Available online at www.sciencedirect.com

ScienceDirect

European Journal of Operational Research 183 (2007) 1466–1476

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

www.elsevier.com/locate/ejor

# Comprehensible credit scoring models using rule extraction from support vector machines

David Martens [a,*], Bart Baesens [b,a], Tony Van Gestel [c,d], Jan Vanthienen [a]

[a] *Department of Decision Sciences and Information Management, K.U. Leuven Naamsestraat 69, B-3000 Leuven, Belgium*
[b] *School of Management, University of Southampton, Highfield Southampton, SO17 1BJ, United Kingdom*
[c] *Basel II Modelling, Risk Management, Dexia Group Place Rogier 11, 1210 Brussels, Belgium*
[d] *Department of Electrical Engineering, ESAT-SCD-SISTA, K.U. Leuven Kasteelpark Arenberg 10, B-3001 Leuven (Heverlee), Belgium*

Received 1 October 2005; accepted 1 April 2006
Available online 29 January 2007

## Abstract

In recent years, support vector machines (SVMs) were successfully applied to a wide range of applications. However, since the classifier is described as a complex mathematical function, it is rather incomprehensible for humans. This opacity property prevents them from being used in many real-life applications where both accuracy and comprehensibility are required, such as medical diagnosis and credit risk evaluation. To overcome this limitation, rules can be extracted from the trained SVM that are interpretable by humans and keep as much of the accuracy of the SVM as possible. In this paper, we will provide an overview of the recently proposed rule extraction techniques for SVMs and introduce two others taken from the artificial neural networks domain, being Trepan and G-REX. The described techniques are compared using publicly available datasets, such as Ripley's synthetic dataset and the multi-class iris dataset. We will also look at medical diagnosis and credit scoring where comprehensibility is a key requirement and even a regulatory recommendation. Our experiments show that the SVM rule extraction techniques lose only a small percentage in performance compared to SVMs and therefore rank at the top of comprehensible classification techniques.

*Keywords:* Credit scoring; Classification; Support vector machine; Rule extraction

## 1. Introduction

Support vector machines are a state-of-the art data mining technique which have proven their performance in many applications [8], such as credit scoring [2], financial time series prediction [14], spam categorization [9] and brain tumor classification [19]. The strength of this technique lies with its ability to model non-linearities, resulting in complex mathematical models. This advantage is also its main weakness: the models may provide a high accuracy compared to other data mining techniques [2] but their comprehensibility is limited. In some domains, such as credit scoring, this lack of comprehensibility is a major drawback and causes a reluctance to use the model [10]. It goes even further:

[*] Corresponding author.
*E-mail addresses:* David.Martens@econ.kuleuven.be (D. Martens), Bart.Baesens@econ.kuleuven.be, Bart@soton.ac.uk (B. Baesens), Tony.Vangestel@dexia.com (T. Van Gestel), Jan.Vanthienen@econ.kuleuven.be (J. Vanthienen).

0377-2217/$ - see front matter © 2006 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2006.04.051

when credit has been denied to a customer, the Equal Credit Opportunity Act of the US requires that the financial institution provides specific reasons why the application was rejected; indefinite and vague reasons for denial are illegal. In the medical diagnostic field as well, clarity and explainability are key constraints. To be able to use the extra accuracy of the SVM, which can result in lives saved or money gained, as well as to obtain a usable, readable model, rules can be extracted from the complex, black-box SVM models. These rules are interpretable by humans and keep as much of the accuracy of the black box as possible.

Two approaches exist to extract rules: decompositional and pedagogical. The first approach is closely intertwined with the internal structure of the SVM, while pedagogical techniques directly extract rules which relate the inputs and outputs of the model. Although rule extraction for neural networks has been extensively researched (a.o. [1,3]), very little literature is available on SVM rule extractions. Because pedagogical techniques typically use the trained model as an oracle to label training examples, pedagogical neural network rule extraction techniques lend themselves as well to Support Vector Machines, which, unlike artificial neural networks, do not suffer from local optima in the weight space, and model selection is limited to choosing values for regularization and kernel parameters.

Other ways to simplify the complex SVM models exist, such as sensitivity analysis [17] and inverse classification [20], but do not provide the same extent of explainability as rule extraction techniques.

## 2. Support vector machines

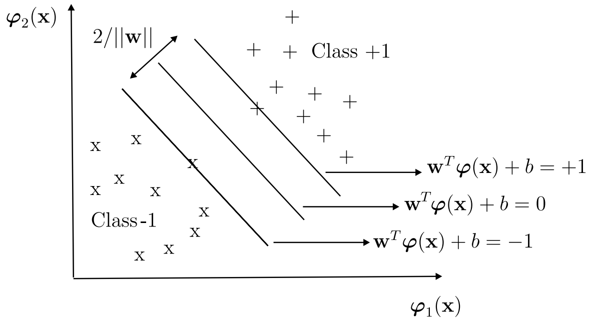
Given a training set of $N$ data points $\{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$, with input data $\mathbf{x}_i \in \mathbb{R}^n$ and corresponding binary class labels $y_i \in \{-1, +1\}$, the SVM classifier, according to Vapnik's original formulation satisfies the following conditions [8,28]:

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

which is equivalent to

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

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 $\mathbf{w}^{\mathrm{T}}\boldsymbol{\varphi}(\mathbf{x}) + b = 0$ discriminating between both classes, as visualized in Fig. 1 for a typical two-dimensional case. By minimizing $\mathbf{w}^{\mathrm{T}}\mathbf{w}$, the margin between both classes is maximized.



