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.

D. Martens et al.: *Rule Extraction from Support Vector Machines: An Overview of Issues and Application in Credit Scoring*, Studies in Computational Intelligence (SCI) **80**, 33–63 (2008)
www.springerlink.com

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 $\mathbf{x}_i \in \mathbb{R}^n$ 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).

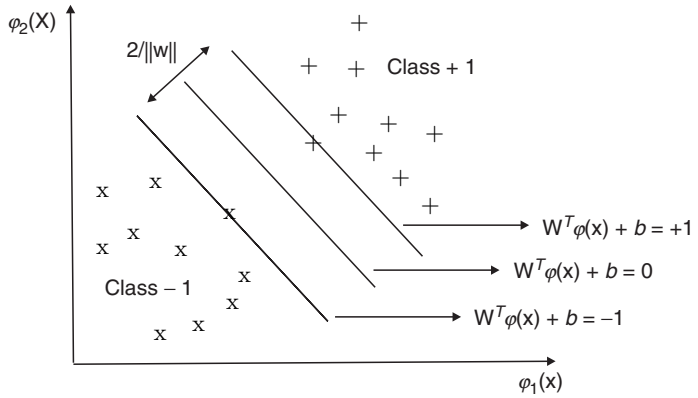

Fig. 1. Illustration of SVM optimization of the margin in the feature space

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:

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

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

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

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:

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

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:

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

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

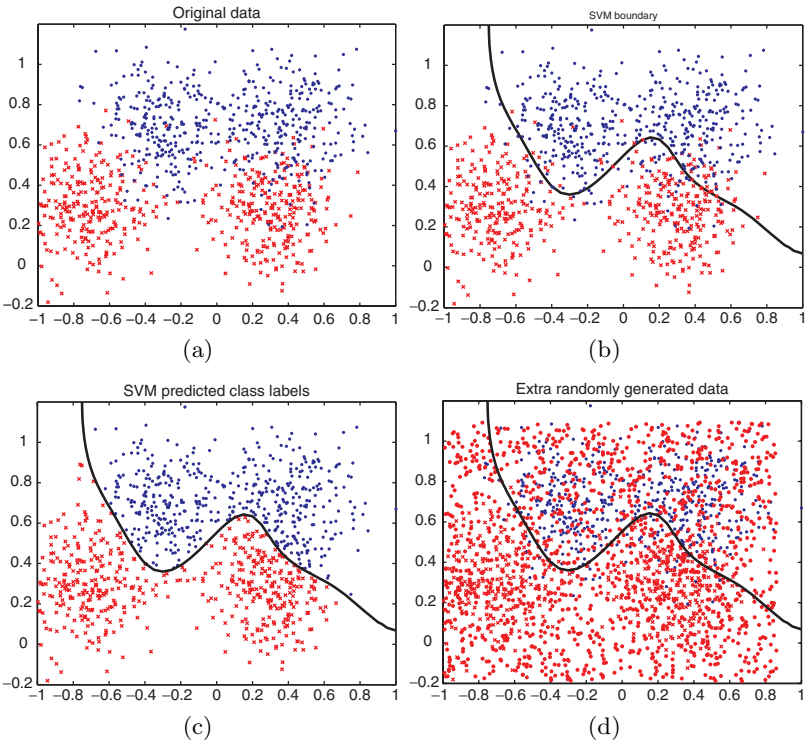

Fig. 2. (**a**) Ripley's synthetic data set, with SVM decision boundary (**b**). In (**c**) the class labels have been changed to the SVM predicted class labels, thereby removing present noise. Artificial data examples can be generated with their class labels assigned by the SVM model, as shown by the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

