Neurocomputing 74 (2010) 178–190

Contents lists available at ScienceDirect

Neurocomputing

journal homepage: www.elsevier.com/locate/neucom

Rule extraction from support vector machines: A review

Nahla Barakat [a,*], Andrew P. Bradley [b]

[a] Department of Applied Information Technology, German University of Technology in Oman, Oman
[b] School of Information Technology and Electrical Engineering (ITEE), The University of Queensland, St. Lucia, QLD 4072, Australia

ARTICLE INFO

Article history:
Received 20 March 2009
Received in revised form
22 January 2010
Accepted 22 February 2010
Communicated by S. Mitra
Available online 27 March 2010

Keywords:
Machine learning
Data mining
Knowledge discovery
Information extraction
Pattern recognition applications
SVMs

ABSTRACT

Over the last decade, support vector machine classifiers (SVMs) have demonstrated superior generalization performance to many other classification techniques in a variety of application areas. However, SVMs have an inability to provide an explanation, or comprehensible justification, for the solutions they reach. It has been shown that the '*black-box*' nature of techniques like artificial neural networks (ANNs) is one of the main obstacles impeding their practical application. Therefore, techniques for rule extraction from ANNs, and recently from SVMs, were introduced to ameliorate this problem and aid in the explanation of their classification decisions. In this paper, we conduct a formal review of the area of rule extraction from SVMs. The review provides a historical perspective for this area of research and conceptually groups and analyzes the various techniques. In particular, we propose two alternative groupings; the first is based on the SVM (model) components utilized for rule extraction, while the second is based on the rule extraction approach. The aim is to provide a better understanding of the topic in addition to summarizing the main features of individual algorithms. The analysis is then followed by a comparative evaluation of the algorithms' salient features and relative performance as measured by a number of metrics. It is concluded that there is no one algorithm that can be favored in general. However, methods that are kernel independent, produce the most comprehensible rule set and have the highest fidelity to the SVM should be preferred. In addition, a specific method can be preferred if the context of the requirements of a specific application, so that appropriate tradeoffs may be made. The paper concludes by highlighting potential research directions such as the need for rule extraction methods in the case of SVM incremental and active learning and other application domains, where special types of SVMs are utilized.

1. Introduction

Over the last three decades, data mining and machine learning techniques have been remarkably successful in extracting interesting knowledge and hidden *patterns* from the ever growing databases. In this context, Frawley et al. [1] provides the following definition for the term 'pattern':

> Given a set of facts (data) F, a Language L, and some measure of certainty C, a pattern S is a statement S in L that describes relationships among a subset Fs of F with certainty C, such that S is simpler (in some sense) than the enumeration of all facts in Fs. [1]

Support vector machines and ANNs are among the most successful machine learning techniques applied in this area [2–10]. However, the superior performance of SVMs and ANNs comes with a significant drawback: the language, L, in which they learn patterns from data is not comprehensible to humans; hence they do not provide an explanation or a comprehensible justification for the knowledge they learn. This has been shown to be one of the main obstacles impeding their practical application [11–13].

1.1. Rule extraction from SVMs: the motivation

Explanation has been one of the most important topics that has attracted researchers' attention, not only in philosophy, where several theories of explanation have been proposed [14], but also in the area of artificial intelligence. In particular, since the early introduction of expert systems, and continuing with case-based reasoning, explanation is still considered one of the most important criteria that influence the acceptance of these techniques by end users [15–18]. Similarly, it has also been shown that the explanation of a classification decision is a crucial requirement for the acceptance of *black-box* models by end users, especially in areas like medical diagnosis and prognosis [11–13,19–22]. Therefore, several methods have been introduced

* Corresponding author.
E-mail addresses: n.barakat@uq.edu.au, nahlabarakat@gmail.com (N. Barakat), bradley@itee.uq.edu.au (A.P. Bradley).

0925-2312/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.neucom.2010.02.016

for rule extraction from ANNs [23], and more recently from SVMs [22,24–37].

One could argue that rule extraction from SVMs is a narrow area of research, which is not expected to grow in the future, given that to date rule extraction from ANNs has not been particularly successful. However, over the past 6 years, this topic has attracted an ever increasing number of researchers from a variety of domains [13,21,22,36–39]. Furthermore, there is a significantly growing trend in the usage of SVMs and continuing development of new types of kernels e.g., [40–44]. Explanation can also help in tasks like image annotation and document mapping, where SVMs have been widely used in building and mapping ontologies for the semantic web [45–47]. Discovering concept drift in active learning by SVMs [48] is also a significant domain which can benefit from the comprehensibility provided by rule extraction algorithms.

1.2. Rule extraction from SVMs: the problem

The task of rule extraction from SVMs is to devise rules from the model (SVM) rather than directly from the data. Therefore, an explanation of the patterns learned and embedded in their structure (model support vectors (SVs) and their associated parameters) is revealed and provided to the end users in a comprehensible form.

1.3. Overview

In this paper we conduct the first formal review for the area of rule extraction from SVMs. The survey conceptually groups and analyzes the literature at a higher level of abstraction, which outlines differing fundamental approaches to the topic, rather than summarizing the main features of individual algorithms, as in [37,49]. This is then followed by a comparative evaluation of the methods salient features and a benchmark of results published to date.

The paper is organized as follows: Section 2 provides a brief introduction to SVMs followed by the review of rule extraction algorithms in Section 3. Section 4 briefly summarizes the classification of rule extraction from ANNs, while similarities between ANNs and SVMs are highlighted in Section 5. In Section 6, we discuss and compare the main features of different algorithm. Some potential directions for future research are also highlighted in Section 7, followed by a summary and conclusions in Section 8. Abstract representations for individual algorithms' main modules are also provided in Appendix A.

2. Support vector machine classifiers: an overview

Support vector machines belong to the maximum margin classifier family and are based on the principle of structural risk minimization (SRM). The SRM principle aims to select a hypothesis function with low capacity from a nested sequence of functions that simultaneously minimizes both the true error rate (classification error on unseen examples), and empirical (training set) error rate [50].

Given the training data set $\{x_i,y_i\}$, $i=1,\ldots,l$, $y_i\in\{-1,1\}$, $x_i\in\Re^n$ (where x_i is an input feature vector of dimensionality n, and y_i the corresponding class label), an SVM finds the optimal separating hyper-plane with the largest margin [50]. Eqs. (1) and (2) represent the separating hyper-planes in the case of separable data sets:

$$x_i w+b\geq +1 \quad \text{for } y_i=+1 \tag{1}$$

$$x_i w+b\leq -1 \quad \text{for } y_i=-1 \tag{2}$$

For the linearly separable case, finding a maximum separating margin $d(d=2/(\|w\|))$ is a constrained optimization problem represented by

$$\text{minimize } \tfrac{1}{2}\|w\|^2 \quad \text{subject to } y_i(wx_i+b)\geq 1 \tag{3}$$

2.1. Soft margin linear SVMs

The soft margin hyper-plane can be obtained by relaxing the optimization problem in (3), through the introduction of ξ_i (positive slack variable) [51] to allow for errors on the training set. Including ξ_i, Eqs. (1) and (2) are modified as follows:

$$x_i w+b\geq +1-\xi_i \quad \text{for } y_i=+1 \tag{4}$$

$$x_i w+b\leq -1+\xi_i \quad \text{for } y_i=-1, \quad \xi_i\geq 0 \quad \forall_i \tag{5}$$

Introducing the regularization parameter C controls the tradeoff between the margin maximization and the training error [51] the optimization problem in (3) is then modified as follows:

$$\text{minimize } \frac{1}{2}\|w\|^2+C\sum_{i=1}^{l}\xi_i \tag{6}$$

$$\text{Subject to } y_i(wx_i+b)\geq 1-\xi_i, \quad \xi_i\geq 0$$

where $\sum_{i=1}^{l}\xi_i$ the upper bound on the training error and C is a regularization parameter [51]. The optimization problem in (6) is called the primal problem. To find an easier solution to this problem it is transformed to its dual [51], by introducing α, the Lagrange multipliers (dual variables) which are the fundamental unknowns in the dual optimization problem [51]. Hence the problem in (6) becomes

$$\text{maximize } w(\alpha)=\sum_{i=1}^{l}\alpha_i-\frac{1}{2}\sum_{i,j=1}^{l}\alpha_i y_i\alpha_j y_j\langle x_i,x_j\rangle$$

$$\text{such that } C\geq\alpha_i\geq 0 \quad \forall_i, \quad \sum_{i=1}^{l}\alpha_i y_i=0 \tag{7}$$

Solving for α, training examples with non-zero α are called support vectors (SVs) and the hyper-plane is being completely defined by the SVs only. As shown in (7), the C parameter constitutes the upper bound on α_i. As a result, three types of support vectors are characterized based on the values of α_i *and* ξ_i as follows:

- Support vectors with $\alpha_i<C$, those are the SVs which lie outside the margin d and will be correctly classified.
- Support vectors with $\alpha_i=C$, and $\xi_i>1$, these SVs lies at the wrong side of the hyper-plane, and represent errors (will be misclassified points by the hyper-plane).
- Support vectors with $\alpha_i=C$, and $0<\xi_i\leq 1$, these SVs will lie inside the margin d (i.e., closer than $1/(\|w\|)$ from the hyper-plane).