$\varphi_2(\mathbf{x})$
2/||w||
Class +1
Class -1
$\mathbf{w}^T\varphi(\mathbf{x}) + b = +1$
$\mathbf{w}^T\varphi(\mathbf{x}) + b = 0$
$\mathbf{w}^T\varphi(\mathbf{x}) + b = -1$
$\varphi_1(\mathbf{x})$

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

In primal weight space the classifier then takes the form

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

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

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

subject to

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

The variables $\xi_i$ 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

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

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

$$y(\mathbf{x}) = \operatorname{sign}\left[\sum_{i=1}^{N} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right], \tag{7}$$

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

$$\max_{\alpha_i} -\frac{1}{2} \sum_{i,j=1}^{N} y_i y_j K(\mathbf{x}_i, \mathbf{x}_j) \alpha_i \alpha_j + \sum_{i=1}^{N} \alpha_i \tag{8}$$

subject to

$$\begin{cases} \sum_{i=1}^{N} \alpha_i y_i = 0, \\ 0 \leqslant \alpha_i \leqslant C, \quad i = 1, \ldots, N. \end{cases} \tag{9}$$

The entire classifier construction problem now simplifies to a convex quadratic programming (QP) problem in $\alpha_i$. 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 nonlinear 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:

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

where $d, c, \sigma, \kappa$ and $\theta$ are constants.

For low-noise problems, many of the $\alpha_i$ will be typically equal to zero (sparseness property). The training observations corresponding to non-zero $\alpha_i$ are called support vectors and are located close to the decision boundary.

As Eq. (7) 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. Rule extraction techniques

Comprehensibility can be added to SVMs by extracting symbolic rules from the trained model. Rule extraction techniques attempt to open up the SVM black box and generate symbolic, comprehensible descriptions with approximately the same predictive power as the model itself. An advantage of using SVMs as a starting point for rule extraction is that the SVM considers the contribution of the inputs towards classification as a group, while decision tree algorithms like C4.5 measure the individual contribution of the inputs one at a time as the tree is grown.

Andrews et al. [1] 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.

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 will therefore typically make use of the support vectors or decision boundary. On the other hand, a pedagogical algorithm considers the trained model as a black box. Instead of looking at the internal structure, these algorithms do not make use of the support vectors or SVM decision boundary, but directly extract rules using the input–output mapping defined by the SVM model. These techniques typically use the trained SVM model as an oracle to label or classify (artificially generated) training examples which are then used by a symbolic learning algorithm. The idea behind these techniques is the assumption that the trained model can better represent the data than the original dataset. That is, the data is cleaner, free of apparent conflicts. 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, support vector machines. The difference between decompositional and pedagogical rule extraction techniques is schematically illustrated in Fig. 2.

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. The most relevant rule types are propositional rules (simple **If**...**Then**... expressions), $M$-of-$N$ rules (**If** at least $M$ of $N$ conditions $(C1, C2, \ldots, CN)$

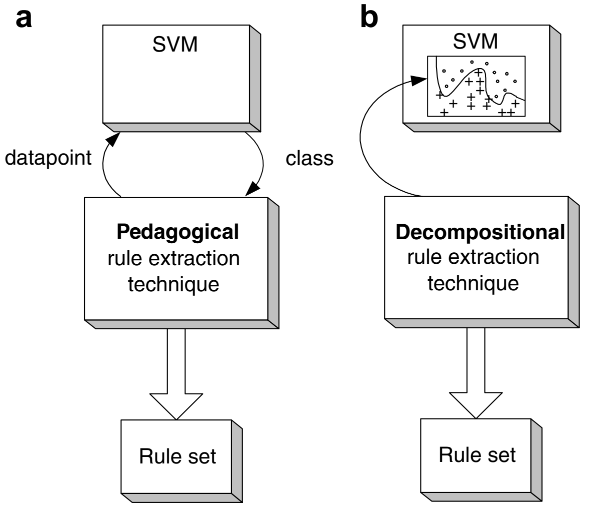


Fig. 2. Pedagogical (a) and decompositional (b) rule extraction technique.

Table 1
Characteristics of SVM rule extraction techniques

| Technique | Translucency | Rule expressiveness |
|---|---|---|
| SVM+Prototype | Decompositional | Propositional rules |
| Fung et al. | Decompositional | Propositional rules |
| C4.5 | Pedagogical | Decision tree |
| Trepan | Pedagogical | *M*-of-*N* rules |
| G-REX | Pedagogical | Propositional rules<br>Fuzzy rules |

**Then**...) and fuzzy rules that allow for more flexibility.

Table 1 provides an overview of SVM rule extraction techniques, and describes the translucency and rule expressiveness. We will evaluate the rule extraction techniques using three performance measures: accuracy, fidelity and number of extracted rules. The accuracy measures the percentage of correctly classified test points and provides a measure for the ability to make accurate predictions on previously unseen cases. The fidelity determines the percentage of test points where the classifier and the extracted rules agree on the class label.

