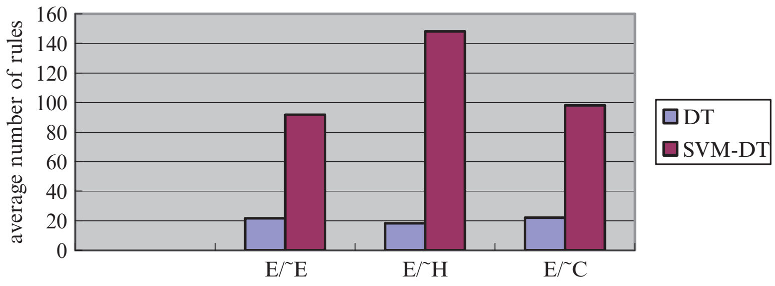


Fig. 2. Comparison of number of rules with DT method and SVM_DT method

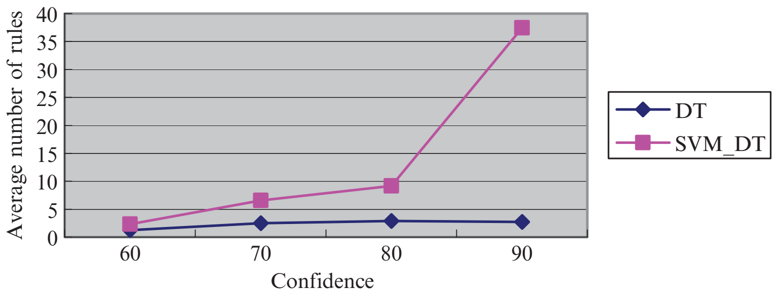


Fig. 3. Comparison average of number of rules with different confidence values of two methods

The average number of rules produced by DT and SVM_DT are shown in Fig. 2. From Fig. 2, we can see SVM_DT generated more rules than DT. In addition, Fig. 3 also shows that the average number of rules produced by SVM_DT is much more than that produced by DT under the same confidence values. This means that SVM_DT not only generates more rules but also generates rules with better quality for prediction. This observation indicates that the training data set processed by SVM is better than the original training data set for rule induction. The reason is that SVM reveals the classification by looking at the critical cases and by selecting the correct output results from SVM; SVM_DT can get the data set that has less noise.

Finally, based on the encoding schemes, we decoded the rules. We obtained a group of logical rules which have biological meaning and then we checked them in the original sequence data according to the logical rules, to verify the accuracy of them. Some of the results are shown in Table 1 with five columns. In the second column there are rules which produced by SVM_DT, and their corresponding rules with biological meaning by decoding based on the encoding schemes are shown in the third column. In the fifth column there are validation examples which are selected from the original sequence according to the logical rule in the third column and the explanation in the fourth column.

Table 1. Two example of protein secondary structure produced by SVM_DT

Rule num.	Rule produced by SVM_DT	Rule with biological meaning	Explanation	Examples
Rule 469	IF A222 $<=$ 0.035 and A260 $>$ 0 and A300 $>$ 0 THEN 'E' [90.6%]	IF Sq[2] in {'C', 'I', 'F', 'W', 'V'} and Sq[3] = 'V' and Sq[4] = 'V' THEN St[3] = 'E' [90.6%]	If the target is 'V', and one amino acid before the target is one of {C,I,F,W,V}, and the one next to the target is 'V', the second structure of the target is 'E' with 81.2% accuracy.	>2FOX: Sequence Length: 138 rule 469: the position is: 108 CVV CEE rule 469: the position is: 109 VVV EEC
Rule 471	IF A481 $>$ 0 THEN '∼E' [96.8%]	IF Sq[7] = 'A' THEN St[1] = '∼E ' [96.8%]	If the sixth amino acids after the target is 'A', the second structure of the target is not 'E' with 82.4% accuracy.	>1TGS: Sequence Length: 56 rule 471: the position is: 0 TSPQREA CCCCCCC >1UBQ: Sequence Length: 76 rule 471: the position is: 21 TIENVKA CHHHHHH

Although the accuracy of the binary classifier by SVM_DT is not better than that of the binary classifier by the SVM, we have gotten the rule sets. We also found that the rules generated have strong biological meaning. For example:

```
IF
    Sq[2]='V' and Sq[4] in {'C', 'I', 'L', 'V'}
    and Sq[5]='G'
THEN
    St[1]='E' [87.1%]
```

This rule can also be explained biologically. The amino acid in position two is the hydrophobic amino acid valine (V) and position four is one of the hydrophobic amino acids ('C', 'I', 'L', or 'V') followed by a glycine (G) in position five. If this forms a sheet (E), then the two hydrophobic amino acids point in the same direction (possibly into the core of the protein), thus stabilizing a sheet.

2.3 Transmembrane Segments Prediction and Understanding Using SVM_DT

Transmembrane (TM) proteins are the integral membrane proteins that can completely cross from the external to the internal surface of a biological membrane. TM proteins are critical targets for drug design. However, because of their hydrophobic properties, the conventional experimental approaches, such as X-ray crystallography or nuclear magnetic resonance (NMR) cannot be easily applied to determine their 3D structures. Therefore, computational or theoretical approaches have become important tools for identifying the structures and functions of TM proteins. Many significant results have been achieved in the prediction of transmembrane segments [24,29]. In spite of these results, the existing methods do not explain the process of the prediction.

In this study, the data set given by Rost et al. is tested and this is labeled as data set of 165 low-resolution. According to Rost et al. [29], the 165 proteins are expert-curated set from the SWISS-PROT database which was originally collected by Möller et al. [30]. The test method with these data sets is a sevenfold cross validation test.

In this research, we use two encoding schemes. One is the combined orthogonal and Blosum62 matrix (OB), the other is position-specific scoring matrix (PSSM) generated by PSI-BLAST. These PSSM values are position-specific scores for each position in the alignment. In this matrix, highly conserved positions have high scores and weakly conserved positions have low scores close to zero. This scheme is originally used to perform the prediction of protein secondary structure by Jones [31]. The author used this PSSM as an encoding profile for his neural network. As another approach, Kim [12] applied this matrix to train the SVM for the prediction of protein secondary structure. According to the author, this PSSM shows better performance than the frequency matrix generated by the multiple sequence alignments. Therefore, in this study, this encoding scheme is applied to test the performance in the prediction of transmembrane segments.

Four methods with different encoding schemes are used in the experiments. Because we focused on the rules extraction for understanding prediction of transmembrane segments, we should get the logical rules which have biological meaning. In the first method, PSSM matrix as encoding schemes are fed into SVM and DT(PSSM_PSSM). In the second method, PSSM matrix as encoding schemes are fed into SVM and the sequences are directly fed into DT(PSSM_SEQ). In the third method, the combined orthogonal and Blosum62 matrix as encoding schemes are fed into SVM and DT(OB_OB). In the fourth method, PSSM matrix as encoding schemes are fed into SVM and the combined orthogonal and Blosum62 matrix as encoding schemes are fed into DT(PSSM_OB).

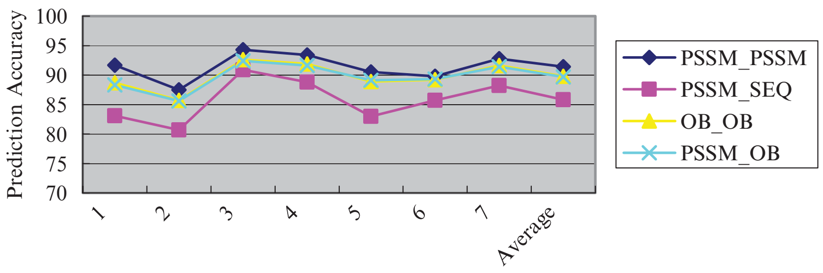


Fig. 4. Comparison of prediction accuracy of seven group rule sets with different encoding schemes

The comparison of prediction accuracy of seven group rule sets with different encoding schemes is shown in Fig. 4. From this figure, we can see the method of (PSSM_PSSM) achieves the highest average prediction accuracy. However, because PSSM is position-specific scoring matrix which is related to the context of amino acid sequence, the rules produced by DT cannot be decoded into logical rules with biological meaning. Although the method of (PSSM_SEQ) has the lower prediction accuracy than the other methods, its rules do not need to be decoded. In Fig. 4, the method of (OB_OB) and (PSSM_OB) show similar accuracy.

In order to analyze the quality of the logical rules, we compare the average rule accuracy, prediction accuracy, and percentage of rule numbers and support of seven group rule sets for the confidence of rules range of 95–100 (OB_OB). We obtain the average rule accuracy is 94.1%, average prediction accuracy is 89.6%, average percentage of rule numbers is 84.8% and average support of seven group rule sets is 83.7%. All of these show that the rules with confidence 95–100 not only have the high rule accuracy, but also have the high percentage of rule numbers and high support. The average percentage of rule numbers and support are all over 80% which means that a majority of rules obtained is of high quality.