2.2. Non-linear SVMs

In the case of non-linear models, SVMs use kernel functions[1] to map non-linearly separable data in the input space $x_i\in\Re^n$ to be

[1] A function that enables direct computation of the inner product $\langle\Phi(x_i)\Phi(x)\rangle$ in feature space as a function of original input points, where Φ is a non-linear map from input space to a feature space (inner product) [51].

linearly separable in a higher dimensional feature space ($\Re^n \rightarrow \Re^F$, where $F \gg n$). In this way, a linear classification problem can be solved in the feature space [52].

Including Kernel functions, the optimization problem now becomes

$$\text{maximize } w(\alpha) = \sum_{i=1}^{l} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{l} \alpha_i y_i \alpha_j y_j K(x_i, x_j),$$
$$C \geq \alpha_i \geq 0 \quad \forall_i, \quad \sum_{i=1}^{l} \alpha_i y_i = 0 \tag{8}$$

The decision function of the SVM is given by

$$f(x) = sign\left(\sum_{s=1}^{sv} \alpha_s y_s K(x_s, x) + b\right) \tag{9}$$

where sv are the model SVs.

3. Rule extraction from support vector machines

The steps of all the algorithms reviewed in this section start with training an SVM with labelled examples to obtain an SVM model from which rules are to be extracted. The rule extraction process involves rule generation utilizing different SVM components. Some methods also use the training data together with other machine learning techniques, such as clustering algorithms or decision tree learners. The performance of the extracted rules is then evaluated in terms of accuracy, comprehensibility, area under the receiver operating characteristic curve (AUC) and accuracy with respect of the SVM from which they were extracted (often referred to as fidelity).

Fig. 1 provides an abstract view of the different rule extraction phases and groups of the reviewed rule extraction algorithms.

3.1. Support vector machines training

Learning a good SVM (model) from training data requires careful selection and optimization of the SVM training parameters. Even though the selection of optimal SVM kernel or other training parameters is application dependent, some of the studies presented in this paper are limited to specific type of kernels. For example, the methods described in Fung et al. [13], Fu et al. [30] and Chen et al. [22] are kernel specific. However, other methods are portable and can be used with any SVM, regardless of the type of kernel.

3.2. Rule extraction

In this section, an overview of rule extraction methods is presented. These methods are grouped into four categories based on the SVM components utilized for rule extraction. In particular, the first category of algorithms treats the SVM as a closed-box, the second extracts rules directly from the SVs, the third utilizes the SVM SVs and decision function, while the fourth utilizes the SVs, the decision function and the training data.

3.2.1. Methods utilizing the SVM model as a closed-box

Methods in this group treat the SVM as a black-box, and extract rules that describe the relationship between the model's inputs and outputs [24,32,35–37].

The idea is to create artificially labelled examples where the target class of the training data is replaced by the class predicted by the SVM as follows (please refer to Section 2):

$$f(x) = sign\left(\sum_{s=1}^{sv} \alpha_s y_s K(x_s, x) + b\right) \tag{9}$$

So, the right hand side of Eq. (9) is dealt with as a closed-box, without even looking at its individual components, in particular SVs.

The artificial data set is then used with another machine learning technique with explanation capability such as decision tree learners, which learn what this SVM has learned.

The main steps involved in these methods are shown in Fig. 2. The most commonly used decision tree learners are modified TREPAN [53], C5 [54] and CART [55]. Similar methods have been used to extract rules from ANNs, which were referred to as learning-based methods [23].

However, the methods in this category are not classifier specific as they do not utilize any model specific components in rule extraction. The aim is simply to model the output of the original classifier using another classifier with better comprehensibility.

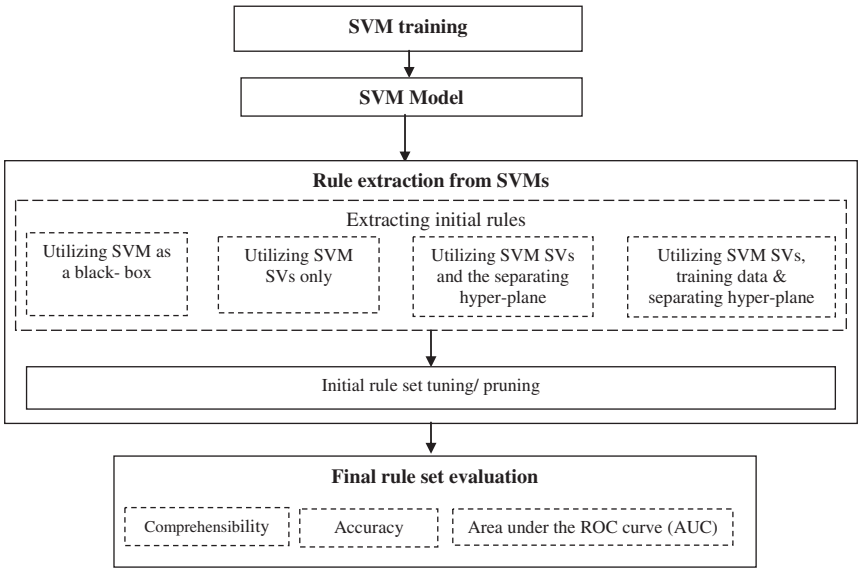

Fig. 1. Modules for rule extraction from SVMs.

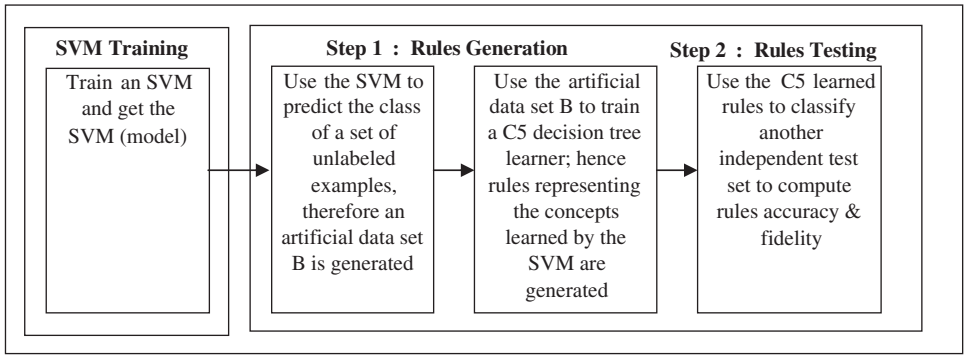

Fig. 2. Decision tree-based rule extraction [24,32,33,35–37].

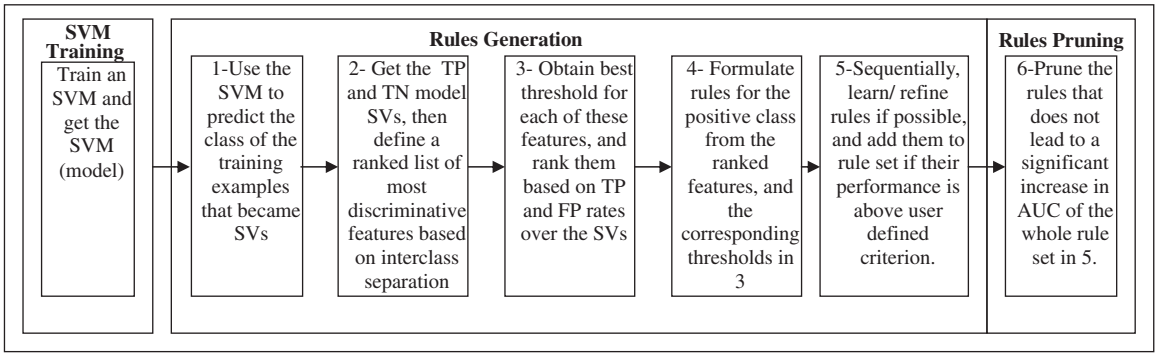

Fig. 3. Abstract representation of SQRex-SVM [34].

As shown in Table 3, relatively small numbers of rules, with both high accuracy and fidelity have been extracted using these methods.

In a related study, *which does not aim to propose a rule extraction method*, He et al. [38] suggested SVM_DT as a method for interpreting prediction of protein secondary structure. However, the method is motivated by the need to integrate the good generalization performance of SVMs and the comprehensibility of decision trees. In particular, the method utilizes an SVM as a preprocessing step for the decision tree. The authors argue that the use of an SVM as a preprocessing step can generate better and/or cleaner data than the original data set, where some bad ingredients and weak cases can be reduced. This argument, however, is only truly valid if optimum training parameters for the SVM can be found. The authors also argue that the extracted rules have biological meaning and can be interpreted [38].