### 3.1. Decompositional rule extraction techniques

#### 3.1.1. SVM+Prototype

A decompositional method for extracting rules from SVMs has been introduced by Nùñez et al. [21] and creates rule-defining regions based on prototype and support vectors. Prototype vectors are generated using clustering and are the representatives of the obtained clusters. Nùñez et al. use vector quantization for the clustering task. Two types of rules can be generated: equation rules and interval rules, respectively corresponding to an ellipsoid and interval region, which can be built in the following manner. Using the prototype vector as centre, an ellipsoid is built where the axes are determined by the support vector within the partition that lies the furthest from the centre. The straight line connecting these two vectors defines the long axes of the ellipsoid. Simple geometrics allow for the other axes to be determined. The interval regions are defined from ellipsoids parallel to the coordinate axes. The SVM+Prototype approach is schematically shown in Fig. 3.

An incremental approach is followed where first a single prototype and associated ellipsoid is generated. A following partition test determines whether the region is transformed into a rule (negative test) or whether new regions will be created (positive partition test). This process is continued until there are no regions with positive partition test or when a predefined number of iterations has passed. The partitioning test tries to keep the number of overlapping regions with different classes as low as possible. The partitioning test will succeed when either the generated prototype belongs to another class, when one of the vertices belong to another class, or when a support vector with different class exists within the region.

This approach may be intuitive and have good accuracy on small datasets, but it does not scale well: with a high number of patterns come just as many rules resulting in low comprehensibility. Also, the clustering will be negatively impacted by overlapping dependent variables.

#### 3.1.2. Fung et al.

Fung et al. extract non-overlapping rules by constructing hypercubes with axis-parallel surfaces [11]. This approach is similar to the one discussed previously, but requires no computationally expensive clustering. Instead, the algorithm transforms the problem to a simpler, equivalent variant and constructs the hypercubes by solving linear programs in $2n$ variables with $n$ being the input space dimension, reducing the required run time to a time order of less than a second.

Each extracted rule represents a hypercube in the $n$-dimensional space with edges parallel to the axis and is therefore of the form:

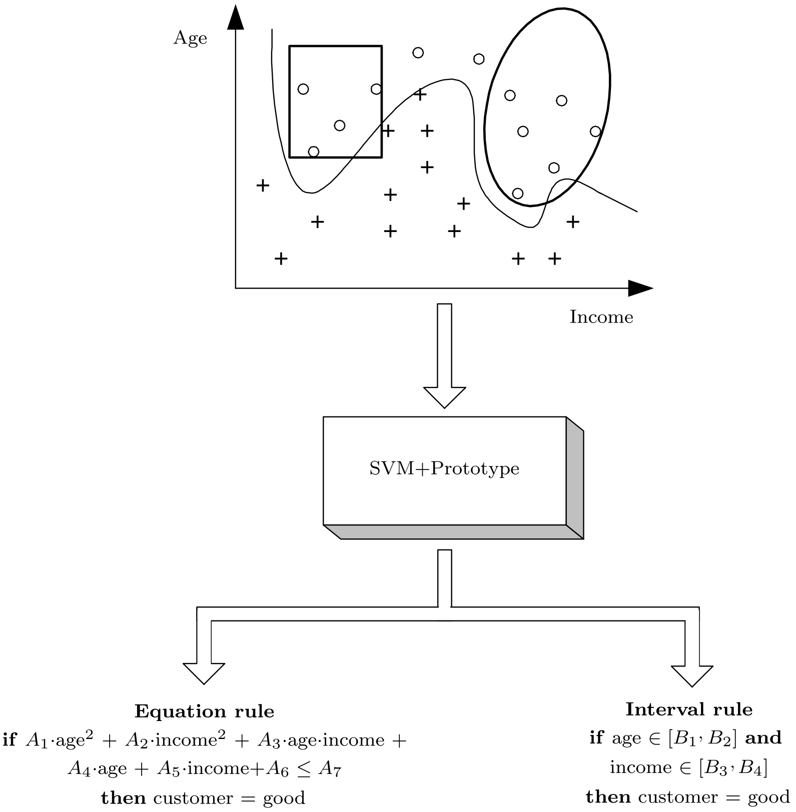

Age
Income
SVM+Prototype
Equation rule
if $A_1$·age$^2$ + $A_2$·income$^2$ + $A_3$·age·income +
$A_4$·age + $A_5$·income+$A_6$ ≤ $A_7$
then customer = good
Interval rule
if age ∈ [$B_1$, $B_2$] and
income ∈ [$B_3$, $B_4$]
then customer = good

Fig. 3. Workings of SVM+Prototype [21].

$$\textbf{if } l_1 \leqslant x_1 < u_1 \quad \textbf{and } l_2 \leqslant x_2 < u_2 \ldots \\ \textbf{and } l_n \leqslant x_n < u_n. \tag{10}$$

The problem of extracting rules of the linear SVM classification model $\mathbf{w}^{\mathrm{T}}\mathbf{x} + b = 0$ for the region $I$ underneath the separating hyperplane with class $C_-$:

$$I = \{x|\mathbf{w}^{\mathrm{T}}\mathbf{x} < b, l_i \leqslant x_i \leqslant u_i, 1 \leqslant i \leqslant n\} \tag{11}$$

can be noted as $P_-(w, b, I)$. To transform the problem to a simpler, equivalent variant, a diagonal matrix $\mathbf{T}$ and vector $\mathbf{d}$ are defined as follows:

$$T_{ii} = \frac{\operatorname{sign}(w_i)}{u_i - l_i}, \tag{12}$$

$$d_i = \begin{cases} u_i & \text{if } w_i \leqslant 0; \\ l_i & \text{if } w_i > 0. \end{cases}$$