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

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

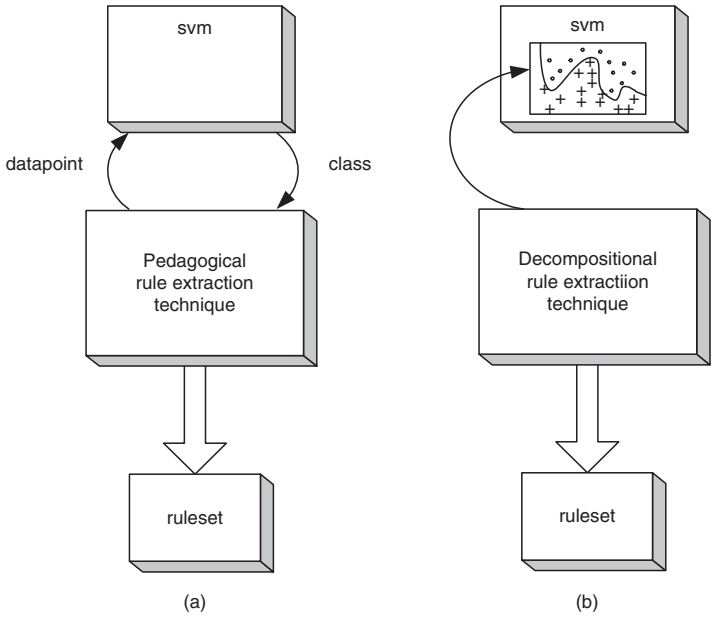

Fig. 3. Pedagogical (**a**) and decompositional (**b**) rule extraction techniques

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.

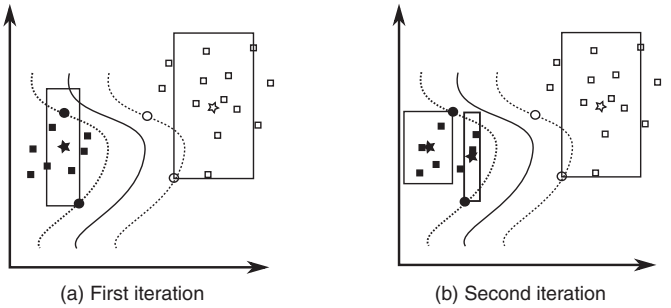

Fig. 4. Example of SVM+Prototypes algorithm

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.

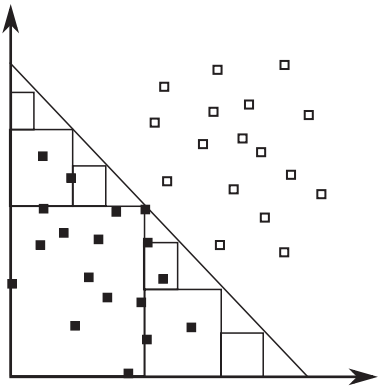

Fig. 5. Example of algorithm of Fung et al.

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. $\text{Gain}(S, x_j)$ 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 $\text{values}(x_j)$ 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:

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

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:

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

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 ≥ 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 ≥ 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 $x^i = (x_1^i, x_2^i, \ldots, x_m^i) \in X = X_1 \times X_2 \times \ldots X_m$, and a partial ordering $\leq$ defined over this input space X.

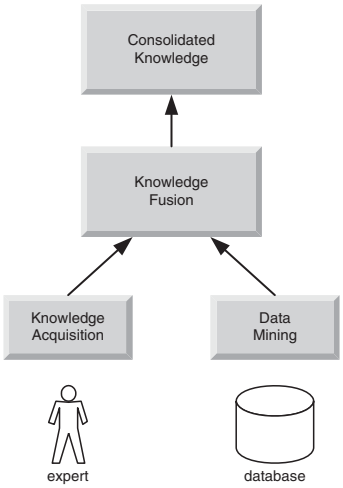

Fig. 6. The knowledge fusion process

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/`.