3.2.2. *Methods utilizing the SVs only*

Methods within this group extract rules utilizing the SVM's SVs. So, unlike methods described in Section 3.2.1, the right hand side of Eq. (9) is no longer a closed-box, where the *only* the SVs are utilized for rule extraction. (SVs $\{x_s,y_s\}$, $y_s \in \{-1,1\}$, $x_s \in SV_s$ are subset of training examples where $\alpha \neq 0$. Please refer to Eqs. (7) and (8)).

Different methods have been used to extract rules from the SVs as summarized in the following paragraphs.

One of the most recent method in this group termed SQRex-SVM [34] extracts rules directly from a subset of the SVs of an SVM using a modified sequential covering algorithm [56] and based on an ordered search of the most discriminative features, as measured by inter-class separation. An initial set of rules is formed from these features, which are then pruned using user defined criteria, and utilizes the concepts of Coverage or PN space [57] and its relationship to the area under the receiver operating characteristic curve (AUC).

The motivation behind using the AUC is to have a reliable method to directly control the tradeoff between the rule set performance and comprehensibility. Fig. 3 provides an overview of the algorithmic steps.

Rules' performance is evaluated in terms of accuracy, fidelity, comprehensibility, TP & FP rates and AUC. Rules produced by SQRex-SVM exhibit both good generalization performance and comprehensibility. In addition, the extracted rules have high fidelity to the SVM from which they were extracted.

The second method in this group by Chaves et al. [31] proposes a method of extracting fuzzy rules from SVMs. The main idea here is to project each feature, in each of the SVs along its coordinate axes, forming a number of fuzzy sets with triangular membership functions of the same length [58]. Fuzzy membership degrees are then computed and each of the SVs is then assigned to the fuzzy set with the highest membership degree. One fuzzy rule is then extracted from each SV. Fig. 4 provides an overview of the algorithm steps.

One limitation of this algorithm is that, it may not be suitable for categorical and binary attributes as the evaluation of membership degree value is non-trivial in these cases. Rules extracted by this method appear to be of poor quality. The best results were 53.2% for accuracy and 74.59% for consistency (fidelity). However, the authors argue that if the objective is explanation, then poor classification performance is acceptable. However, to argue that a rule set provides an explanation to the SVM, it should at least exhibit high fidelity to that SVM. Furthermore, the extracted rule set comprehensibility is low as the number of the extracted rules is large, with all the attributes appearing as antecedents.

The last method in this category utilizes a decision tree learner with the SVM's SVs. The steps here are similar to those described in Sections 3.2.1 in that it also creates an artificial data set; but instead of using all the training data, it utilizes only the SVM SVs

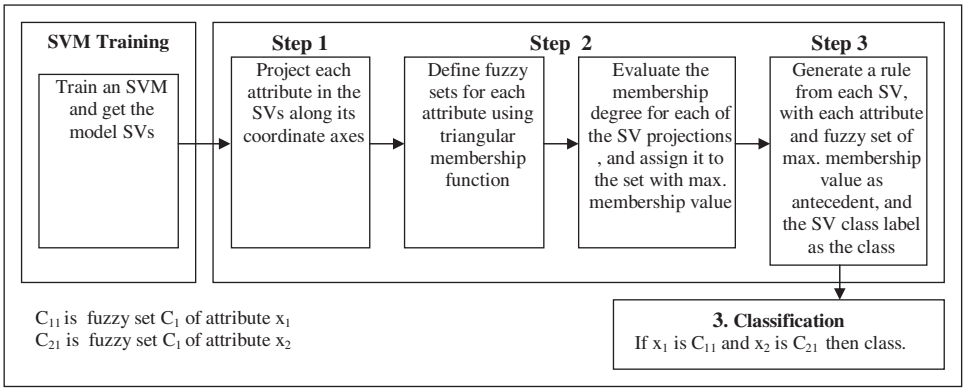

Fig. 4. Abstract representation of the fuzzy rule extraction algorithm [31].

[33]. So, for each of the SVs, the original class labels is replaced with the SVM predicted class according to Eq. (9). A C5 decision tree learner [54] is then used to generate rules from the modified SVs.

The extracted rules by this method [33] demonstrate a high degree of fidelity, accuracy and comprehensibility as shown in Table 3. However, one limitation of this algorithm is its sensitivity to the noise in the training data sets, as the rules are extracted from all types of the SVM model support vectors.

The method proposed by Martens et al. [37] is similar to methods described in Section 3.2.1, but has an additional step which generates additional training examples 'close to' randomly selected SVs whose class labels are predicted by the SVM. The generated examples are then used with the modified training data to train different decision tree algorithms that learns what the SVMs have learned.

3.2.3. *Methods utilizing SVM SVs and the separating hyper-plane*

Pursuing the same idea of utilizing SVs, Fu et al. [30] suggested a method (RulExSVM) for rule extraction from non-linear SVMs, trained with a radial basis function (RBF) kernel utilizing the SVM decision boundary in addition to the SVs.

The idea behind this method [30] is to find hyper-rectangles whose upper and lower corners are defined by finding the intersection of each of the SVs ($x_s \in SV_s$) with the separating hyper-plane (obtained by solving the optimization problem in Eq. (8)):

$$\left(\text{maximize } w(\alpha)=\sum_{i=1}^{l}\alpha_i-\frac{1}{2}\sum_{i=1,j=1}^{l}\alpha_i y_i \alpha_j y_j K(x_i,x_j) \quad C\geq\alpha_i\geq 0 \quad \forall_i, \quad \sum_{i=1}^{l}\alpha_i y_i=0\right)$$

where $K(x,x_i)=\exp[-||x-x_i||^2/(2\sigma^2)]$ and σ a constant

RulExSVM proceeds in three phases: initial rule generation, then tuning and finally rule set pruning. In the initial phase, hyper-rectangles are defined by the intersections of lines extended from each of the SVs with the SVM decision boundary obtained by Eq. (8). In the tuning phase, these hyper-rectangles are resized to exclude outliers from other classes. Finally redundant hyper-rectangles (rules) are removed in the pruning phase. Fig. 5 shows the different phases of RulExSVM.

One limitation of this approach is that it is only valid for rule extraction from RBF kernel SVMs. However, the authors argue that this method could be extended to other types of kernels though they provide no framework for this. Another potential limitation is the requirement for normalization of all attributes to lie between (0, 1) before training. Normalization should be handled with care in applications such as medical diagnosis because some normalization methods may be susceptible to noise in the data. In addition, it can be problematic to enforce an ordering upon categorical features. Furthermore, this also adds an extra burden on the rule extraction process, as the attributes have to be transformed back to the original values for the extracted rules to make sense. The approach also does not provide a method to avoid defining redundant hyper-rectangles at the rule generation phase, rather than removing them later at the pruning phase. In the reported results on a breast cancer data set, the author reported seven rules for the majority class (benign), while the minority class (malignant) is assumed to be the complement (default). The performance results of the extracted rules by the algorithm are shown in Table 3.

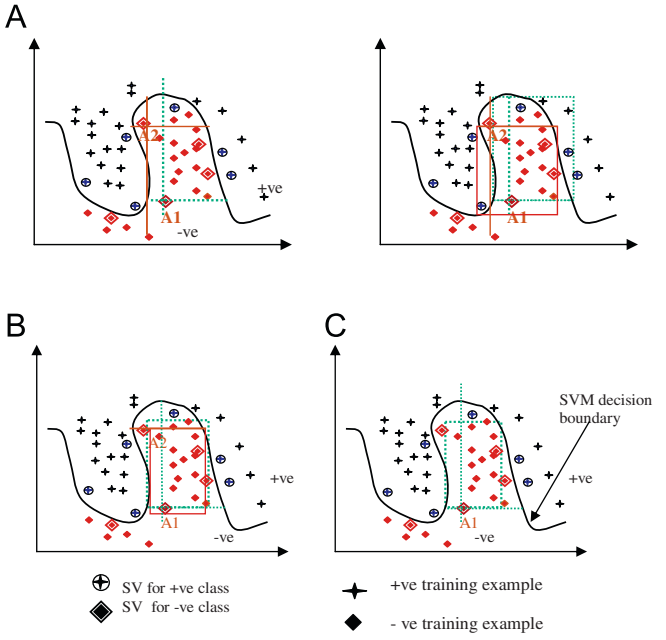

Fig. 5. RulExSVM phases in a two dimensional space. Adapted from Fu et al. [30]: (A) Rule generation phase: lines extended and hyper-rectangles constructed for A1 and A2 SVs. (B) Tuning phase: excluding outliers. (C) Pruning phase: removing overlapped hyper-rectangle A2.

3.2.3. *Methods utilizing SVs, training data and separating hyper-plane*

Similar to the methods described in Section 3.2.3, methods in this category also utilize SVs and the separating hyper-plane; however, they also utilize the training data. Training data is used to form regions in the input space, while SVs and the separating hyper-plane are used to control the sizes of these regions.