Then the transformation $\mathbf{y} = \mathbf{T}(\mathbf{x} - \mathbf{d})$, results in the reduced rule extraction problem $P_-(\tilde{\mathbf{w}}, 1, I_0)$ for classifier $\tilde{\mathbf{w}}\mathbf{y} = 1$ and the unit hypercube as region, with

$$\tilde{\mathbf{w}} = \frac{\mathbf{w}^{\mathrm{T}}\mathbf{T}^{-1}}{b - \mathbf{w}^{\mathrm{T}}\mathbf{d}}. \tag{13}$$

Each hypercube corresponding to an extracted rule has one vertex that lies on the hyperplane, which simplifies the problem and allows generating disjoint rules, as shown by Fig. 4. Given a region $I$, the optimal rule can be defined in different ways. The first one is by maximizing the volume of the axis-parallel hypercube. The second one is by maximizing the point coverage. Using the volume maximization criteria rules are generated corresponding

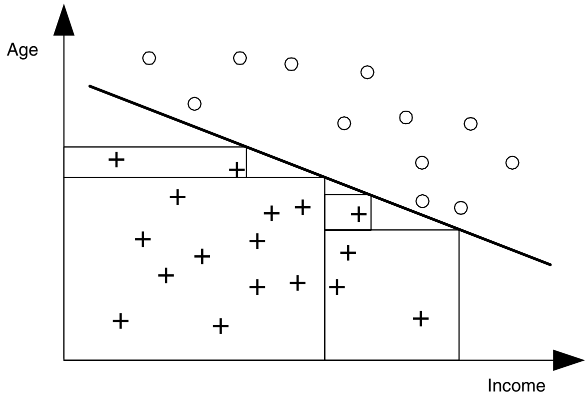


Fig. 4. Workings of Fung et al. [11].

to hypercubes with maximal volume. For each rule, the optimization problem is the maximization of the log volume.

$$\max_{x\in\mathbb{R}^n} \log\left(\prod_{i=1}^{n} x_i\right) \quad \text{s.t.} \sum_{i=1}^{n} w_i x_i = b, \quad 0 \leqslant x \leqslant 1. \tag{14}$$

For the point coverage criteria, the cardinality $|C|$ is maximized by Eq. (15), with $x^*$ the point on the decision boundary.

$$C = (A_- \cap \{x|\mathbf{w}^{\mathrm{T}}x < 1\}) \cap \{x|0 \leqslant x \leqslant x^*\}. \tag{15}$$

A drawback of this algorithm is that it can only be applied to SVMs with linear kernel, as the decision boundary must be linear.

### 3.2. Pedagogical rule extraction techniques

#### 3.2.1. C4.5

A first pedagogical rule extraction technique is based on the popular C4.5 algorithm [25]. 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:

$$\text{Entropy}(S) = -p_1\log_2(p_1) - p_0\log_2(p_0), \tag{16}$$

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 $x_j$:

$$\text{Gain}(S,x_j) = \text{Entropy}(S) - \sum_{v\in\text{values}(x_j)} \frac{|S_v|}{|S|}\text{Entropy}(S_v), \tag{17}$$

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 [24]. 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:

$$\text{Gainratio}(S, x_j) = \frac{\text{Gain}(S, x_j)}{\text{SplitInformation}(S, x_j)} \text{ with}$$
$$\text{SplitInformation}(S, x_j) = -\sum_{k\in\text{values}(x_j)} \frac{|S_k|}{|S|}\log_2\frac{|S_k|}{|S|}. \tag{18}$$

The tree induction algorithm 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 [4] to extract rules from SVMs. A problem that arises however is that the deeper a tree is expanded, the less data points are available to use to decide upon the splits. The next technique we will discuss tries to overcome this issue.

#### 3.2.2. Trepan

Trepan was first introduced in [6,7]. It is originally conceived as a pedagogical tree extraction algorithm extracting decision trees from trained neural networks with arbitrary architecture. Trepan grows a tree by recursive partitioning, using a best-first expansion strategy. Trepan allows splits with *at least M-of-N* type of tests. At each step, a queue of leaves is further expanded into sub-trees until a stopping criterion is met. In order to mimic the behavior of the generated black-box model Trepan first relabels the training observations according to the classifications made by the model. The relabelled training dataset is then used to initiate the tree growing process.

To deal with the problem of having fewer and fewer training observations available for deciding upon the splits or leaf node class labels at lower levels of the tree, Trepan can enrich the training data with additional training instances which are then also labelled (classified) by the model itself. The black box model (be it a neural network, a support vector machine or any other classification model) is thus used as an oracle to answer class membership queries about artificially generated data points. This way, it can be assured that each node split or leaf node class decision is based upon at least $S_{\min}$ data points where $S_{\min}$ is a user defined parameter. In other words, if a node has only $m$ training data points available and $m < S_{\min}$, then $S_{\min} - m$ data points are additionally generated and labelled by the network. This process is often referred to as *active learning*.

These extra data points are generated taking into account the distribution of the data and the constraints from the root of the tree to the node under consideration. More specifically, at each node of the tree, Trepan estimates the marginal distribution of each input. For a discrete valued input, Trepan simply uses the empirical frequencies of the various values whereas for a continuous input $x$, a kernel density estimation method is used to model the probability distribution $f(x)$ as follows [27]:

$$f(x) = \frac{1}{m}\sum_{j}^{m}\left[\frac{1}{\sqrt{2\pi}\sigma}\exp^{-\left(\frac{x-\mu_j}{2\sigma}\right)^2}\right], \qquad (19)$$

whereby $m$ is the number of training examples used in the estimate, $\mu_j$ is the value of the input for the $j$th example, and $\sigma$ is the width of the Gaussian kernel. Trepan sets $\sigma$ to $\frac{1}{\sqrt{m}}$.