We also analyze the rules encoded by PSSM_SEQ. The comparison of average confidence, accuracy, support, and percentage of rule numbers of seven group rule sets with confidence (90–100) is shown in Fig. 5. The results of experiment also indicate that the average prediction accuracy of rules is 93.4 for all of the rules with a confidence greater than 90. At the same time, its support is 78.0 and the percentage of rule numbers is 62.6. This means that these rules not only have high quality, but also are the majority of the rules obtained. From Fig. 5, we could find that the rules with confidence value from 97 to 99 even have a higher support value and percentage of rule numbers. The corresponding accuracies of the rules are also very high. These observations suggest that these rules are more important and valuable.

Empirical results show that the prediction accuracy is usually lower than the confidence of the rules. However, they are usually very consistent and proportional in values. A rule with a high rule confidence often produces high

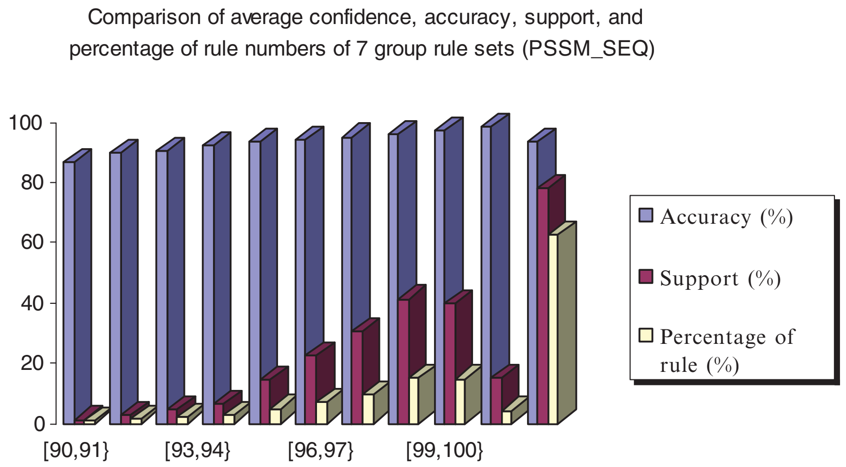


Fig. 5. Comparison of average confidence, accuracy, support, and percentage of rule numbers of seven group rule sets (PSSM_SEQ)

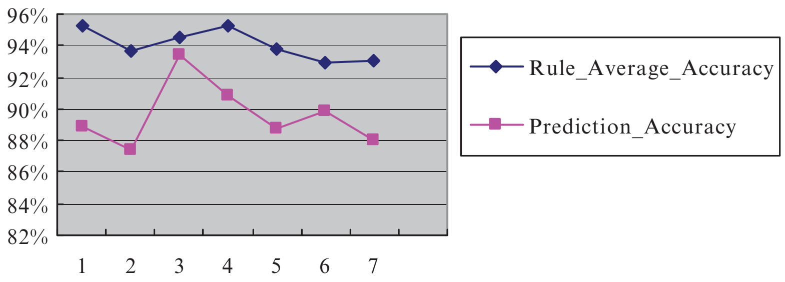


Fig. 6. Comparison of average rule accuracy, prediction accuracy of seven group rule sets for the confidence of rules range of 95–100 (OB_OB)

prediction accuracy, while a rule with a low confidence usually generates low prediction accuracy, just as Fig. 6 shows.

We decoded the rules into logical rules with biological meaning according to encoding schemes. Table 2 shows one example of logical rules and their explanation with SVM_DT (PSSM_SEQ). There are three differences between the logical rules produced by PSSM_SEQ and by OB_OB:

1. Because the rules generated by PSSM_SEQ have biological meaning, they do not need to be decoded. However, the rules generated by OB_OB should be decoded into logical rules by encoding schemes.
2. The number of rules produced by PSSM_SEQ is much greater than that of the rules produced by OB_OB. Usually, the number of rules of one set by PSSM_SEQ is about 2,000, and the number of rules in one set by OB_OB is about 200.

Table 2. One example of logical rule and their explanation with SVM_DT (PSSM_SEQ)

1	Logical rule with biological meaning	Rule 3570:
		A5 = I
		A7 = L
		A11 = F
		−> class +1 [85.6%]

2	Rule explanation	If the second amino acid before the target is 'I', the fouth amino acid following the target is 'F' , and at the same time, the target is 'L', the segment of the target is 'T' (transmembrane) with an confidence 85.60%, prediction accuracy is 83.33% when we do the experiment on the test data.

3. The rules generated by PSSM_SEQ are simpler than that of rules by OB_OB.

The reason of these differences is that the input attributes of DT in PSSM_SEQ are characters, while the input attributes of DT in OB_OB are continuous.

3 Extracting Rule from SVM Based on Association Rule

3.1 Association Rule Based Method

In the previous section, the learning-based rule extraction approaches applied decision tree as a second learning algorithm to extract the rules from SVM [25]. The advantage of DT is that the significance of rules is measured by their contribution to the overall accuracy of the classifier therefore systematic accuracy-based rule pruning is possible [32]. This method searches for rules locally based on a heuristic by adding one capable attribute at a time according to the order of goodness.

An alternative of the DT algorithm is an association rule (AR) based scheme. This method searches for all rules globally based on the cooperative prediction of several attributes and assesses each rule individually without considering the interaction with other rules [32]. The final rule set covers the training data in all possible ways hence the number of rules are usually large compared with DT method. The set with the large number of rules has the potential to find the true classification template from the training data if the over-fitting rules are pruned properly. Therefore, in AR based approach, rule pruning has been a main interest to the researchers.

Recently Hu et al. [33] attempted to apply the AR based method to extract rules from SVM on transmembrane segments prediction. They devised the pattern based classification with predictive association rules (PCPAR) scheme based on classification based on predictive association rules (CPAR) method [34] to handle the dataset with a sliding window scheme. This section begins with basic concepts of AR mining algorithm. Then traditional and recent AR mining algorithms are presented and finally rule generation method based on SVM_PCPAR scheme is described in detail.

3.2 Association Rule Mining

Basic Concepts

A formal definition of association rule mining is as follows [35].

Let $I = \{i_1, i_2, \ldots, i_m\}$ be a set of literals, or items. Let X be an itemset which is a subset of I. Let $D = \{t_1, t_2, \ldots, t_n\}$ be a set of transactions called a transaction database. Each transaction t has a transaction identifier, *tid* and a transaction itemset such as $t = (t_{id}, t$ - $itemset)$. A transaction t contains an itemset X if $X \subseteq T$.

In a transaction database D, each itemset X has a support, $supp(X)$ which is the ratio of transactions in D containing X.

$$supp(X) = |X(t)|/|D|, \tag{1}$$

where $X(t) = \{t \text{ in } D|t \text{ contains } X\}$. A large or frequent itemset is defined as an itemset whose support is equal to, or greater than, the user-specified minimal support threshold.

An association rule is an implication $X \rightarrow Y$, where itemsets X and Y are disjoint, $\mathrm{X} \cap \mathrm{Y} = \phi$. In each association rule, there are two quality measures, support and confidence. The support is the number of occurrences of each pattern and the confidence is the strength of implication. These measures are defined formally as follows:

- The support of a rule $X \rightarrow Y$ is the support of $X \cup Y$
- The confidence of a rule $X \rightarrow Y, conf(X \rightarrow Y)$ is the ratio

$$|(X \cup Y)(t)|/|X(t)|, \text{or } supp(X \cup Y)/supp(X).$$

When a transaction database D is given, mining association rules is generating all association rules which have support and confidence values equal to, or greater than, the user-specified minimal support and confidence threshold respectively.

Association Rule Mining Algorithms

Most of the traditional association rule mining algorithms are based on support-confidence model which is suitable for analyzing the market basket data [36]. For example, a Apriori is a famous and commonly-used algorithm based on this model [37].

However, in other applications such as bioinformatics or system traces, the number of occurrences may not be a good metric to measure the significance of a pattern [38]. In bioinformatics, researchers try to find statistically important sequential patterns from the sequential data. Since the frequency of each symbols in a sequence may not evenly distributed (some symbols occur more often than other symbols), a pattern with common symbols occur more often than that with rare symbols. Therefore, the frequency (support) may not always indicate the importance of a pattern. Researchers should consider both the frequent patterns and the "surprising" patterns [38]. Sometimes a few numbers of "unexpected" rare patterns could provide more information than a large number of "expected" frequent patterns. Wang and Yang adopted the information metric [39] to characterize these surprising patterns. In their research, information is used to measure the degree of "surprise" when a pattern actually occurs. Also, the information gain metric is devised to characterize the accumulated information of a pattern.