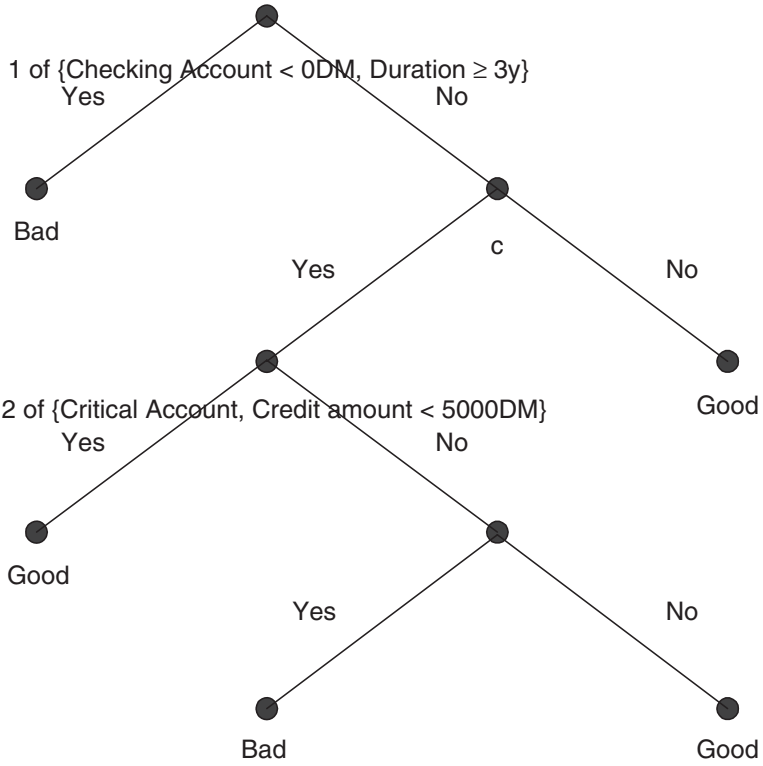

Fig. 7. Trepan tree

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

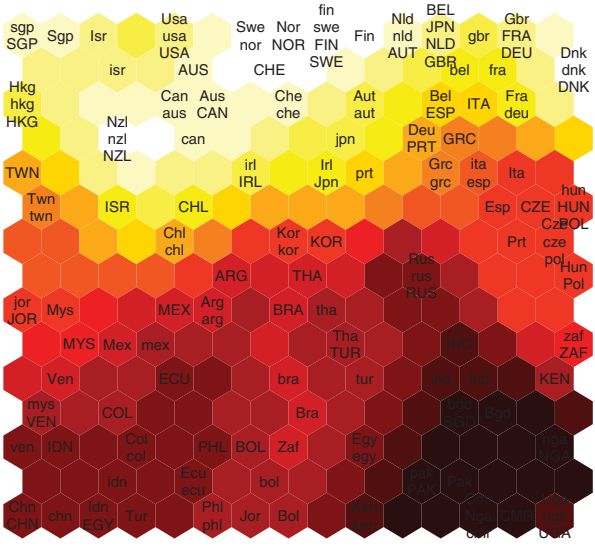

Fig. 9. Visualizing corruption index with the use of SOMs [31]

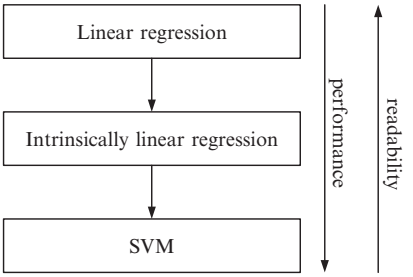

Fig. 10. From linear to non-linear models

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).

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

[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.

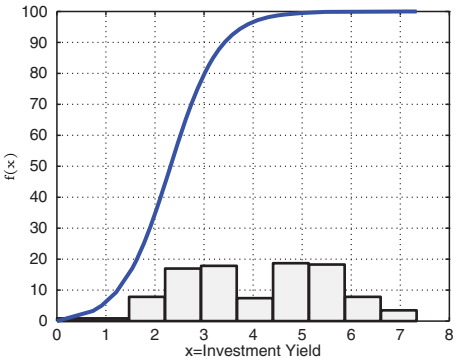

Fig. 11. Visualisation of the univariate non-linear transformations applied to the investment yield variable in the intrinsically linear model [62]

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.