Trepan has been mainly used to generate rules from neural networks. In this paper we propose to use an SVM model as an oracle to label data points. A MATLAB toolbox to generate rules using any black box model as oracle has been implemented [5] and made publicly available.

### *3.2.3. G-REX*

A technique recently suggested, named G-REX (Genetic Rule EXtraction) [16], is a pedagogical method to extract rules from artificial neural networks with the use of genetic programming [18], which is based on Darwin's principle of 'survival of the fittest'. A pool of candidate rules (which can be Boolean rules, decision trees, *m*-of-*n* rules or even fuzzy rules) is continuously evaluated against an evaluation/fitness function, which incorporates all requirements on comprehensibility, fidelity and accuracy. The best rules are kept and combined using genetic operators to raise the fitness over time. The selection operator chooses an individual that is allowed to reproduce with a probability that is proportional to its fitness; this operator is known as roulette wheel selection. The reproduction phase encompasses crossover and mutation. After a number of generations the most fit program, according to the defined fitness function, is chosen as the extracted rule.

An example genetic program is provided in Fig. 5, which describes a rule for a good customer. For multi-class problems, the class variable is also included in the genetic program.

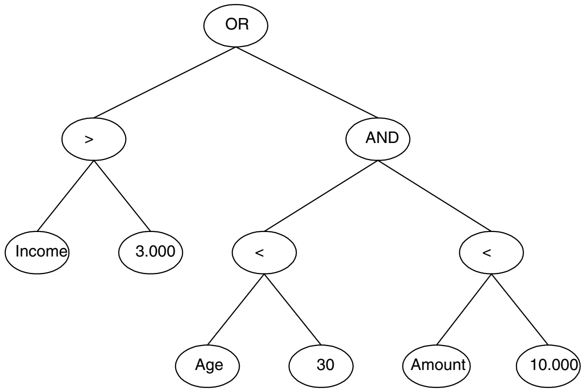


Fig. 5. Example genetic program.

As for C4.5 and Trepan, the trained black-box model is mimiced by relabelling training data according to its predictions. An alike extension from artificial neural networks to support vector machines presents itself.

## 4. Experiments

### *4.1. Experimental setup*

To evaluate and compare the rule extraction techniques described previously, we applied them to a number of datasets. Tests were done on Ripley's synthetic dataset [26] which has two variables and thus allows for visualization of the model and extracted rules. We also tested on the commonly used iris dataset, the breast cancer and australian credit scoring dataset from the UCI data repository [15], and a real-life bankruptcy dataset, all from domains where comprehensibility is a major requirement. We also included C4.5 (on the actual data) and logistic regression (logit) to benchmark the resulting rules with traditionally used classification techniques.

To get a fair view of the performances, we conducted 20 runs for each dataset, using the following setup each time. First we randomly shuffled the data, and a training and test set was chosen in a 2–1 ratio. Next, the SVM model with RBF kernel was trained,[1] where grid-search was used to determine the $\sigma$ and $\gamma$ hyperparameters. Rules were extracted with Trepan, which uses the actual training data and the trained SVM model as an oracle. C4.5 was trained on the modified training set, that is the training set with class labels changed to the

[1] We opted for radial basis function (RBF) kernels because of their superiority, which is experimentally demonstrated in [13].

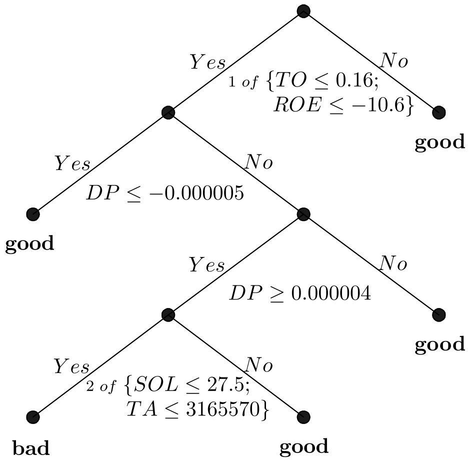
SVM predicted labels. Similarly G-REX was run on the modified dataset. The actual and modified test sets were then used to determine respectively the accuracy and fidelity of the generated rules.

### 4.2. Credit scoring

Two credit scoring datasets are included in our experiments. The first one is the Australian credit approval dataset, which concerns credit card applications and is retrieved from [15]. For confidentiality reasons, all attribute names and values have been changed to meaningless symbols. The second credit scoring dataset consists of bankruptcy data of firms with middle-market capitalization (mid-cap firms) in the Benelux countries (Belgium, The Netherlands, Luxembourg) [12], and is obtained from a major Benelux financial institution. Firms in the mid-cap segment are defined as follows: they are not stock-listed, the book value of their total assets exceeds 10 million euro, and they generate a turnover that is smaller than 0.25 billion euro. A Trepan tree for this Bene-C dataset with accuracy of 87.9% and fidelity of 90.5% is shown in Fig. 6.