Besides the Wang and Yang's approaches, first order inductive learner (FOIL), predictive rule mining (PRM) and classification based on predictive association rules (CPAR) also applied the information metric for the rule generation. FOIL [40] is a greedy algorithm that repeatedly searches for the attribute with the highest information gain. Once this attribute is appended to a rule, all the examples which are not satisfying the rule are removed from both the positive and negative examples. After the rule is added into a rule set, this process is repeated until all positive examples in the data set are covered. For selection of attributes, "FOIL Gain" is defined such as follows to measure the information gained from appending this attribute to the current rule.

$$gain(p) = |P^*| \left(\log \frac{|P^*|}{|P^*| + |N^*|} - \log \frac{|P|}{|P| + |N|} \right), \tag{2}$$

where $|P|$ and $|N|$ are positive and negative examples that satisfy the current rule. After attribute p is added to the rule, there are $|P^*|$ positive and $|N^*|$ negative examples satisfying the new rule's body.

The FOIL algorithm was later further improved by Yin and Han to achieve higher accuracy and efficiency. It is called the predictive rule mining (PRM) algorithm [34]. PRM algorithm is a "weighted" version of FOIL [34]. In PRM, if an example is covered by a rule, without removing it, its weight is reduced by multiplying a decay factor. This algorithm was then further improved by the same authors to produce CPAR [34].

Association Rule Based Classifiers

Associative classification is an approach to integrate association rule (AR) mining and classification [41]. It applies AR mining algorithm to generate the whole set of association rules. Based on this complete rule set, a small subset of significant rules is selected and this set is used for prediction. Two typical AR based classifiers are classification based on associations (CBA) [41] and classification based on multiple association rules (CMAR) [42]. These classifiers are based on support and confidence framework which is not suitable for large dataset.

Classification based on predictive association rules (CPAR) is a more advanced AR based classifier based on information metric [34]. In CPAR, Laplace accuracy is used to measure the accuracy of rules. Given a rule r, it is defined as follows:

$$Laplace\ accuracy\ (r) = \frac{(N_c + 1)}{(N_{total} + m)}, \tag{3}$$

where m is the number of classes, N_{total} is the total number of examples that satisfy the rule's body, among which N_c examples belong to the predicted class, c of the rule. For classification, the best k rules of each class are selected from the rule sets of each class. By comparing the averaged Laplace accuracy of the best k rules of each class, the class with the best accuracy is chosen as the predicted class.

The pattern based classification with predictive association rules (PCPAR) [33] is a modified version of classification based on predictive association rules (CPAR). It is devised to handle the dataset with sliding window scheme. The rule generation part of PCPAR is the same as that of CPAR algorithm except the fact that in PCPAR each attribute window is able to participate in the AR training with different initial weight. The main differences of PCPAR and CPAR are in the post processing and the classification scheme. CPAR algorithm doesn't have any post processing step after rule generation. The PCPAR algorithm incorporates the post processing step to create more general patterns by decoding and merging the rules. For example, the following rules are the same even though the antecedents display different feature values. If we decode these rules, the antecedents of the following rules have the meaning of the amino acid 'EE' occurring position 5 and 6, 6 and 7, and 7 and 8, respectively.

$$\begin{gathered}\{87, 107\} \rightarrow \{261\},\\ \{107, 127\} \rightarrow \{261\},\\ \{127, 147\} \rightarrow \{261\}.\end{gathered}$$

As can be observed from the above, the absolute location of each attribute is not important in the sliding window scheme. Rather we should focus on the pattern of the features. With the example above, by decoding and rule merging, we can find a pattern of 'EE' occurring somewhere in a window. This pattern is simpler and also more general than the rules.

The PCPAR classification is based on the patterns created from the post process (decode-merge process) after rule mining. Each test data is checked against all the patterns of each class and the final class is determined based on the following cases. For each test instance, there are four possible situations:

1. It matches with the positive patterns only.
2. It matches with the negative patterns only.
3. It matches with both the positive and negative patterns.
4. It matches with none of them.

In the first and the second case, the final class is positive and negative class respectively. In the third case, by comparing the normalized numbers of patterns matched, the class with bigger number of patterns is selected as a final class. Finally, if no matched pattern is found with a test instance, the class is selected as a negative class by default.

Rule Generation Based on SVM_PCPAR Model

SVM_PCPAR model borrows the idea from the SVM_DT [25] for combining classifiers. This algorithm combines the SVM with a new AR based classifier, pattern based classification with predictive association rules (PCPAR) with the following process (Fig. 7).

First, SVM is trained with the two highly performed encoding profiles including the orthogonal and Blosum62 combined matrix and PSSM. Next, with the output of SVM, correctly predicted set is chosen as a new training set for AR mining. These two steps are the pre-process for the AR mining. The rationale of this pre-process is that since SVM usually has strong generalization ability, some noise or uncertain instances can be filtered out by

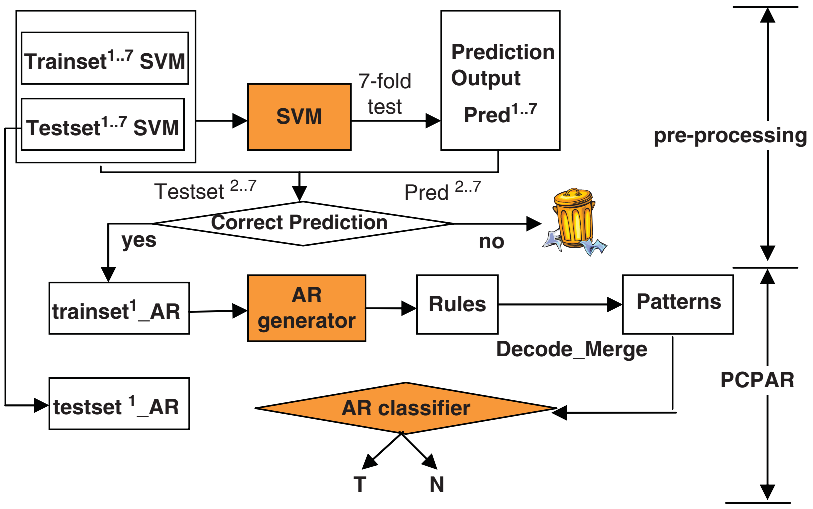


Fig. 7. SVM_PCPAR model

Positive rules					Decoded positive rule body	Positive pattern	
(428)	{31 50 170}	→	{262}	0.97	L 2 I 3 I 9	*LI*****I****	0.92
(429)	{150 231 250}	→	{262}	0.96	I 8 L 12 I 13	*******I***LI	0.92
(430)	{50 70 150}	→	{262}	0.96	I 3 I 4 I 8	**II***I*****	0.94
(431)	{30 50 130}	→	{262}	0.96	I 2 I 3 I 7	***V**I*I****	0.94
(432)	{110 130 210}	→	{262}	0.95	I 6 I 7 I 11	****LFI******	0.91
(433)	{130 150 230}	→	{262}	0.95	I 7 I 8 I 12	***FAI*******	0.91
(434)	{150 170 250}	→	{262}	0.95	I 8 I 9 I 13	****II*F*****	0.92
(435)	{80 130 170}	→	{262}	0.95	V 4 I 7 I 9	*L**LLV******	0.91
(436)	{91 114 130}	→	{262}	0.95	L 5 F 6 I 7	L*L*L*I******	0.97
(437)	{60 110 150}	→	{262}	0.95	V 3 I 6 I 8	**I***L*V****	0.91
Negative rules					**Decoded negative rule body**	**Negative pattern**	
(1)	{132}	→	{261}	1.0	K 7	******K******	0.99
(2)	{107 127}	→	{261}	1.0	E 6 E 7	*****EE******	1.00
(3)	{127 147}	→	{261}	1.0	E 7 E 8	****A*R******	0.99
(4)	{87 107}	→	{261}	1.0	E 5 E 6	******K**L***	0.99
(5)	{67 87}	→	{261}	1.0	E 4 E 5	*****SR******	1.00
(6)	{47 67}	→	{261}	1.0	E 3 E 4	******E*****A	1.00
(7)	{147 167}	→	{261}	1.0	E 8 E 9	*****GE******	1.00
(8)	{167 187}	→	{261}	1.0	E 9 E 10	****E**K*****	1.00
(9)	{81 122}	→	{261}	1.0	A 5 R 7	******R**A***	0.99
(10)	{132 191}	→	{261}	1.0	K 7 L 10	******PE*****	0.99