The first method by Núñez et al. [25–28] introduced the SVM+ prototype method. The main idea of this method is to utilize a clustering algorithm to determine prototype vectors for each class, which are then used together with the SVs to define regions (ellipsoids and hyper-rectangles) in the input space. Geometrical methods are used to define the region as follows: Initially, one ellipsoid (hyper-rectangle) is created, with the prototype vector as the centre of this ellipsoid (cluster), and an SV of the same class to define the first axis and the vertex of this ellipsoid. Only SVs with $\alpha < C$ (*lies outside the margin d, and at a distance* $\geq 1/(\|w\|)$ from the hyper-plane) are chosen for that purpose (please refer to Eq. (6)).

The rest of the axes and the associated vertices are then determined. If no outliers are found in the defined region (negative partition test), the region is translated into a rule; otherwise, an iterative procedure is followed to divide this ellipsoid (hyper-rectangle) into more regions that adjust to the SVM decision boundary. An ellipsoid is then translated into an equation rule, which is not easy to interpret by end users, while a hyper-rectangle is translated into an interval rule in the form

IF $X1 \in [a,b]$ AND $X2 \in [c,d]$ AND $X3 \in [e,f]$ ... THEN CLASS

Fig. 6 shows different phases of the algorithm.

The extracted rules are of high accuracy and fidelity as shown in Table 3. However, the algorithm has two main limitations: the first is that the number and the quality of the extracted rules depend upon the initial values of the prototype vectors which define the centres of the clusters, and in turn are affected by the initial parameters of the clustering algorithm. The second limitation is that the method does not scale well. In the case of larger number of input features and/or training examples, the comprehensibility of the extracted rule set will deteriorate as more clusters are likely to be formed, while all the features are present as rules antecedents [37].

In a similar study, Zhang et al. [29] proposed the hyper-rectangle rule extraction (HRE) algorithm which also utilizes a clustering algorithm. The algorithm first uses the support vector clustering (SVC) algorithm [59] to find prototype vectors for each class. Small hyper-rectangles are then defined around these prototypes using a nested generalized exemplar algorithm, which are incrementally grown until one of the stopping criteria is met. The stopping criteria are meant to control the size of the hyper-rectangles, hence the quality of the rules generated. Propositional again, IF-Then rules in the form IF $X1 \in [a,b]$ AND $X2 \in [c,d]$ AND $X3 \in [e,f]$... THEN CLASS are extracted by projecting the hyper-rectangles onto coordinate axes. The quality of the rules is measured in terms of accuracy, support and confidence. Fig. 7 shows the different stages and the stopping criteria for HRE.

As shown in Table 3, the HRE has high accuracy. However, the comprehensibility of the rules is low as all the input features appear as rule antecedents. One more potential disadvantage of this algorithm is the difficulty of selecting the SVC algorithm parameters. These parameters are critical in defining the number of clusters and hence the number of rules, as reported in Zhang et al. [29].

In another study, Fung et al. [13] suggested a different approach for rule extraction from linear SVMs, or from any hyper-plane-based linear classifiers, based on a linear programming formulation of the SVMs with a linear kernel.

The approach handles rule extraction as multiple constrained optimization problems to generate a set of non-overlapping rules. Each extracted rule defines a non-empty hyper-cube which has

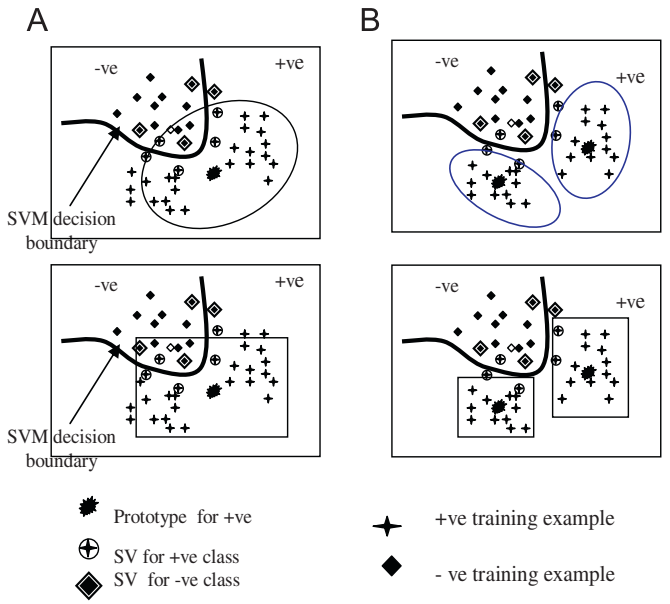

Fig. 6. SVM+ phases [26]: (A) One region (cluster) including outliers. (B) Region division to exclude outliers.

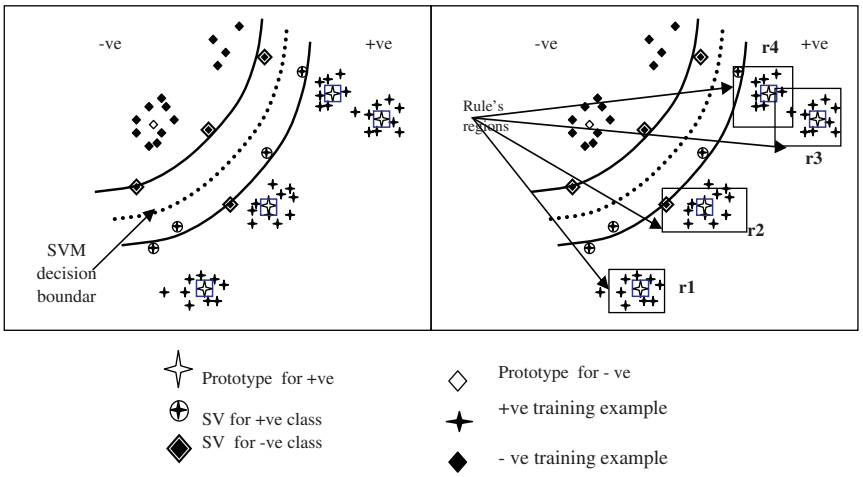

Fig. 7. Hyper-rectangle region generated for each cluster per class, adapted from [29]: (A) Iteration '1'. (B) Iteration '*n*' showing different stopping criteria.

one vertex that lies on the separating hyper-plane. The rule extraction algorithm iterates over the training data in each half-space, to find rules for the examples which have not been covered by any previous rule using a depth first search. The extracted rules by this method are in the form: $(x \le a)$ AND $(y \le b)$ AND $(z \le c)$…THEN CLASS.

Two criteria for finding (growing) an 'optimal rule' are used: volume maximization of the hyper-cube (VM) and point coverage maximization (PCM) which maximizes the number of examples covered by a hyper-cube. Rules that cover only one training example are discarded.

As shown in Table 3, the algorithm produces large number of rules, which have not been evaluated on an independent test set.

One limitation of this approach is that the extracted rules (generated hyper-cubes) cover the training data, so, they may not provide an explanation for new unseen data. However, the authors have suggested modifying the volume maximization rule extraction method to generate only one rule with maximum volume that may contain (generalize over) the test examples.

In a related study, and again by finding non-empty hyper-cubes, Chen et al. [22] propose a novel and an interesting rule extraction algorithm for gene expression data to improve the comprehensibility of SVMs. The algorithm constitutes one component of a multiple kernel SVM (MK-SVM) scheme, consisting of feature selection (MKI), prediction modeling and rule extraction MK-SVMII.

In the feature selection module, a new single feature kernel (MKI) is proposed which transforms the computationally expensive feature selection problem into finding sparse feature coefficients representing the weight of a single feature kernel. Features with zero coefficients have no impact on the output of SVM and can be discarded.

Using the MK-SVMI Kernel, and making some substitutions into the ordinary SVM quadratic programming formulation, a new mixture coefficient is introduced. If an input vector has at least one non-zero mixture coefficient, the corresponding input data vector becomes an SV. In this case the Lagrange multiplier in the ordinary SVM formulation is replaced by the mixture coefficient in MK-SVMII.

The rule extraction method proposed for the MK-SVMII is similar to the one addressed in Fung et al.'s [13] which solve multiple optimization problems to find rules that described non-empty hyper-cubes in each class half-space. However, in MK-SVMII, SVs are used as vertices of hyper-cubes, where a series of hyper-cubes approximates the subspace of each class.

Chen et al. [22] argue that a rule with linguistic labels can facilitate the analysis and the comparison gene expression levels reported by different techniques. In fact, we believe that this argument is also valid for the results reported by different techniques in other domains as well, which may have a positive significant contribution to sharing results and concepts, which constitutes the main objective of semantic web.