### 4.3. Ripley

Ripley's dataset has two variables and two classes, where the classes are drawn from two normal distributions with a high degree of overlap. Since as many datapoints can be taken as wanted, we deviated from our 2–1 ratio for training and test set, and used a training set of size 250 and a test set of 1000 data points.



| | |
|---|---|
| DP | Fin. Debt Pay. after 1 year |
| TO | Turnover |
| TA | Total Assets |
| ROE | Return on Equity |
| SOL | Solvency Ratio |

Fig. 6. Trepan tree scoring Belgian and Dutch Corporations.

Fig. 7 shows the prediction values of both the SVM (accuracy 91.4%) and logit (accuracy 88.6%) classifiers, together with the generated Trepan tree (accuracy 90.2% and fidelity 97.6%). Note that the splits in the Trepan tree are all of the form 1 *of condition* and are simply shown as *condition.*

### 4.4. Iris

The iris dataset is a commonly used dataset in the pattern recognition literature and contains three classes of 50 instances each, where each class refers to a type of iris plant.

The following rule, with an accuracy of 96.0% and fidelity of 94.0%, was extracted by G-REX:

```
if (petal width ⩽1.6) then
    if (petal length ⩽2.6) then Setosa
    else Versicolour
else Virginica
```

A generated Trepan tree with an accuracy of 98%, A4 being the petal width is shown in Fig. 8.

### 4.5. Medical diagnostic

For the Wisconsin Diagnostic Breast Cancer dataset, the task consists of classifying breast masses as being either benign or malignant. For this, nine attributes of a sample are listed that are deemed relevant. Our experiments show a very good performance achieved by SVM (average accuracy of 96.3%); but their lack of clarity makes them useless for doctors who need to make the diagnosis. The extracted rules on the other hand, provide very comprehensible guidelines while keeping a high performance.

For the technique proposed by Fung et al., the generated rule [11] is:

**if** `(Cell Size⩽3) & (Bare Nuclei⩽1) & (Normal Nucleoli⩽7)` **then** `benign`

and has an accuracy of 95.2%. SVM+Prototype has also been applied to the Wisconsin Diagnostic Breast Cancer datasets [22]. The equation rules have an accuracy and fidelity of respectively 96.6% and 98.5%.

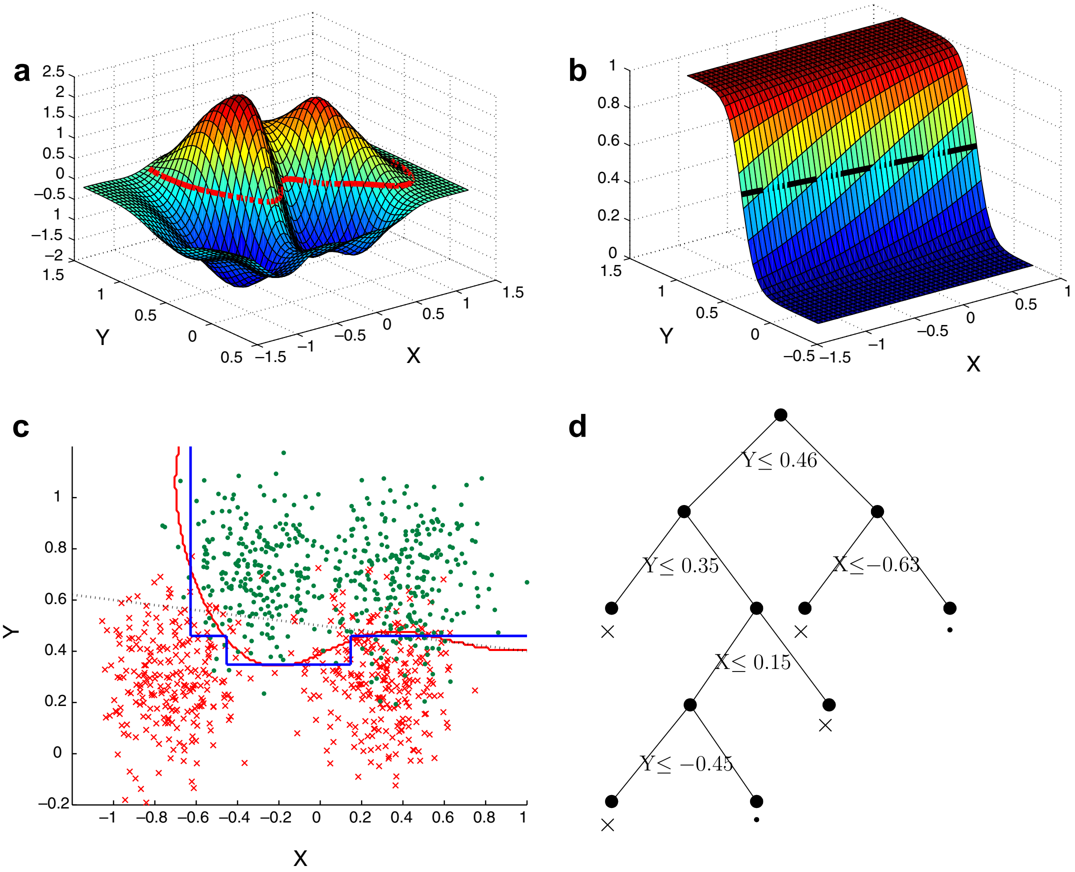


Fig. 7. (a) SVM and (b) logit prediction values on Ripley's dataset. Setting the cut-off at respectively 0 and 0.5 results in the two-dimensional (c) SVM (—) and logit (·) classifiers. Also shown are the Trepan rules (—), with (d) the accompanying Trepan tree.