Fig. 8. Example of Decode_Merge process in SVM_PCPAR model

this process [25]. Third, the new training data is normalized to adjust the format for PCPAR training. Fourth, the normalized data is applied to PCPAR to train and generate the rules. Once the rule sets are generated, they are decoded into biologically meaningful rules using a decode table. The example of decoded rule bodies and patterns obtained by merging the same rules are presented in Fig. 8. The same rules are identified by examining the decoded rule body. For example, if a positive rule body is decoded into (V 3 I 6 I 8), it means that amino acids V, I, I occur at position 3, 6 and 8 in a slide window. Since the encoding profile of our AR based classifier is composed of the sliding windows of amino acid sequences, these positions could be any of (1, 4, 6), (2, 5, 7), (4, 7, 9), (5, 8, 10), (6, 9, 11), (7, 10, 12), and (8, 11, 13) with the window size 13. The decoded rule body (V 3 I 6 I 8) can be merged with (V 4 I 7 I 9) since these are the same. Because of this reason, we should rely on the relative positional expression (pattern) rather than the absolute positional information. If we use the previous example again, the positive rule body, (V 3 I 6 I 8) can be expressed as the positive pattern, (V**I*I). It means that only if this pattern comes somewhere within a window, it becomes a positive class. Here, the ‘*’ can be considered as a “don’t-care” character. Based on this kind of patterns defining positive and negative classes respectively, the test data can be classified with pattern match. The PCPAR classification algorithm performs this to determine the final class of the amino acid in the middle of a sliding window.

The first column is the positive and negative rules with the Laplace accuracy values, the second is the decoded rule bodies based on the decode table, and the third column is the patterns created from the rule merge process. The Laplace accuracy values in the third column are averaged values from the same rules.

4 Rule Clustering and Super_rule Generation

By combining SVM with decision tree, we extract rules for understanding transmembrane segments prediction. However, rules we have gotten are as many as 20,000. Such a large number of rules are difficult for researcher to interpret and analyze [48]. This can often hamper the knowledge discovery process. Clearly, it will not be satisfactory for researchers to simply use arbitrary small subset of rules because the subset of rules can't cover all the data of the domains.

Many researchers have addressed this problem by proposing a number of approaches to produce a suitable rule set. Usually, rules have to be pruned and grouped at the post mining stage, so that only a reasonable number of rules have to be inspected and analyzed. For example, one method of reduced error pruning is used in the decision tree [20] which takes each of nodes in the tree as pruning candidate. Another successful method for finding high accuracy is rule post-pruning used in C4.5 [20]. Chawla [43] presented pruning of association rules using directed hypergraph which maps a set of association rules into a directed hypergraph and systematically removing circular paths, redundant and backward edges that may obscure the relationship between the target and other frequent item. The clustering is often used for the grouping rules to deal with huge rules. Association rule clustering system (ARCS) is the example [16] which clusters all those two attribute associate rules where the right-hand side of the rules satisfies the segmentation criteria. This approach is based on the geometric properties. In the papers [44–46], the main idea of the approaches is distance based clustering for association rules. The distance is strongly correlated with support, and high support rules will on average tend to have higher distances to everybody else.

In this study, the target of the research can be described as "global" in the sense that we want to compress the rule base into smaller one set with the assurance that very little useful information is lost as possible. Moreover, the rules are not created by frequent items as the primitive association rule mining methods do. Therefore, unlike the methods in [43–46], we present the method of rule clustering and super rules generation based on the conceptual clustering [47]. In conceptual clustering, a group of objects forms a class only if it is describable by a concept. Different from conventional clustering, conceptual clustering consists of two steps: First, discover the appropriate clusters.

Second, form descriptions for each cluster. In the study, descriptions of each cluster are super rules.

Therefore, a novel approach of rule clustering for super-rule (C_SuperRule) generation is presented. We use the clustering to group huge number of rules based on similarity, and then aggregate the rules in each cluster to generate new super-rules. These super-rules represent the consensus rule pattern and the essential underlying relationship of classification. Since the super-rules come from each of clusters, the researchers not only can understand the general trend and ignore the noise but also can interactively focus on the key aspects of the domain by using super-rules and selectively view the detailed rules in the corresponding of cluster.

In this research, the rules are used for protein structure prediction by amino acid sequence. It means that if the length of the sliding windows is L, the corresponding structure of amino acid at the position of [L/2] will be predicted. Therefore, we could take the rules as follows:

For a rule $\Re : \mathrm{B} \rightarrow \mathrm{A}, \mathrm{A} \in [-1, 1]$,
B: IF Sq[1]∩Sq[2] ∩. . .∩Sq[L], Sq[i]=Q[1]∪Q[2]∪. . .∪Q[20],Q[k]∈ Λ, or Q[k] = ϕ, Λ[20] = {A,R,N,D,C,Q,E,G,H,I,L,K,M,F,P,S,T,W,Y,V}.

B could be express as a matrix M : L × 20,

$$M_{i,j} = \begin{cases} 1 \qquad if \qquad sq[i] = \Lambda[j], 1 \leq j \leq 20, 1 \leq i \leq L \\ 0 \qquad\qquad\qquad\qquad or \end{cases}$$

```
For example: IF      Sq[1] = {R,S,T}
                 and Sq[2] = {N,T}
                 and Sq[3] = {A,G}
                 and Sq[5] = {D}
             THEN
                 St[7] = N
```

The precondition part of the rule can be a matrix as Fig. 9.

We use k-means to cluster rules according to the similarity of rules. K-means clustering algorithm is the most widely used method in partition category due to its fast speed and easy understanding. The method uses an object called centroid which is the mean point in a cluster, and tries to minimize the intra-cluster distance between any point in the cluster and the centroid. We applied this method in our classification rules clustering by combining similar rules together to generate more general and error-tolerant rules. First, based on the prediction results of the rules we classify the rule set into positive rule set and negative rule set. Then, the rules in positive rule set and negative rule set are separately clustered by combining similar rules together. Next, the score matrix of each rule cluster is calculated. The score matrix of the cluster means the frequency for the specific amino acid

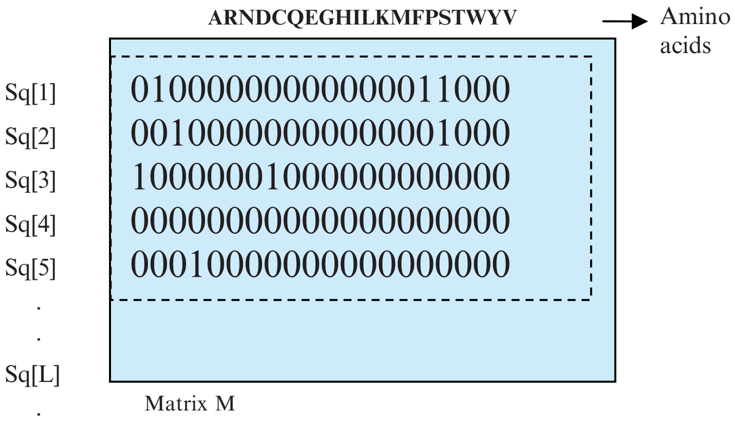


Fig. 9. The example of matrix of the rule

residue in a given window position for a cluster. It is obtained by the following formula:

$$SC_{ij} = \frac{\sum_{k=1}^{n} M_{ij}^{k}}{n} \times 100, \qquad (4)$$

where, n is the number of rules in the cluster.

Finally, the super_rules are generated according to a given threshold in the clusters. The pseudo-code of the C_SuperRule algorithm is shown in Fig. 10.

In this study, the experiment of the C_SuperRule is performed based on the rules created by the method in Sect. 2.3. We use random method to generate initial centroids positions; we set K equals to 20 for transmembrane prediction rules, 30 or 35 (depends on the result) for non-transmembrane prediction rules. Comparison of percentage of rule numbers of seven group super-rules for the accuracy range of prediction "T" is shown as Table 3. From Table 3 we can see the percentage of rule number of accuracy over 85 is about 60%. It means the super-rules have high quality. The example of super-rules and explanation is shown in [50]. The super-rules are very useful in guiding biological experiments. In the clustering, we get the cluster score matrix, such as Fig. 11; these indicate the profile of the amino acid in each position of 13 windows for transmembrane prediction. The higher frequency of the amino acid in the position implies that in this position this amino acid is more important for the corresponding structure. We believe that it will be very useful information for biology.