The following measures for the extracted rules quality are suggested in addition to the rules comprehensibility: Soundness, which measures the number of times a rule is correctly fired; completeness, which measure the number of times the sample is correctly classified by a specific rule, which are not covered by any previous rule and false-alarm, which measures the number of times each rule is misfired [22]. The completeness, soundness and false-alarm measures are then used to control the generated rules quality, then for rules pruning and ordering to get a final rule set with both better comprehensibility and performance.

The extracted rules by this method have good comprehensibility, good generalization performance while compact gene subsets were found [22]. Furthermore, the authors argue that multiple good diagnostic gene subsets found to be useful in defining possible pathways of genetic network.

In this section, rule extraction methods have been analyzed based on the model components utilized for rule extraction. Please also refer to Table 2 in Section 6, where the **main features of rule extraction algorithms** are summarized and compared.

4. Related work: rule extraction from ANNs taxonomy

As discussed in Section 1, there has been significant number of algorithms for rule extraction from ANNs. Andrews et al. [23] proposed a taxonomy composed of five criteria to evaluate and group these algorithms into different families. The five criteria suggested in this taxonomy are translucency, rule quality, expressive power, portability and algorithmic complexity. Most of these criteria have been also used to classify SVM rule extraction algorithms, in particular the translucency and rules quality metrics [37,49]. The following paragraphs summarize these criteria.

4.1. Translucency

Translucency refers to the extent to which the details of the ANN internal model structure are utilized by rule extraction algorithm. Therefore, rule extraction algorithms are classified into three types: decompositional, pedagogical and eclectic.

The decompositional (or transparent) algorithms extract rules at the level of individual units (hidden and output units) of the ANN which are then combined to form rules. In most of these algorithms, the output from each hidden and output unit is close to a binary one or zero, which is then mapped to the rule consequent. This is done by searching for a set of links whose summed weights exceed the unit bias regardless of the activation value on other links [23].

In the Pedagogical, or learning-based, rule extraction techniques, the trained ANN is treated as a black-box, and the algorithms extract rules that map the network input features to the output function computed by the network [60]. These methods are used in conjunction with other learning algorithm with explanation capability which learns what the ANN has learned.

The eclectic algorithms are hybrid techniques which borrow components from both decompositional and learning-based approaches [23].

4.2. Quality

The quality of the extracted rules is a key measure of the success of the rule extraction algorithm. Four criteria for assessing the quality of the extracted rules were suggested in [23]. The four criteria are accuracy, fidelity, consistency and comprehensibility. Accuracy measures the correctness of classification of previously unseen examples, and is given by

$$\text{Accuracy} = \frac{\text{No. of patterns correctly classified by the rules}}{\text{Total number of patterns on a test set}}\%$$

Fidelity indicates the extent to which the rules mimic the behavior of the ANN from which they were extracted:

$$\text{Fidelity} = \frac{\text{No. of patterns where the classification of rules agree with the classification of ANN}}{\text{Total number of patterns on a test set}}\%$$

Table 1
Similarities between ANNs and SVMs.

Criteria	ANNs	SVMs
Model type (for classification)	Black-box/separating hyper-plane	Black-box/separating hyper-plane
Parameters to be learned	Weight, thresholds	α_i (Lagrange multiplier), threshold
Model variables	Network structure, initial weights	Kernel function, regularization parameter
Dealing with nonlinearity	Hidden layers, nonlinear activation functions	Mapping to higher dimensional feature space, non-linear kernels
Learned knowledge	Embedded in the network architecture (e.g., the number of hidden units), the activation function and a set of weights	Represented by the model support vectors, and the weights associated to them (α) and the threshold
Decision function	$f(x)=sign(\sum_{i=1}^{lhu} w_i Z_i(x))$ where lhu are last hidden layer, w_i are the weights from lhu to the output layer	$f(x)=sign(\sum_{s=1}^{sv} \alpha_s y_s K(x_s,x)+b)$ where sv is the model support vectors and α_i is Lagrange multiplier

Consistency is the extent to which the generated rule sets produce the same classification of unseen examples, even if they are extracted from different models trained on the same problem. For comprehensibility evaluation, the number of rules plus the number of antecedents per rule are commonly used in the literature. However, this can only be seen as a rough surrogate for a hard to measure concept.

4.3. Expressive power

This measure refers to type or the language of the extracted rules. Propositional (IF Then) rules are the most commonly extracted rules. However, other types are also extracted, such as fuzzy rule sets [61], and finite state machines [62].

4.4. Portability

Portability refers to the independence of a rule-extraction method of the ANN architecture and/or a training method.

4.5. Algorithmic complexity

This dimension refers to time and computational complexity of the rule extraction algorithm. For example, Golea [63] showed that extracting best N-of-M rule from a single-layer network is NP hard. Furthermore, he also showed that extracting minimum disjunctive normal form (DNF) expression from a trained feed-forward ANN is NP hard in the worst case.

5. Similarities between ANNs and SVMs

Table 1 summarizes the similarities between ANNs and SVMs [52,58,64].

From Table 1, it can be seen that both SVMs and ANNs share common properties, and both of them produce black-box models, which was the main motivation behind the rule extraction studies. Furthermore, it can be concluded that the rule extraction classification proposed by Andrews et al. [23] considering the translucency dimension is also valid for SVMs at the conceptual level. For example, methods discussed in Section 3.2.1 can be categorized as learning based, while methods described in Sections 3.2.2–3.2.4 can be considered as decompositional, as they utilize SVM internal components (SVs). However, methods like [33,37] can also be considered as eclectic, as they utilize SVM SVs, and also use decision tree learners to extract rules, as in the learning-based methods.

6. Discussion

The main features of the reviewed rule extraction methods are shown in Table 2. The table rows show the different criteria of each algorithm and the extracted rules. In particular the main idea of the algorithm, the portability (yes/no), the language of the extracted rules (crisp/fuzzy rules), if the methods have utilized an additional machine learning technique (yes/no), or feature selection module (yes/no), whether the extracted rules include relevant features only (yes/no) and if they overlap (yes/no). The last three rows describe the quality of the extracted rules in terms accuracy, fidelity and comprehensibility (low/moderate/high). The algorithms' precise performance results over common benchmark data sets methods in terms of comprehensibility (number of rules/number of antecedents), accuracy and fidelity are shown in Table 3. The rows represent different performance measures and data sets, while columns represent different rule extraction algorithms.

Results in Table 3 suggest the following:

- The most comprehensible rule sets are produced by the approaches in [24,32,34–36], followed by RulExSVM [30]. These rules also appear to have good generalization performance compared to the SVM form which they were extracted. Therefore, these methods are suitable for applications that require comprehensible rules, like medical diagnosis. However, in He et al. method [38] produced large number of rules.
- The methods suggested in [24–28,32,34–36,65] are also portable, as they are kernel independent.
- The SVM+ introduced by Núñez et al. [25–28] also produces a small number of rules, however, all the features are present as antecedents of these rules. This limits their explanation capability as no indication is given about the most important features for the classification.
- The methods described in [13] (hyper-cubes) produce large number of rules.
- Rules extracted from data sets which have discrete variables have better accuracy and fidelity than the rules extracted from data sets with continuously valued attributes.
- Chen et al. [22] introduced, for the first time, a rule extraction method from SVMs for gene expression data, which produces rule sets of good comprehensibility and soundness.

It is worthwhile to notice here that except for the gene expression data sets, the majority of the data sets used to test the rule extraction methods in these studies have a relatively small number of features. Therefore, one may question both the scalability of these methods, and their performance when applied to high dimensional data sets. Finally, the consistency of the rules is better in the methods that do not use clustering algorithms. This shows sensitivity to the selection of the prototype vectors and hence the defined clusters (rules' regions).

Table 2
Main features of rule extraction algorithms.

Criteria	Chen et al. [22]	Barakat/Bradley [34]	Nunez et al. [28]	Torres and Rocco [35] Martens et al. [36]	Barakat/ Diederich [24,33]	Fung et al. [13]	Zhang's et al. [29]	Fu et al. [30]	Chaves et al. [31]
Basic idea	Extract rules that cover non-empty hyper-cubes with SVs as vertexes in each half space class by solving several optimization problems	Extracts rules directly from the SVM SVs using a modified sequential covering based on an ordered search of the most discriminative features	Extract rules from ellipsoids and hyper-rectangles formed using clustering algorithms and geometrical methods	Use the value of decision function for the training set and artificially generated SVs [37] with modified TREPAN and other decision tree learners to extract rules	Use the value of decision function for a testing data set/SVs with decision tree learner to extract the rules	Extract rules that cover non-empty hyper-cubes in a certain half space by solving several optimization problems	Extract rules from hyper-rectangles formed using SVC algorithms and the nested generalized exemplar algorithm and some heuristic conditions	Extract rules from hyper-rectangles formed using the model support vector and the SVM decision function (hyper-curve)	Extract fuzzy rules utilizing the model SVs. Fuzzy sets with max. membership value as antecedents, SV class as consequent
Portable	No	Yes	Yes	Yes	Yes	No	Yes	No	Yes
Language of the rules	Crisp If-then rules	Crisp if-then rules	Crisp if-then rules	Crisp if-then rules	Crisp if-then rules	Crisp if-then rules	Crisp if-then rules	Crisp if-then rules	Fuzzy if-then rules
Use of additional machine learning techniques	No	Sequential covering algorithm	Clustering Algorithm	Decision tree learners	Decision tree learners	No	Clustering Algorithm & nested generalized exemplar algorithm	No	No
Use of feature selection	Yes	Yes	No	No	No	Yes (embedded)	No	No	No
Rules include relevant features only	Yes	Yes	No	Yes	Yes	Yes	No	Yes	No
Rules overlapping	No	Yes	Yes	Yes	Yes	No	Yes	Yes	Yes
Rule quality									
Accuracy	High	High	High	High	High	Not reported	High	High	Low
Fidelity	Not reported	High	High	High	High	Not reported	Not reported	High	Low
Comprehensibility	High	High	Moderate/low	High	High	Low	Low	High	Low

Table 3
Benchmarking of results of the different rule extraction methods on common data sets.