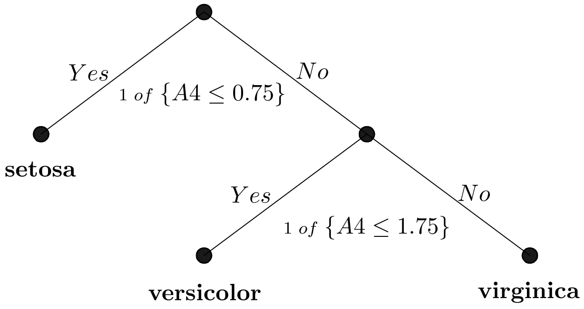


Fig. 8. Trepan tree classifying iris plants.

Table 2
Number of instances, continuous and categorical variables for the included datasets

| | Instances | Nb continuous variables | Nb categorical variables |
|---|---|---|---|
| Ripley | 1250 | 2 | 0 |
| Iris | 150 | 4 | 0 |
| BCW | 699 | 0 | 9 |
| Austr | 690 | 6 | 8 |
| Bene-C | 422 | 15 | 0 |

### 4.6. Results

In Table 2, the dataset properties are provided, listing the number of instances, and the number of continuous and categorical variables. Table 3 summarizes the results of our experiments. For each dataset, the accuracy, fidelity (if applicable) and number of generated rules are displayed. The best performances are in boldface, the ones with no significant difference at the 5% level from the top with respect to a paired $t$-test are in italic, and the others in normal script. Furthermore, to easily see the rule extraction technique with the highest accuracy for each dataset, we additionally underlined this performance measure. Note that the performance measures for SVM+Prototype and the technique by Fung et al. come from published papers and not

Table 3
Average out-of-sample performance for rule extractions from SVMs

| Technique | Ripley | | | Iris | | | BCW | | | Austr | | | Bene-C | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Acc | Fid | #R | Acc | Fid | #R | Acc | Fid | #R | Acc | Fid | #R | Acc | Fid | #R |
| logit | 88.0 | | | *96.4* | | | *96.1* | | | *85.7* | | | 87.0 | | |
| C4.5 | 88.0 | | 5.2 | 94.5 | | 3.4 | 94.6 | | 9 | 84.2 | | 5.6 | 80.2 | | |
| SVM | **90.3** | | | **97.0** | | | **96.3** | | | **85.7** | | | **96.5** | | |
| Trepan | 89.5 | **97.5** | 7.3 | 96.2 | *97.0* | 6.7 | 95.0 | **97.2** | 5.4 | 85.1 | **99.0** | 2.6 | 82.0 | *84.3* | 6.1 |
| C4.5 | 89.1 | 96.5 | 5.7 | 95.1 | *96.8* | 4.3 | 94.4 | *96.4* | 5.2 | 85.1 | *98.8* | 2.9 | 80.2 | *84.4* | 16.6 |
| G-REX | 89.0 | 95.8 | 2.4 | 94.8 | **97.2** | 4.0 | 95.1 | *97.0* | 2.2 | 71.5 | 71.5 | 4.1 | 83.6 | **85.1** | 4.0 |
| SVM+Pr | | | | 96.0 | 98.0 | 7 | 96.3 | 98.2 | 5.1 | | | | | | |
| Fung et al. | | | | | | | 95.2 | | 2 | | | | | | |

our own experiments. Therefore we only list them and do not use them in our comparison.

Trepan obtained the best average performance in our experiments. It consistently performed better than C4.5 with comparable comprehensibility but was more computationally demanding to reach these results. Since G-REX allows for the comprehensibility requirements to be included in the fitness function, it was overall able to extract very compact rules with no significant performance degradation. It can be observed that the SVM classifiers performed best on all datasets. The rules extracted from these SVM models have an accuracy that is comparable or better than the included traditional classification techniques with a comprehensibility that even surpasses them.

Of the included pedagogical rule extraction techniques, only Trepan goes beyond the relabelling of the training data and actually adds extra artificial training points, which might explain the better performance. None of the rule extraction techniques takes into account any indication of confidence, such as distance from the margin. Including such a confidence measure might increase the performance and is an interesting focus of future research.

## 5. Conclusion

Rule extraction techniques generate classification models that have clear advantages. First of all, they are comprehensible and therefore easy to incorporate in real-life applications where clarity of the classifications made is needed. Secondly, the extracted rules only lose a small percentage in accuracy of the black box model from which they are generated. Since support vector machines are among the best performing classifiers, rules extracted from SVMs achieve an accuracy that often surpasses that of the classical methods, such as C4.5 and logit. Using the SVM model instead of the original data points eliminates the apparent conflicts and creates a cleaner dataset. In our experiments, the rules generated by C4.5 on the data with labels predicted by the SVM even outperform the C4.5 rules that result from the dataset with the actual class labels. These advantages make it appropriate to consider SVMs and their extracted rules for applications where both accuracy and comprehensibility are required. One no longer needs to settle for the traditional comprehensible, yet less accurate classification methods.

SVMs are known to perform well on high-dimensional datasets with few datasets (see a.o. [23]). The credit scoring data of our experiments come from the retail and corporate area for which a rather sufficient number of observations are available in relation to the number of inputs. Hence, the credit scoring problems considered could not be coined as high-dimensional problems. It would be interesting to investigate the performance of SVM rule extraction in high-dimensional credit scoring context, e.g. in the context of low default portfolios.