The experiments show that these super-rules not only have high quality but also are different from the rules before clustering because these super-rules are produced by aggregating the detailed rules and indicate the general trend.

```
C_SuperRule
input: Logical Rule set $L\_R_i$, i=1,...,N
output: Super-rule set S_R
Process:
    $S\_R = \Phi$
    FOR i = 1 to N { /* for each $L\_R_i$ set, classify into $R_+$ and $R_-$*/
        IF        $L\_R_i$ is positive rule
                  $L\_R_+ = L\_R_+ \cup L\_R_i$
        ELSE
                  $L\_R_- = L\_R_- \cup L\_R_i$
        ENDIF
    END IF
    FOR $L\_R_+, L\_R_-$
        / rules clustering by combining similar rules together /
        {rule_number1, $C\_R_+$} = k-means ($L\_R_+$)
        {rule_number2, $C\_R_-$} = k-means ($L\_R_-$)
    ENDFOR
    FOR i = 1 to rule_number1
        Calculate the cluster score matrix of $SC\_R^i_+$
        $S\_R_i$ = create_super_rule($SC\_R^i_+$)
        $S\_R = S\_R \cup S\_R_i$
    ENDFOR
    FOR i = 1 to rule_number2
        Calculate the cluster score matrix of $SC\_R^i_-$
        $S\_R_i$ =create_super_rule($SC\_R^i_-$)
        $S\_R = S\_R \cup S\_R_i$
    ENDFOR
```

Fig. 10. C_SuperRule Algorithm

Table 3. Comparison of percentage of rule numbers of super-rules which covers each accuracy range of prediction "T"

	Accuracy	Percentage of rule numbers
1	95–100	26.9
2	90–95	18.9
3	85–90	15.5
4	80–85	15.2
5	<80	23.5

5 Conclusions

The explanation of a decision is important for the acceptance of machine learning technology in bioinformatics applications such as protein structure prediction. In this chapter, we present two approaches for rule generation from SVM for protein structure prediction, and we also discuss the rule clustering for huge number of rules at the post mining stage. Empirical results on several

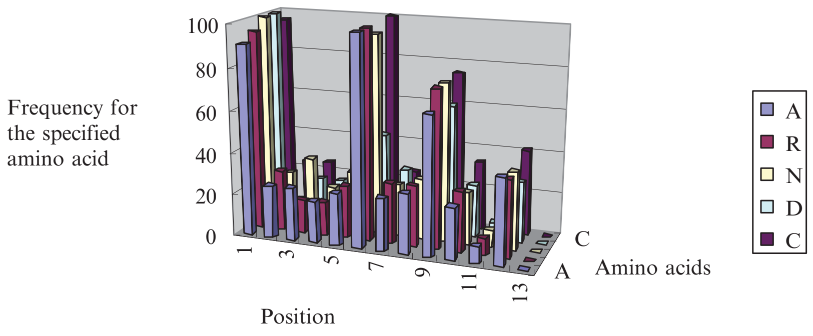


Fig. 11. One of the cluster score matrix of {A,R,N,D,C}

data sets demonstrate the efficacy of our methods. The explanation of the rules is very useful in biology domain. These rules with biological meaning not only indicate what a prediction is, but also how a decision is made. These rules can guide the “wet experiments” since they can help to identify the explicit sequence features causing the prediction and to recognize the specific mutation invalidating the prediction or presumably altering the behavior of the protein. Therefore, we can narrow the experiment scope and focus only on certain changes in the amino acid sequence.

There is still much work need to be done to improve the current approaches. First, we trained the SVM with the encoded training data. When we generated rules produced by SVM_DT, we needed to decode them into the biologically meaningful rules. One of the encoding schemes such as PSSM, we could get higher prediction accuracy than other encoding schemes. However, we were not able to decode them into the logical rules with biological meaning. One of the methods researchers used is to replace kernel functions of SVMs with PSSM. This will solve the decoding problem. But how to use it and which matrix is suitable for use as a kernel function for protein structure prediction is our future work.

Second, Andrews et al. [26] have presented a framework of rule quality evaluation, namely FACC, for evaluating the quality of the rules extracted from neural networks. In detail, the FACC framework comprises four criteria, namely fidelity, accuracy, consistency, and comprehensibility. Zhou [51] points out that the accuracy, consistency, and comprehensibility (ACC) instead of the FACC framework should be used for rule extraction using neural networks, while the fidelity, consistency, and comprehensibility (FCC) instead of the FACC framework should be used for rule extraction for neural networks. In the study, we focused on the accuracy and comprehensibility. How to evaluate the quality of the rules generated from SVM still needs to be studied.

6 Acknowledgements

The authors would like to thank Professor Thorsten Joachims for making SVM^{light} software available, Professor F. Coenen for making association rule mining software available, and thank RuleQuest Research for a ten-day evaluation licence of C5.0 software available. This research was supported in part by the scholarship under the State Scholarship Fund of China, the Nation of 985 project and Jiangsu Science Fund of BK2007105, and the U.S. National Institutes of Health (NIH) under grants R01 GM34766-17S1, P20 GM065762-01A1, and the U.S. National Science Foundation (NSF) under grants CCF-0514750, and CCF-0646102. This work was also supported by the Georgia Cancer Coalition and computer hardware used was supplied by the Georgia Research Alliance.

References

1. Barakat, N. and Diederich, J.: Learning-based Rule-Extraction from Support Vector Machine. The third Conference on Neuro-Computing and Evolving Intelligence (NCEI'04) (2004).
2. Burges, C.J.C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery, 2(2):121–167 (1998).
3. Casbon, J.: Protein Secondary Structure Prediction with Support Vector Machines (2002).
4. Chandonia, J.M. and Karplus, M.: New Methods for accurate prediction of protein secondary structure. Proteins (1999) 35, 293–306.
5. Chen, C.P., Kernytsky, A. and Rost, B.: Transmembrane helix predictions revisited. Protein Science, vol. 11, (2002), pp. 2774–2791.
6. Cho, Y.H., Kim, J.K. and Kim, S.H. :A personalized recommender system based on web usage mining and decision tree induction. Expert Systems with Applications, Volume 23, Issue 3, 1, (2002), 329–342.
7. Sohn, S. Y. and Moon, T.H.: Decision Tree based on data envelopment analysis for effective technology commercialization. Expert Systems with Applications, Volume 26, Issue 2, (2004), 279–284.
8. Henikoff, S. and Henikoff, J.G.: Amino Acid Substitution Matrices from Protein Blocks. PNAS 89, 10915–10919 (1992).
9. Hu, H., Pan, Y., Harrison, R. and Tai, P. C.: Improved Protein Secondary Structure Prediction Using Support Vector Machine with a New Encoding Scheme and an Advanced Tertiary Classifier. IEEE Transactions on NanoBioscience, Vol. 3, No. 4, Dec. 2004, pp. 265–271.
10. Hua, S. and Sun, Z.: A Novel Method of Protein Secondary Structure Prediction with High Segment Overlap Measure: Support Vector Machine Approach. J. Mol. Biol. (2001) 308: 397–407.
11. Joachims, T.: SVMlight. http://www.cs.cornell.edu/People/tj/svm_light/ (2002).
12. Kim, H. and Park, H.: Protein Secondary Structure Prediction Based on an Improved Sup port Vector Machines Approach (2002).