Criteria	Data sets	Barakat and Bradley [34]	Torres and Rocco [35]; Martens et al. [36]	Barakat and Diederich [24,33]		Fung et al. [13]		Nunez et al. [28]		Zhang et al. [29]	Fu et al. [30]
		SQRex_SVM	TREPAN/ M-TREPAN	Learning based	Eclectic	Max area	Max coverage	Ellipsoid	Interval	HRE	Ru-lExSVM
No of rules/ antecedent	Diabetes	**3/1**		***2/1***	**2/2**					Not reported	16/4.5
	Breast cancer	**3/1**	5.4/?	***3/1***	**3/2**	37/5	10/5	3/?	5.8/9		7/5.3
	Heart diseases	**3/1**		***3/2***	**2/2**	32/6	18/6	5/?	20/13		
	Iris	**1/2**	6.7/?					7/?	4.7/4	?/4	3/1.3
Accuracy on test set %	Diabetes	92		91.0	93.0	Not reported				88.7	80.3
	Breast cancer	96	95.0	85.0	82.0			96.5	96.2		97.5
	Heart diseases	74		74.0	83.0			86.3	83.7	95.0	
	Iris	**100**	96.2					96.0	96.0	97.0	98.0
Fidelity %	Diabetes	88		93.0	92.0	Not reported				Not reported	**98.2**
	Breast cancer	97	97.2	85.0	91.0			98.0	98		**99.3**
	Heart diseases	89		79.0	88.0			97.0	96		
	Iris	91	97.0					98.0	99.3		**99.2**

- Chen et al. [22], Chaves et al. [31] and Martens et al. [37] rule extraction method results are not listed as the data sets used are different, so comparison is not feasible.
- The results presented in this table are the published by the authors, and no additional experiments have been executed.

From the previous discussion, it can be seen that the different methods produce rules of differing accuracy and fidelity and that for the majority of the methods, a relatively large number of rules are generated.

The following paragraphs also propose an alternative categorization and analysis of the algorithms discussed in Section 3.

6.1. Region-based rules

In this category, regions are formed in the input space, utilizing the SVM decision functions and/or the model SVs, which are then mapped to rules. Three types of region are formed: Ellipsoids [25–28], hyper-rectangles [25–29] and hyper-cubes [13,22]. In (SVM+) [25–28] and 'HRE' [29], the authors used clustering algorithms to define prototype vectors which are then utilized with SVs to construct the region, while in 'RulExSVM' [30], the algorithm utilizes the SVM decision hyper-curve and the support vectors to construct hyper-rectangles. Fung et al. [13] and Chen et al. [22] defined the hyper-cube region by solving multiple constrained optimization problems.

The rules extracted from the hyper-rectangles [25–29] include all the input features in their antecedents, even if some of these features do not contribute to the classification decision. In the case of higher dimensional data sets the comprehensibility of these rules will suffer. Therefore, utilizing and additional feature selection module could be useful for the quality of the extracted rules by these methods.

The method described in [13] produced a relatively large number of rules, which again limits their explanation capability. Perhaps adding a pruning phase could have been useful for reducing the number of regions/rules.

The rules extracted from gene expression data utilizing the MK-SVMII [22] are comprehensible, and of good performance. However, the methods described in Fung et al. [13], Fu et al. [30] and Chen et al. [22] are SVM kernel specific.

6.2. Decision tree-based rules

Methods in this category utilize decision tree learners to learn what the SVM has learned. The main idea is to create an artificial data set (the original (target) class of the training data is replaced by the SVM predicted class) then, a decision tree learner is used to devise rules from that data set [24,32,35–37]. However, the methods described in [33] create the artificial data set from the SVs only.

Martens et al. [37] have also proposed generating a large number of extra examples close to the SVM SVs, to be used, in addition to the training data as an artificial data set. However, generating large number of examples (in most cases, more than the original training data) from randomly selected SVs does not help for rule extraction. As mentioned in the introduction, rule extraction is important for applications such as medical diagnosis, or security related issues. Generating 1000 extra examples may lead to creating cases which have never existed in reality, so it may not help in devising relevant rules. In addition, this will add extra burden on the rule extraction steps, and slows down the algorithm.

On the data sets examined, methods in this family produce concise number of rules with good accuracy and fidelity. Furthermore, these methods can be used for rule extraction from any SVM kernel type.

6.3. Fuzzy rule extraction

As shown in Section 3, the fuzzy rule extraction method [31] belongs to the family utilizing SVs for rule extraction. However, unlike the rest of the described methods in this paper, this method extracts fuzzy rules. The rules produced by this method are of low accuracy, fidelity and comprehensibility. Therefore, this method could also have benefited from a feature selection step prior to projecting the SVs along the coordinate access. The number of projections could have been reduced, which may lead to better rules' comprehensibility.

6.4. Sequential covering rule extraction

The SQRex-SVM method [34] also extracts rules directly from the SVs using a modified sequential covering algorithm. Rules are generated based on an ordered search of the most discriminative

features, as measured by interclass separation. In this method, a tradeoff between rules comprehensibility and performance is enforced. The algorithm extracts rules of both good generalization performance and comprehensibility.

7. Future research questions

As summarized in the discussion, we have not highlighted any new issue specific to rule extraction from SVMs as opposed to rule extraction from ANNs. However, we can conclude that the same issues related to rule quality and different tradeoffs raised for ANNs are still pending and are not resolved to date for either ANNs or SVMs

Clearly, a potential area for future research is to develop algorithms that are able to directly tradeoff performance and comprehensibility. Naturally, this family of algorithms needs a reliable measure of performance in order to correctly estimate the effect of producing comprehensible rules on the generalization performance, and the extent to which these rules still provide a good explanation to the SVM model. This is an important topic of research as some studies such as Provost et al. [66] and Bradley [67] have shown that accuracy alone is not a reliable measure for comparing the performance of two different classifiers (in this case the SVM and the extracted rules or different rule sets extracted from an SVM). However, Barakat and Bradley [65] have suggested using AUC to assess the quality of the extracted rules from an SVM.

One more aspect that needs to be considered in rule extraction algorithms is the relative importance of features. This will lead to rules that are both more accurate and comprehensible. In the reviewed methods, only SQRex-SVM [34] and MK-SVMII [22] have explicitly executed a feature selection step prior to rule extraction, which has contributed to extracting rule sets of good comprehensibility.

Another important topic in this area requiring further investigation is how to extend rule extraction methods to the case of SVM incremental and active learning, and how to embed a rule extraction method into an online data mining applications. In these cases, more attention should be paid to the time complexity of the proposed algorithms.

8. Summary and conclusions

We have reviewed and analyzed the published algorithms for rule extraction from SVM classifiers and compared their main features and performances according to multiple criteria. However, these comparisons do not offer a clear and reliable picture as to which method is the most favorable in general. Therefore, a specific method can only be preferred in the context of the requirements of a specific application area and the final purpose of the extracted rules. Clearly, one should favor methods that produce the most comprehensible rules, exhibit the best generalization performance and are equivalent (in some sense) to the SVM model from which they were extracted. Unfortunately, as the results in Table 2 show, there is currently no one method that can fulfill all of these criteria simultaneously. Therefore one should consider the following two tradeoffs:

- Tradeoff between accuracy and comprehensibility.
  This depends on the purpose of rule extraction. If the extracted rules are required to provide an explanation only, and are not intended to replace the SVM classifier, then the accuracy can be sacrificed (to some extent) to get better explanation. The SQRex-SVM method [34] is the only method to date which was successful in enforcing such a tradeoff:
- Tradeoff between fidelity and comprehensibility.
  The same argument applies here for the tradeoff between the fidelity and comprehensibility. If the SVM model is a complex one, it will generally need a larger number of rules to give the same performance as that of SVM. This tradeoff control was introduced in the TREPAN algorithm for rule extraction from ANNs [60]. So, again if the purpose is only to provide explanation, then fidelity can be compromised (within limits) to favor better explanation. Again, The SQRex-SVM [34] is the only method to date which was successful in enforcing this tradeoff.