## Acknowledgements

We would like to thank the Flemish Research Council (FWO, Grant G.0615.05) for financial support, and Ulf Johansson, Rikard Knig, Lars Niklasson and Martyn Ford for making available their software.

## References

[1] R. Andrews, J. Diederich, A.B. Tickle, Survey and critique of techniques for extracting rules from trained artificial neural networks, Knowledge-Based Systems 8 (6) (1995) 373–389.

[2] B. Baesens, T. Van Gestel, S. Viaene, M. Stepanova, J. Suykens, J. Vanthienen, Benchmarking state-of-the-art

classification algorithms for credit scoring, Journal of the Operational Research Society 54 (6) (2003) 627–635.

[3] B. Baesens, R. Setiono, C. Mues, J. Vanthienen, Using neural network rule extraction and decision tables for credit-risk evaluation, Management Science 49 (3) (2003) 312–329.

[4] N. Barakat, 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.

[5] A. Browne, B. Hudson, D. Whitley, P. Picton, Biological data mining with neural networks: Implementation and application of a flexible decision tree extraction algorithm to genomic problem domains, Neurocomputing 57 (2004) 275–293.

[6] M.W. Craven. Extracting comprehensible models from trained neural networks. Ph.D. thesis, University of Winsconsin-Madison, 1996. Supervisor-J.W. Shavlik.

[7] M.W. Craven, J.W. Shavlik, Extracting tree-structured representations of trained neural networks, Advances in Neural Information Processing Systems 8 (1996) 24–30.

[8] N. Cristianini, J. Shawe-Taylor, An introduction to Support Vector Machines and Other Kernel-Based Learning Methods, Cambridge University Press, New York, NY, USA, 2000.

[9] H. Drucker, D. Wu, V. Vapnik, Support vector machines for spam categorization, IEEE-NN 10 (5) (1999) 1048–1054.

[10] D.W. Dwyer, A.E. Kocagil, R.M. Stein, Moody's kmv riskcalc v3.1 model, 2004.

[11] G. Fung, S. Sandilya, R. Bharat Rao, Rule extraction from linear support vector machines, in: KDD '05: Proceeding of the Eleventh ACM SIGKDD International Conference on Knowledge Discovery in Data Mining, ACM Press, New York, NY, USA, 2005, pp. 32–40.

[12] T. Van Gestel, B. Baesens, J. Suykens, D. Van den Poel, D.-E. Baestaens, M. Willekens, Bayesian kernel based classification for financial distress detection, European Journal of Operational Research 172 (3) (2006) 979–1003.

[13] T. Van Gestel, J.A.K. Suykens, B. Baesens, S. Viaene, J. Vanthienenand G. Dedene, B. De Moor, J. Vandewalle. Benchmarking least squares support vector machine classifiers. CTEO, Technical Report 0037, K.U. Leuven, Belgium, 2000.

[14] T. Van Gestel, J.A.K. Suykens, D.-E. Baestaens, A. Lambrechts, G. Lanckriet, B. Vandaele, B. De Moor, J. Vandewalle, Financial time series prediction using least squares support vector machines with the evidence framework, IEEE Transactions on Neural Networks 12 (4) (2001) 809–821.

[15] S. Hettich, S.D. Bay. The uci kdd archive, 1996. <http://kdd.ics.uci.edu>.

[16] U. Johansson, R. König, 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.

[17] J.T.Yao. Sensitivity analysis for data mining. In: 22nd International Conference of NAFIPS Proceedings, 2003, pp. 272–277.

[18] John R. Koza, Genetic Programming: On the Programming of Computers by Natural Selection, MIT Press, Cambridge, Mass, 1992.

[19] C. Lu, T. Van Gestel, J.A.K. Suykens, S. Van Huffel, I. Vergote, D. Timmerman, Preoperative prediction of malignancy of ovarium tumor using least squares support vector machines, Artificial Intelligence in Medicine 28 (3) (1999) 281–306.

[20] M.V. Mannino, M.V. Koushik, The cost-minimizing inverse classification problem: A genetic algorithm approach, Decision Support Systems 29 (3) (2000) 283–300.

[21] H. Nùnez, C. Angulo, A. Catala, Rule extraction from support vector machines. In: European Symposium on Artificial Neural Networks Proceedings, 2002, pp. 107–112.

[22] H. Nùnez, C. Angulo, A. Catala. Rule based learning systems from SVM and RBFNN. Tendencias de la mineria de datos en espana, Red Espaola de Minera de Datos, 2004.

[23] N. Pochet, F. De Smet, J.A.K. Suykens, B.L.R. De Moor, Systematic benchmarking of microarray data classification: Assessing the role of non-linearity and dimensionality reduction, Bioinformatics 20 (17) (2004) 3185–3195.

[24] J.R. Quinlan, Induction of decision trees, Machine Learning 1 (1) (1986) 81–106.

[25] J.R. Quinlan, C4.5 Programs for Machine Learning, Morgan Kaufman Publishers Inc., San Francisco, CA, USA, 1993.

[26] B.D. Ripley, Neural networks and related methods for classification, Journal of the Royal Statistical Society B 56 (1994) 409–456.

[27] D.W. Silverman, Density Estimation for Statistics and Data Analysis, Chapman and Hall, 1986.

[28] V.N. Vapnik, The Nature of Statistical Learning Theory, Springer-Verlag, New York, Inc., New York, NY, USA, 1995.