13. Lim, T.S., Loh, W.Y. and Shih, Y.S.: A Comparison of Prediction Accuracy, Complexity, and Training Time of Thirty_Tree Old and New Classification Algorithm. Machine Learning, Vol. 40, no. 3, pp. 203–228, Sept. 2000.
14. Lin, S., Patel, S. and Duncan, A.: Using Decision Trees and Support Vector Machines to Classify Genes by Names. Proceeding of the Europen Workshop on Data Mining and Text Mining for Bioinformatics, 2003.
15. Mitchell, M.T.: Machine Learning. McGraw-Hill, US (1997).
16. Lent, B., Swami, A. N. and Widom, J. Clustering association rules. In ICDE, 1997, pages 20–231.
17. Noble, W.S.: Kernel Methods in Computational Biology. B. Schoelkopf, K. Tsuda and J.-P. Vert, ed. MIT Press (2004) 71–92.
18. Núñez, H., Angulo, C. and Catala, A.: Rule-extraction from Support Vector Machines. The European Symposium on Artifical Neural Networks, Burges, ISBN 2-930307-02-1, 2002, pp. 107–112.
19. Kretschmann, E., Fleischmann, W. and Apweiler, R.: Automatic Rule Generation for protein Annotation with the C4.5 Data Mining Algorithm Applied on SWISS-PROT. Bioinformatics, (2001), 17(10).
20. Quinlan, J.R.: C4.5:Programs for Machine Learning. San Mateo, Calif: Morgan Kaufmann, 1993.
21. Rost, B. and Sander, C.: Prediction of protein Secondary Structure at Better than 70% Accuracy. J. Mol. Biol. (1993) 232, 584–599.
22. Vapnik, V.: Statistical Learning Theory. John Wiley & Sons, Inc., New York (1998).
23. Yang, Z.R. and Chou, K.: Bio-support Vector Machines for Computational Proteomics. Bioinformatics 20(5), 2004.
24. Sikder, A.R. and Zomaya, A.Y.: An "overview of protein-folding techniques: issues and perspectives," Int. J. Bioinformatics Research and Applications, Vol. 1, issure 1, pp. 121–143, 2005.
25. He, J., Hu, H., Harrison, R., Tai, P.C. and Y. Pan, "Transmembrane segments prediction and understanding using support vector machine and decision tree," Expert Systems with Applications, Special Issue on Intelligent Bioinformatics Systems, vol. 30, pp. 64–72, 2006.
26. Andrews, R., Diederich, J. and Tickle, A.: A Survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks. Knowledge-Based Systems (1995), 8(6), pp. 373–389.
27. Tickle, A., Andrews, R., Mostefa, G. and Diederich, J.: The Truth will come to light: Directions and Challenges in Extracting the Knowledge Embedded within Trained Artificial Neural Networks. IEEE Transactions on Neural Networks, (1998), 9(6), pp. 1057–1068.
28. Zhou., Z.-H. and Jiang, Y.: NeC4.5.: neural ensemble based C4.5. IEEE Transactions on Knowledge and Data Engineering, (2004), 16(6): 770–773.
29. Chen, C.P., Kernytsky, A. and Rost, B.: Transmembrane helix predictions revisited. Protein Science, vol. 11, (2002), pp. 2774–2791.
30. Möller, S., Kriventseva, Apweiler, E.: V. and R.: A collection of well characterized integral membrane proteins. Bioinformatics, vol. 16, (2000), pp. 1159–1160.
31. Jones, D. T.: "Protein Secondary Structure Prediction Based on Position-specific Scoring Matrix," J. Mol. Biol, vol. 292, (1999), pp. 195–202.

32. Wang, K., Zhou, S. and Y. He, "Growing Decision Trees On Support-Less Association Rules," presented at Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD'00), Boston, MA, 2000.
33. Hu, H., Wang, H., Harrison, R., P.C. Tai, and Y. Pan, "Understanding the Prediction of Transmembrane Proteins by Support Vector Machine using Association Rule Mining," presented at IEEE Symposium on Computational Intelligence in Bioinformatics and Computational Biology (CIBCB '07), Honolulu, Hawaii, 2007.
34. Yin, X. and Han, J. "CPAR: Classification based on Predictive Association Rules," presented at SIAM Int. Conf. on Data Mining (SDM'03), San Fransisco, CA, 2003.
35. Zhang, C. and Zhang, S.: Association Rule Mining: Models and Algorithms: Springer-Verlag Berlin and Heidelberg GmbH & Co. K, 2002.
36. Agrawal, R., Imielinski, T. and A. Swami: "Database mining: A performance perspective," presented at IEEE Transactions on Knowledge and Data Engineering, 1993a.
37. Agrawal, R. and Srikant, R.: Fast Algorithms for Mining Association Rules, presented at 20th Int'l Conference on Very Large Databases, Santiago, Chile, 1994.
38. Wang, W. and Yang, J.: Mining Sequential Patterns from Large Data Sets: Springer, 2005.
39. Blahut, R.: Principles and Practice of Information Theory: Addison-Wesley Publishing Company, 1987.
40. Quinlan, J. R. and Cameron-Jones, R. M.: FOIL: A Midterm report, presented at European Conference on Machine Learning (ECML-93), Vienna, Austria, 1993.
41. Liu, B., Hsu, W. and Ma, Y.: Integrating classification and association rule mining, presented at The Fourth International Conference on Knowledge Discovery and Data Mining (KDD-98)′, New York, 1998.
42. Jayasinghe S, H. K. and White S.H.: Energetics, stability, and prediction of transmembrane helices., J. Mol. Biol., vol. 312, pp. 927–934, 2001.
43. Chawla, S., Davis, J., Pandey, G. On Local Pruning of Association Rules Using Directed Hypergraphs. Proceedings of the 20th International Conference on Data Engineering, ICDE 2004: 832.
44. Gupta, G., Strehl, A. and Ghosh. J. Distance based clustering of association rules. In Intelligent Engineering Systems Through Artificial Neural Networks (Proceedings of ANNIE 1999), ASME Press, November, 1999., volume 9: pages 759–764.
45. Lele, S., Golden, B., Ozga, K. and Wasil, E. Clustering Rules Using Empirical Similarity of Support Sets Lecture Notes In Computer Science; Vol. 2226 archive, Proceedings of the 4th International Conference on Discovery Science table of contents, 2001, Pages: 447–451.
46. Toivonen, H., Klemettinen, M., Ronkainen, P. and Mannila. H. Pruning and grouping discovered association rules. In MLnet Workshop on Statistics, Machine Learning and Discovery in Databases, April, 1995: pages 47–52.
47. Han, J. and Kambr, M.: Data Mining concepts and Techniques, Higher Education Press, Morgan Kaufmann Publishers. 2001.
48. Wang, J. ed.: Encyclopedia of Data Warehousing and Minging, Hershey, PA: IGI, 2005, 190–195.

49. He, J. Hu, H. Harrison, R., Tai, P.C. and Pan, Y.: Rule Generation for Protein Secondary Structure Prediction with Support Vector Machines and Decision Tree, IEEE Transactions on NanoBioscience, Vol. 5, No. 1, March 2006, pp. 46–53.
50. He, J. Hu, H. Harrison, R., Tai, P.C., Dong, Y. and Pan, Y : Rule Clustering and Super_rule Generation for Transmembrane Segments Prediction, Proceedings of IEEE Computational Systems Bioinformatics Conference (CSB 2005), August 8–11, 2005, Califormia, USA, Poster, pp. 224–227.
51. Zhou, Z.-H. Rule extraction:using neural networks or for neural networks? Journal of Computer Science and Technology, 2004, 19(2), 249–253.

Subject Index

Accent in speech, 205
  definition and usage, 205
  machine learning experiments results, 213
  methodology, 211
  recognition and common patterns, 206
  rule extraction from SVM, 209
  rule extraction results, 218
Accent recognition, 206
  accent characteristics, 206
  ASR, 207
  MFCCs, 208
AdviceMisfit in RL, 69, 70
Algorithm, 92
  advantage, 104
  convergence properties, 93
  for deriving ellipsoid, 113
  for deriving set of rules, 121
  for incremental SVM learning, 104
Analogy and explanation, 5
ANNs, *see* Artificial Neural Networks
AR mining algorithm, 240
ARCS, *see* Association rule clustering system
Artificial intelligence (AI), 6, 8
  abduction in, 8
  and explanation, 6
  in medicine, 85
Artificial neural networks (ANNs), 3
  and explanation, 4
  and SVM in credit scoring, 37
  for share movements, 185
  stock returns, 186
Association rule (AR), 240
  associative classification in, 240
  based method in SVM rule extraction, 238
  clustering system, 244
Authorship identification and rule extraction from SVM techniques, 18
Automatic speech recognition (ASR), 205
  application, 205
  training, 208

Bag-of words (BOW) technique, 14, 16, 25
Basel II Capital Accord in credit scoring, 53
Bioinformatics, SVM role, 228
Biological experiments by SVM_DT and SVM_PCPAR, 229
Black box model of SVM, 228
Black-box classifiers, 84
  for regulatory requirements reduction, 84
  patient treatment, influence on, 85
Breadth-first spanning tree (SVMT-rule), 135, 142, 143
  and DFS SVMT-rule, 159
  in cancer diagnosis, 156
Burges algorithm in SVMs, 175

C4.5 and C5.0 tree-induction algorithm, 230
C4.5 tree induction techniques, 46, 230
CAD, *see* Computer-aided diagnosis
Cancer diagnosis, BFS SVMT-rule technique in, 156
Capital asset pricing method (CAPM), 185
CART tree induction techniques, 46
Causal explanations, 4, 7
Centroid for error-tolerant rules, 245
CFCM algorithm, numerical estimation, 178
CFCM and Fixed Point algorithms, comparison, 179
Class bias ratio (CBR), 152
2-Class SVM tree, 137
  testing algorithm, 144
Classification based on associations (CBA), 241
Cleveland Heart Disease and CFCM prototypes, 179
Cleveland heart, for rule extraction algorithm, 96
Clustering algorithm, 121
  determine_prototypes, 121
Clustering appropriate coefficients $f(k)$ context determination, 178
Clustering vectors in prototype selection, 176
CMAR, *see* Classification based on multiple association rules
Code availability in SVM rule extraction techniques, 53
Comparable firm analysis, for IPOs, 187
Computational biology, SVM role, 228
Computer-aided diagnosis
  FDA and pre-market approval, 85
  for automated interpretation, 85
  full-blown clinical trial, 85
  lung CAD dataset, 96, 99
Conditional Fuzzy C-Means (CFCM), 177
Conditional probability and explanation, 4, 5
Corruption Perceptions Index, SOMs usage, 57
CPAR, *see* Classification based on predictive association rules
Credit scoring application, 53
  in Basel II and model classification, 53, 54
Cross-industry IPOs, 188
  leave-one-out cross-validation results, 194
  Ripper rules, 195
  See5 confusion matrix for, 195
C_SuperRule
  generation, 244, 245
  pseudo-code, 246
Cyan rectangles, 89