Finally, if the purpose of rule extraction is to generate a rule set that replaces the SVM (which is not likely to be the case in the studies presented in this paper), then comprehensibility can be sacrificed to favor better fidelity and accuracy. In some cases, the extracted rules have better generalization performance than the SVM model from which they were extracted. This points to another tradeoff between accuracy and fidelity, as suggested by Zhou [68] for rule extraction from ANNs.

Acknowledgments

The authors appreciate the valuable comments of associate editor and the anonymous reviewers, which have positively contributed to the quality of the paper.

Appendix A. Abstract representation for different phases in rule extraction algorithms

As described in Section 3, all the rule extraction algorithms start by training an SVM and end with an evaluation of the extracted rules' quality in terms of a number of different performance metrics. Figs. 2–7 show an abstract view of the main phases for each family of algorithms.

A.1. Decision tree-based rule extraction

As shown in Fig. 2, after SVM training, and in the rule generation phase, the value of the SVM decision function (as per Eq. (9) is utilized to create a synthetic data set that represents the concepts learned by the SVM. The data set is then used to train a decision tree learner which simply learns what the SVM has learned. In step 2, rule quality is evaluated on an independent test set.

A.2. Sequential covering rule extraction

Fig. 3 shows the main steps for the only algorithm in this family [34]. SQRex-SVM proceeds in two main steps after the SVM training: rule generation, then rule pruning. In the rule generation phase, the algorithm first picks only TP and TN SVs which are then utilized to find a set of most discriminative features. The most discriminative features are the ones with a statistically significant difference ($p<0.05$) between their means in TP and TN SVs. The best threshold for predicting the positive class for each of these features is then defined. These thresholds are then used to rank the features based on their performance as measured by TP and FP rates. Ranked features are used to formulate initial rules for the positive class, with the negative class as a default. Rules are learned/refined sequentially from both TP and true negative (TN) SVs.

A pre-pruning strategy is adapted to prune rules with performance below a user defined threshold. Rules with acceptable performance are added to the initial rule set. A post-pruning

method is then utilized to prune the rules that do not result in a statistically significant ($p < 0.05$) increase in the rule set AUC.

A.3. Fuzzy rule extraction

Unlike the rest of the methods in this review, Chaves et al. [31] suggested a method to extract a fuzzy rule set from a trained SVM. Fig. 4 shows the main phases, which starts by projecting each feature in the SVs along its coordinates. The maximum membership degree to a predefined fuzzy set is then obtained for each of the SVs. As shown in step 3, a fuzzy rule is generated from each SV for each class. The rules' quality is then evaluated on an independent test set.

A.4. Region-based rule extrtraction

A.4.1. RulExSVM phases

This method [30] utilizes the SVM decision boundary and SVs to define regions from which rules are extracted as described in Section 3.2.3. Fig. 5A shows the lines extended from SVs A1 and A1, and the hyper-rectangles defined by their intersections with the SVM decision boundary. Fig. 5b shows the tuning phase, where the two hyper-rectangles were chopped along coordinate axes to exclude outliers from other classes, while Fig. 5c shows the pruning phase which removes the overlapping hyper-rectangle A2 (redundant rules).

A.4.2. SVM+prototype phases

As described in Section 3.2.4, the algorithm [25,28] utilizes the SVM SVs, training data and the separating hyper-plane to define regions (ellipsoids and hyper-rectangles), which are then refined and translated into rules.

Fig. 6A shows the first iteration of the algorithm where an initial ellipsoid (hyper-rectangle) is defined with outliers (positive partition test). Fig. 6B shows a later iteration after the division of the ellipsoid (hyper-rectangle) to exclude outliers.

A.4.3. Hyper-rectangle rule extraction (HRE) phases

Again, and as described in Section 3.2.4, the algorithm [29] utilizes the SVM SVs, training data and the separating hyper-plane for rule generation. Fig. 7A shows the initially defined, small hyper-rectangles around the prototype vectors. Fig. 7B shows larger hyper-rectangles. Given a user defined minimum confidence threshold (MCT), the algorithm defines the following criteria are defined to stop growing the hyper-rectangles:

(1) All examples of a cluster are covered by a hyper-rectangle and the confidence is less than the (MCT), as for the r1 hyper-rectangle in Fig. 7B.
(2) The hyper-rectangle covers a support vector of the cluster, as for the r4 hyper-rectangle in Fig. 7B.
(3) The hyper-rectangle covers a training vector from a different class and the confidence is less than the MCT, as for the r2 hyper-rectangle in Fig. 7B.
(4) The hyper-rectangle covers a prototype of a different cluster as for the r3 hyper-rectangle in Fig. 7B.

References

[1] W. Frawley, G. Piatetsky-Shapiro, C. Maltheus, Knowledge Discovery in Databases: An Overview, MIT Press, 1991.
[2] H. Byun, S.-W. Lee, Applications of support vector machines for pattern recognition: a survey, in: S.-W. Lee, A. Verri (Eds.), Lecture Notes in Computer Science, 2002, pp. 213–236.
[3] R. Burbidge, M. Trotter, B. Buxton, S. Holden, Drug design by machine learning: support vector machines for pharmaceutical data analysis, Computers and Chemistry 26 (2001) 5–14.
[4] J. Cai, A. Dayanik, H. Yu, N. Hasan, T. Terauchi, W. Grundy, Classification of cancer tissue types by support vector machines using micro array gene expression data, in: Proceedings of the Eighth International Conference on Intelligent Systems for Molecular Biology (ISMB 2000), 2000.
[5] L. Cai, T. Hofmann, Hierarchical document categorization with support vector machines, in: Proceedings of the CIKM'04, 2004, pp. 78–87.
[6] I. El-Naqa, Y. Yang, M. Wernick, N. Galatsanos, M. Nishikawa, A support vector machine approach for detection of microcalcifications, IEEE Transactions on Medical Imaging 21 (2002) 1552–1563.
[7] G. Guo, S. Li, Content-based audio classification and retrieval by support vector machines, IEEE Transactions on Neural Networks 14 (2003) 209–215.
[8] T. Joachims, Transductive inference for text classification using support vector machines, in: Proceedings of the International Conference on Machine Learning (ICML), 1999.
[9] Kalatzis, I. Pappas, N. Piliouras, D. Cavouras, Support vector machines based analysis of brain SPECT images for determining cerebral abnormalities in asymptomatic diabetic patients, Medical Informatics and the Internet in Medicine 28 (2003) 221–230.
[10] K. Papik, B. Molnar, R. Schaefer, Z. Dombovari, Z. Tulassay, J. Feher, Application of neural networks in medicine – a review, Medical Science Monitor 4 (1998) 538–546.
[11] J. Wyatt, Nervous about artificial neural networks? Lancet 346 (1995) 1175–1177.
[12] C.J. Wyatt, D.G. Altman, Prognostic models: clinically useful or quickly forgotten? British Medical Journal 311 (1995) 1539–1541.
[13] G. Fung, S. Sandilya, R. Rao, Rule extraction from linear support vector machines, in: Proceedings of the Eleventh SIGKDD International Conference on Knowledge Discovery and Data Mining, 2005.
[14] M. Friedman, Explanation and scientific understanding, Journal of Philosophy 71 (1974) 5–19.
[15] J. Wallis, E.H. Shortliffe, Explanatory power for medical expert systems: studies in the representation of causal relationships for clinical consultation, Methods in Information in Medicine (1982).
[16] R. Ramberg, Construing and testing explanations in a complex domain, Computers in Human Behavior 12 (1996) 29–48.
[17] C. Lacave, F.J. Diez, A review of explanation methods for heuristic expert systems, Knowledge Engineering Review 19 (2004) 133–146.
[18] P. Cunningham, D. Doyle, J. Loughrey, An evaluation of the usefulness of case-based explanation, in: Proceedings of the Fifth International Conference on Case-Based Reasoning (ICCBR 2003), 2003, pp. 122–130.
[19] S. Gallant, Connectionist expert system, Communications of the ACM 31 (1988) 152–169.
[20] R.L. Ye, P.E. Johnson, The impact of explanation facilities on user acceptance of expert systems advice, MIS Quarterly (1995) 157–172.
[21] H.-J. Hu, R. Harrison, P.C. Tai, Y. Pan (Eds.), Current Methods for Protein Secondary-Structure Prediction Based on Support Vector Machines, John Wiley & Sons, Inc., 2007.
[22] Z. Chen, J. Li, L. Wei, A multiple kernel support vector machine scheme for feature selection and rule extraction from gene expression data of cancer tissue, Artificial Intelligence in Medicine 41 (2007) 161–175.
[23] R. Andrews, J. Diederich, A. Tickle, A survey and critique of techniques for extracting rules from trained artificial neural networks, Knowledge Based Systems 8 (1995) 373–389.
[24] N. Barakat, J. Diederich, Learning-based rule-extraction from support vector machines: performance on benchmark data sets, in: Proceedings of the Conference on Neuro-Computing and Evolving Intelligence, Knowledge Engineering and Discovery Research Institute (KEDRI), Auckland, New Zealand, 2004.
[25] H. Núñez, C. Angulo, A. Catala, Rule-extraction from support vector machines, in: Proceedings of the European Symposium on Artificial Neural Networks, 2002, pp. 107–112.
[26] H. Núñez, C. Angulo, A. Catala, Support vector machines with symbolic interpretation, in: Proceedings of the VII Brazilian Symposium on Neural networks (SBRn'02), 2002.
[27] H. Núñez, C. Angulo, A. Catala, Rule based learning systems from SVM and RBFNN, 2002.
[28] H. Núñez, C. Angulo, A. Catala, Rule-based learning systems for support vector machines, Neural Processing Letters 24 (2006) 1–18.
[29] Y. Zhang, H. Su, T. Jia, J. Chu, Rule extraction from trained support vector machines, in: Proceedings of the Advances in Knowledge Discovery and Data Mining: Ninth Pacific-Asia Conference PAKDD2005, 2005, pp. 61–70.
[30] X. Fu, C. Ongt, S. Keerthit, G. Hung, L. Goh, Extracting the knowledge embedded in support vector machines, in: Proceedings of the IEEE International Conference in Neural Networks, 2004, pp. 291–296.
[31] A.C. Chaves, M. Vellasco, R. Tanscheit, Fuzzy rule extraction from support vector machines, in: Proceedings of the Fifth International Conference on Hybrid Intelligent Systems, 2005.
[32] N. Barakat, J. Diederich, Learning-based rule extraction from support vector machines, in: Proceedings of the 14th International Conference on Computer Theory and Applications (ICCTA'2004), 2004.
[33] N. Barakat, J. Diederich, Eclectic rule-extraction from support vector machines, International Journal of Computational Intelligence 2 (2005) 59–62.