Data mining capabilities of SVMs, 11
DataMisfit, in reinforcement learning, 69
Decision tree (DT), 215, 228
  analysis of SVM, 215
  induction techniques, 45
  learning and advantages, 230
Decompositional rule extraction techniques, 12, 39–41
Depth-first spanning tree, for SVM tree construction, 142, 143, 154
Discounted cash flow (DCF) analysis, for IPOs, 187
DT and SVM_DT rules, comparision, 233

Eclectic rule extraction techniques, 12
Encoding schemes used in protein structure prediction, 232
Explanation
  and analogy, 5
  and generalization, 6
  best explanation identification, 8
  casual and step-wise explanations, 7
  definition, 3
  theoretical approaches, 4
  types, 4
Explanation capability of symbolic artificial intelligence, 9
Explanation-based generalization in rule extraction, 6
  and stages, 7
  components, 7

Extraction steps of MFCCs, 208

Fine Needle Aspiration (FNA), 98
First order inductive learner (FOIL), 240
Fixed Point and CFCM algorithms, comparison, 179
FOIL Gain, 240
Food and Drug Administration (FDA), and CAD systems, 85
Fraud detection
  method, 158
  rule extraction statistics for, 157
  use SVMT-Rule rule extraction, 158
Fu's algorithm, 146
Fung, G, algorithm by, 44
Fuzzy logic algorithms, 126
Fuzzy rules (F-rules), 164, 165

Gaussian kernel, 111
Gestalt theory in psychology, 6
Gram-Schmidt orthogonalization, 116
Group rule sets, average rule accuracy comparison, 237

Hyper-rectangle rule extraction (HRE) technique, 136
  from 2-class SVM, 147
  from one-class SVM, 150
Hyperplane-based classifiers
  rule extraction from, 87
  support vector machines for, 83

Incremental approach in credit scoring, 57
Inductive Logic Programming (ILP), 74
  skill transfer, 75
Information retrieval system in rule extraction techniques, 14, 15
Initial public offerings, 185
  as case study, 186
  computer software and services, 189
  cross-industry, 188
  representation, 191
  single-industry, 192
  valuation of, 187
Interval-type rule, for support vector machines (SVM), 123
Inverse classification for credit scoring, 56

Ionosphere
  datasets and CFCM prototypes, 179
  for rule extraction algorithm, 96
IPO, *see* Initial public offerings

J48 and JRip analysis of SVM, 215
J48, JRip analyses, rules for, 221
JRip Improved analysis, 221

k-nearest neighbor (kNN) learning algorithm, 174
Kernel reduction methods, 167, 232
Knowledge expression, 111
Knowledge fusion problem in SVM rule extraction techniques, 51
Knowledge-Based Support Vector Regression (KBKR), 70
Knowledge-based SVMS and rule extraction, 101

Laplace accuracy, CPAR, 241
Learning process and explanation-based generalization, 7
Least squares SVM's, rule extraction for, 88
Linear dependency removal from SVM, 168
Linear fisher discriminant (LFD), rule extraction for, 88
Linguistic interpretation of F-rules, 165
Log-odds, 231
Logic association rules for SVMT rule extraction, 145
LOO accuracy comparison for machine learning method, 216
Lung CAD algorithm, 99
LV-SVMs, *see* Least Squares SVM's

Machine learning experiments, 212, 213
  results, 213
  technologies, 227
Mc-Nemar Test, 188, 195
Mean accuracy and AUC of machine learning system, 220
Mel Frequency Cepstral Coefficients, 208
Methodology of accent recognition
  machine learning experiments, 212
  rule extraction and evaluation, 213
  speech data and extraction, 211

MFCCs, *see* Mel Frequency Cepstral Coefficients
ModelSize, in reinforcement learning, 69
Multidimensional Scaling (MDS) algorithm, 171

NASDAQ computer index, 189, 191
Nearest Neighbor Rules (PN-rules), 166
Neural networks (NNs), 165, 228
  and support vector machine (SVM), 109
  for protein structure prediction, 227
  rule extraction for, 6
  rule set in, 13
Nuclear Magnetic Resonance (NMR), 235
Nunez's extraction method from SVM, 136

OB, *see* Orthogonal and Blosum62 matrix
One-*versus* rest technique, for SVM decision function determination, 127
Optimal rule, 90, 91
Orthogonal and Blosum62 matrix, 231, 232, 235
Orthonormal vectors, 113, 116
  determine_next_axis_vertex_procedure, 115

Pattern based classification with predictive association rules (PCPAR), 229, 239, 241, 242
  algorithm, 241
  classification, 241
Pedagogical rule extraction from SVMs, 17, 39–41
Phonemes in speech, 209
Pima Indians Diabetes and CFCM prototypes, 179
PMA, *see* Pre-market approval
Policy-transfer rules in RL, 72
Position-specific scoring matrix, 235
Post-pruning method for transmembrane segments prediction, 244
Pre-market approval, for clinical trial, 85
Precision in machine learning experiments, 212
Predictive association rules classification, 239–241
Predictive rule mining (PRM), 240
PRM algorithm, 240
Probability of default (PD), 54
Probability theory and explanation, 5
Projection matrix, 116
Propositional logical rules and prototype-based rules, 163
Protein structure prediction, 227, 245
  amino acid sequence, 245
  by SVM_DT, 231, 234
Prototype algorithms, 174
Prototype selection, methods, 176
Prototype threshold rules (PT-rules), 165, 166
Prototype vectors, 136
  and support vectors, 110
  for ellipsoid, 136
Prototype-based rules (P-rules) and SVMs
  advantages, 163, 164
  informative prototypes, 174
  limitations, 174
  types, 165, 166
Proximal SVMs (PSVM), rule extraction for, 88
PSSM, *see* Position-specific scoring matrix
PSSM SEQ (7 group rule sets), rule number, 237

Q-functions and learning in RL, 68–71
Quadratic programming (QP), 36
  and SVM, 168

Radial basis function networks (RBFN), 133
Rapid back propagation, 193
  performance, 201
  use of Rulex, 193
RBP, *see* Rapid Back Propagation
Recall in machine learning experiments, 212

Reinforcement learning (RL), 67
  advice taking, 68
  rule extraction, 71
  SVMs in, 69
Relational reinforcement learning (RRL), 81
Relevance Vector Machines in SVM+Prototypes algorithm, 43
Removed linearly-dependent support vectors (RLSV) in SVM, 168, 171
Re-RX rule extraction technique, 48
  basic idea, 48
  in credit scoring application, 49
  outline of, 49
Ripper rule learner, for classification rules induction, 193
Ripper technique and rule set in SVM rule extraction, 45, 54, 55
RoboCup project, and transfer learning techniques, 76
  and skill-transfer experiments, 79
  policy-transfer experiments, 78
ROC curves, accuracy and area, 214, 220, 221
RS-SVM in SVs number reduction, 171
RS126 data set in protein structure prediction, 231
Rule extraction from support vector machine (RulExSVM), 136
Rule extraction, 210, 228
  algorithm, 92, 127, 210
    effectiveness, 96
  and evaluation, 213
  and knowledge-based SVMS, 101, 102
  by JRip, 219
  for neural networks, 84, 109
    extraction methods, 110
  for regulatory experiments reduction, 86
  from hyperplane classifiers, 87
  from reinforcement learning, 71
    policy transfer rules, 72
    skill transfer rules, 73, 75
  from SVM, 135, 228, 229
    SVM_DT, 228
    SVM_PCPAR, 229
  from SVM for accent
    objectives, 211
    SVM, 209
  from SVM techniques, 3
    alternatives to, 56
    and authorship identification, 18
    and boosting trials, 20, 24
    and emotion classification problem, 20, 24
    and topic detection problem, 19
    classification and regression tree, 23
    classification system, 26
    evaluation, 40
    generalisation improvement and data mining capabilities, 11
    information retrieval, 14
    SVM+Prototypes method, 42
    translucency, 11–13
    transparency, 10
    usages, 36, 37
    user explanation capability, 9, 10
  in transfer learning, 67
  methods
    for neural networks, 248
    from SVM, 228
  point coverage maximization criteria, 91
  results, 213
  use of support vectors, 110
  volume maximization criteria, 90
Rule induction techniques, 45
Rule mining
  algorithms, 239
  concepts, 239
Rule traces uses in symbolic AI, 9, 10

Schölkopf algorithm, 174
Securities and Exchange Commission (SEC), 186
See5/C5, 193
  for data analysis, 193
  for single-industry IPOs, 198
Self Organizing Maps (SOMs), usages in credit scoring, 56
Sensitivity analysis and inverse classification in credit scoring, 56
Similarity-Based Methods (SBM) framework, 164

Single-industry IPOs
  attribute summary of, 192
  leave-one-out results for, 197
  rule extraction from SVM results for, 196
Skill transfer algorithm in pseudocode, 76
Skill transfer rules in RL, 75
Software verification in SVM technique, 10
Speech accent classification
  data and extraction, 211
  recognition systems, 210
  signal and applications, 207
SRM, *see* Structural risk minimization
Statistical learning theory, 109
Step-wise explanations, 4, 7
Structural risk minimization, 229
Super rule generation and rule clustering, 244, 246
Support vector clustering (SVC), 136
Support vector machines (SVMs), 209, 227–229, 247, 248
  algorithm clustering, 84
  and ANN, comparison of, 37
  and decision tree combination, 230
  and quadratic programming, 168
  and SVM-rule, 157
  and SVMT rule, 156, 159
  classification, 126
  confusion matrix for, 194
  error analysis, 223
  focus, 135
  for hyperplane-based classifiers, 83
  for protein structure prediction, 227
  for share movements, 185
  hyper-rectangle rule extraction (HRE), 136
  in reinforcement learning, 69
  in rule extraction, 3, 16, 49
    alternatives to, 56
    black box model, regression and code availability, 53
    classification scheme, 38
    for speech accent classification, 205, 209
    high dimensional data and knowledge fusion problem, 51
    rule expressiveness, 49
    SVM model, 37
    SVM+prototypes methods, 42
    usages, 36, 37
  interval-type rule for, 123
  learning techniques, 22
  linear algorithm implementation, 192
  linear separation, 103
  neuronal networks, 109, 191
  one-*versus*-rest method for, 127
  optimization, 35, 169
  performance, 214
  problems solving classification, 230
  properties, 110, 229
  statistical learning theory, 109
  ten-fold cross validation, 96
  training and prediction, 188
Support vectors
  and prototype vectors, 110
  and RulExSVM, 136
  ellipsoid determination, 113
  for classification task and knowledge expression, 111
  for unitary vectors selection, 123
  optimal number of, 171
  procedure
    determine_first_axis_vertex, 114, 123
    determine_next_axis_vertex, 115, 124
    determine_second_vertex, 115
  removal of, 169
  separation surface, 119
  update_prototype procedure, 115
SVM logarithm, cost function and mean accuracy dependence, 172, 173
SVM rule extraction techniques, 9, 36, 52
  alternatives to, 56
  black box model, regression and code availability, 53
  classification scheme, 38
  for speech accent classification, 205, 209
  high dimensional data and knowledge fusion problem, 51
  rule expressiveness, 49
  SVM model, 37
  SVM+Prototypes method, 42
  usages, 36, 37

SVM trees-rule
  algorithms, 143
  and SVM rule, 136, 156, 159
  and SVMT rule, 136
  benefits, 135
  2-class SVM tree, 137
    class imbalance and overlap, coping with, 145
    testing algorithm, 144
  data partitioning method, 140
  datasets classification, 135
  extraction
    logic association rules, 145
    SVM nodes interpolation, 146
  for face membership authentication application, 137
  for traditional computational intelligence, 136
  fraud detection using, 158
  hyper-plane, 153, 155
  one-class SVM for, 152
  procedure, 151
  spanning, 141
  use of synthetic dataset, 152
SVM+Prototypes method in rule extraction, 42, 43
SVM_DT (SVM with decision trees), 227
  and DT, average number of rules, 233
  logical rules, 233
  logical rules and difference, 237, 238
  method, 231
  protein structure prediction, 231
SVM_PCPAR, 242
  decode merge process, 243
  method, 229
  rule generation, 242
Symbolic artificial intelligence system, 9

Takagi Sugeno (TS) fuzzy inference system, 167
Ten-fold cross validation, for SVM parameter, 96
Text classification, pre-processing in, 16
Text extraction in text classification, 16
TM proteins prediction by SVM_DT, 235
  encoding schemes used, 235
  group rule sets used, 236
Topology-preserving maps, *see* Self Organizing Maps (SOMs)
Transfer learning technique in rule extraction
  advice-taking, 68
  case study, 76
  goal, 68
  rule extraction in reinforcement learning, 71
  SVMs in reinforcement learning, 69–71
Translucency in rule extraction techniques, 39
  and rule quality, 13
  in neural network, 12
Transmembrane (TM) proteins, 235
Transparency in SVM technique, 10
Trees from J48 analyses, 216
Trepan rule extraction algorithm, 47
  advantages and disadvantages, 47
  trepan tree rule set, 54

Unitary vector, 114
  linear subspace usage, 116

Vapnik chervonenkis (VC), 229
Venture capitalists, IPOs use by, 186

WDBC, *see* Wisconsin Diagnosis Breast Cancer
Wet experiments, 248
White box, 224
Wisconsin Diagnosis Breast Cancer, 96
  fine needle aspiration (FNA), 98
  for rule extraction algorithm, 96
  LP-SVM algorithm, 98

Author Index

Altman, D.G., 188, 195
Andrews, R., 8–13, 26, 38, 41, 109, 110, 128, 186, 193, 248

Barakat, N., 13, 17, 41
Berkhin, P., 17
Berthold, M., 126
Blake, C.L., 127
Burges, C., 169, 175

Cawsey, A., 14
Chawla, S., 244
Chen, Y., 167
Chiu, T., 17
Clouse, J., 69
Cohen, W.W., 193
Cortes, C., 109
Craven, M., 11–13, 109, 110
Craven, M.W., 40, 210
Cristianini, N., 109

Davis, R., 9
Diederich, J., 13, 17, 41, 51
Domingos, P., 126
Downs, T., 168, 171
Driessens, K., 69
Duda, R., 118, 128, 130
Dzeroski, S., 69

Frank, E., 29, 109, 126, 128
Fu, X., 136
Fung, G., 41, 43, 53

Gallant, S., 9
Gardner, M.J., 188, 195
Geva, 193
Gilbert, N., 9
Golea, M., 12, 13
Grochowski, M., 174
Guyon, I., 131

Han, J., 240
Hand, D., 126
Hayward, R., 27
Hu, H., 232, 238
Hua, S., 231, 232

Ibbotson, R.G., 187, 199

Jaffe, J.F., 187, 199
Jankowski, N., 174
Joachims, T., 23, 188, 192, 194
Jones, D.T., 235

Kaufman, L., 118, 131
Kim, H., 231, 232, 235
Kohonen, T., 56
Krishnan, 228
Kwok, J.T., 171

Łęski, 177
Lin, S., 228
Litt, A., 3–5, 8

Ma, J., 127
Martens, D., 5, 9
Martinez, T.R., 174

Merz, C.J., 127
Mitchell, T., 126
Mitra, S., 109, 128
Mitsdorffer, R., 186
Möller, S., 235
Moonchul, 187, 199
Moore, J.D., 10
Morton, K., 5, 28, 84, 135, 206

Núnez, H., 110, 130, 133

Pedrycz, W., 177

Quinlan, J.R., 193

Refenes, A.N., 186
Ritter, R.J., 187, 188, 199
Rost, B., 231, 235
Rousseeuw, P. J., 118, 131

Salzberg, S., 126, 127
Sander, C., 231
Schölkopf, B., 175, 179
Shavlik, J.W., 11–13, 40, 109, 110, 210
Shawe-Taylor, J., 109
Stone, P., 76, 77
Strang, G., 112, 116
Sutton, R., 76, 77
Swartout, W.R., 10

Taskar, B., 11, 27, 28
Tatham, M., 206
Thagard, P., 3–5, 8
Tickle, A., 11–13, 26, 41, 109
Tipping, M., 43
Torrey, L., 72, 73, 75–77
Tsang, I.W., 171

Utgoff, P., 69

Van Gestel, T., 58
Vapnik, V., 109, 127
Vapnik V.N., 34

Wang, J.Z., 167
Wang, W., 240
Wilson, D.R., 174
Witten, I.H., 29, 109, 126, 128

Yang, J., 240
Yin, X., 240

Zapranis, A.D., 186
Zhang, Y., 136
Zhao, Y., 127
Zhou, Z., 128
Zhou, Z.-H., 248