[34] N. Barakat, A.P. Bradley, Rule extraction from support vector machines: a sequential covering approach, IEEE Transactions on Knowledge and Data Engineering 19 (2007) 729–741.
[35] D. Torres, C. Rocco, Extracting trees from trained SVM models using a TREPAN based approach, in: Proceedings of the Fifth International Conference on Hybrid Intelligent Systems, 2005.
[36] D. Martens, B. Baesens, T.V. Gestel, J. Vanthienen, Comprehensible credit scoring models using rule extraction from support vector machines, European Journal of Operational Research (2006).
[37] D. Martens, B. Baesens, T.V. Gestel, Decompositional rule extraction from support vector machines by active learning, IEEE Transactions on Knowledge and Data Engineering 21 (2009) 177–190.
[38] J. He, H.-J. Hu, R. Harrison, P.C. Tai, Y. Pan, Rule generation for protein secondary structure prediction with support vector machines and decision tree, IEEE Transactions on Nanobioscience 5 (2006) 46–53.
[39] H. Lakany, B.A. Conway, Understanding intention of movement from electroencephalograms, Expert Systems 24 (2007) 295–304.
[40] A. Moschitti, D. Pighin, R. Basili, Semantic role labeling via tree kernel joint inference, in: Proceedings of the 10th Conference on Computational Natural Language Learning, New York, USA, 2006.
[41] 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 within the evidence framework, IEEE Transactions on Neural Networks 12 (2001) 809–821.
[42] Jayadeva, R. Khemchandani, S. Chandra, Twin support vector machines for pattern classification, IEEE Transactions on Pattern Analysis and Machine Intelligence 29 (2007) 905–910.
[43] B. Waske, J.A. Benediktsson, Fusion of support vector machines for classification of multisensor data, IEEE Transactions on Geoscience and Remote Sensing 45 (2007) 3858–3866.
[44] T. Joachims, A support vector method for multivariate performance measures, in: Proceedings of the International Conference on Machine Learning (ICML), 2005.
[45] G. Siolas, F. d'Alche-Buc, Support vector machines based on a semantic kernel for text categorization, in: Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks (IJCNN'00), 2000, p. 5205.
[46] K. Rohit, R. Mooney, Semi-supervised learning for semantic parsing using support vector machines, in: Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (NAACL/HLT-2007), Rochester, NY, 2007, pp. 81–84.
[47] D. Zhang, W.S. Lee, Web taxonomy integration using support vector machines, in: Proceedings of the 13th International Conference on World Wide Web, New York, NY, USA, 2004, pp. 472–481.
[48] R. Klinkenberg, T. Joachims, Detecting concept drift with support vector machines, in: Proceedings of the Seventeenth International Conference on Machine Learning, San Francisco, CA, 2000, pp. 487–494.
[49] D. Martens, J. Huysmans, R. Setiono, J. Vanthienen, B. Baesens, Rule extraction from support vector machines: an overview of issues and application in credit scoring, in: J. Diederich (Ed.), Rule Extraction from Support Vector Machines, Springer, Berlin/Heidelberg, 2008, pp. 33–63.
[50] C. Burges, A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery, Kluwer Academic Publishers, 1998.
[51] N. Cristianini, J.S. Taylor, An Introduction to Support Vector Machines and Other Kernel-based Learning Methods, Cambridge University Press, Cambridge, 2000.
[52] B. Schoelkopf, A.J. Smola, Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond, MIT Press, Cambridge, 2002.
[53] 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.
[54] J.R. Quinlan, C4.5: Programs for Machine Learning, Morgan Kaufmann, SanMateo, CA, 1993.
[55] L. Breiman, J. Friedman, R. Olshen, C. Stone, Classification and Regression Trees, Wadsworth and Brooks, Monterrey, CA, 1984.
[56] T. Mitchell, Machine Learning, McGraw-Hill, 1997.
[57] J. Fürnkranz, P.A. Flach, ROC 'n' rule learning – towards a better understanding of covering algorithms, Machine Learning 58 (2005) 39–77.
[58] V. Kecman, Learning and Soft Computing: Support Vector Machines, Neural Networks, and Fuzzy Logic Models, MIT Press, Cambridge, 2001.
[59] A. Ben-Hui, D. Horn, H. Sidgelmann, V. Vapnik, Support vector clustering, Machine Learning Research 2 (2001) 125–137.
[60] M. Craven, J. Shavlik, Extracting Tree-structured Representation of Trained Networks, Advances in Neural Information Processing Systems, MIT Press, Cambridge, 1996, pp. 24–30.
[61] M. Faifer, C. Janikow, K. Krawiec, Extracting fuzzy symbolic representation from artificial neural networks, in: Proceedings of the 18th International Conference of the North American Fuzzy Information Processing Society (NAFIPS), 1999.
[62] C. Giles, W. Omlin, Extraction, insertion, and refinement of symbolic rules in dynamically driven recurrent network, Connection Science 5 (1993) 307–328.
[63] M. Golea, On the complexity of rule extraction from neural networks and network querying, in: Proceedings of the Rule Extraction from Trained Artificial Neural Networks Workshop, Society for the Study of Artificial Intelligence and Simulation of Behavior Workshop Series (AISB'96), 1996, pp. 51–59.
[64] H. Khuu, H.K. Lee, J.-L. Tsai, Machine learning with neural networks and support vector machines, University of Wisconsin, 2004.
[65] N. Barakat, A.P. Bradley, Rule extraction from support vector machines: measuring the explanation capability using the area under the ROC curve, in: Proceedings of the 18th International Conference on Pattern Recognition (ICPR'06), 2006, pp. 812–815.
[66] F. Provost, T. Fawcett, R. Kohavi, The case against accuracy estimation for comparing induction algorithms, in: Proceedings of the International Conference on Machine Learning, 1998, pp. 445–453.
[67] A.P. Bradley, The use of the area under the ROC curve in the evaluation of machine learning algorithms, Pattern Recognition 30 (1997) 1145–1159.
[68] Z.H. Zhou, Rule extraction: using neural networks or for neural networks, Journal of Computer Science and Technology 19 (2004) 249–253.

Nahla Barakat is an Associate Professor in Computer Science, and is currently with the German University of Technology in Oman. She holds a Ph.D. degree in Computer Science from the University of Queensland, Australia. In addition to academia, she has more than 10 years of industry experience in the area of IT in a multinational environment. Her current research interests include machine learning, medical data mining and data management.

Andrew P. Bradley is an Associate Professor in Biomedical Engineering at The University of Queensland, Brisbane, Australia. He received his Ph.D. from The University of Queensland in 1996 and since then has held various research positions in Australia, the United Kingdom and Canada. His research interests are currently focused on biomedical applications of pattern recognition in signal and image analysis. He is a Chartered Electrical Engineer and Senior Member of the IEEE.