Rule Extraction from Support Vector Machines: A Geometric Approach

Lu Ren and Artur Garcez
Department of Computing, City University London
cf529@soi.city.ac.uk, aag@soi.city.ac.uk

***Abstract*— This paper presents a new approach to *rule extraction* from Support Vector Machines. SVMs have been applied successfully in many areas with excellent generalization results; rule extraction can offer *explanation capability* to SVMs. We propose to approximate the SVM classification boundary through querying followed by clustering, searching and then to extract rules by solving an optimization problem. Theoretical proof and experimental results then indicate that the rules can be used to validate the SVM results, since maximum fidelity with high accuracy can be achieved.**

I. Introduction

In recent years, Support Vector Machines (SVMs) have been utilised in many applications offering excellent generalization results. In many cases, however, developers prefer not to use SVMs because of their inability to explain how the results have been obtained. For example, if used for stock predictions, an SVM can provide users with a mechanism for forecasting, but the knowledge associated with the predictions may be incomprehensible; users may have to trust the prediction results while unable to validate the rules of the data.

The area of *rule extraction* addresses the above problem. By extracting rules from SVMs, we can explain the reasoning process and validate the learning system. Rule extraction helps, therefore, to integrate the symbolic and connectionist approaches to AI, offering ways of combining the statistical nature of learning with the logical nature of reasoning.

Since the early 1990s, various algorithms to extract rules from trained neural networks have been proposed, notably [4], [2], [17], [11], [14], [12], [5]. Some of these search for rules by decomposing the networks and extracting rules for each unit, and some extract rules directly from the input-output values of the networks, thus treating them as black-boxes. Recently, SVMs started to be considered for rule extraction because of their excellent generalization capability. Angulo et al [19] used support vectors and prototypes to draw regions indicating an equation rule or interval rule. Barakat and Diererich [15] used support vectors to construct synthetic data, feed the data into a decision tree learner, and extract rules. Fung et al [6] proposed an algorithm that approximates linear SVM hyperplanes by a set of rules.

In our opinion, a satisfactory extraction method, striking a balance between the need for correctness and efficiency, is still lacking. Most of the extraction methods are designed for specific architectures and training sets, or are affected by the tradeoff between efficiency and rule accuracy. In this paper, we tackle both of these issues by proposing a new any-time rule extraction algorithm, which uses the SVM as an oracle (black-box) and synthetic data for querying and rule extraction, thus making fewer assumptions about the training process and the SVM training data. The algorithm is not restricted to a specific SVM classifier such as the linear classifier considered in [6], neither does it depend on the availability of specific training sets for rule extraction. Instead, it seeks to capture the information encoded in the geometry of the SVM by approximating the region separated by the SVM classification boundary through querying [12] and searching, and then extracting rules by solving an optimization problem, which we describe in detail in the sequel. We also prove the soundness and completeness of our approach, and run experiments and compare our approach with other extraction methods. We examined rule accuracy, fidelity and comprehensibility in two applications: the iris flower dataset and the breast cancer-wisconsin dataset. The results indicate the correctness of our approach through maximum fidelity.

The paper is organised as follows: Section 2 gives a brief introduction to SVMs. Section 3 describes our new extraction algorithm. Section 4 presents the proofs. Section 5 contains the experimental results, and Section 6 concludes and discusses directions for future work.

II. Support Vector Machines

We consider the problem of classifying *n* points in the *m*-dimensional input space R^m. Consider the training data set $\{(\mathbf{x}_i, y_i)\}$, $i = 1, ..., n$, $y_i \in \{1, -1\}$ and $\mathbf{x}_i \in R^m$. In the case of linear SVMs the decision function is a separating hyperplane $f(x) = sign(w \cdot x + b)$. The optimal classification hyperplane that maximizes the distance between class $A_+ = 1$ and $A_- = -1$ can be found by *minimizing* $1/2\|w\|^2$ *subject to* $y_i(w \cdot \mathbf{x}_i + b) \geq 1$.

The Lagrangian J below has been introduced to solve this problem: $J = \frac{1}{2}w^T w - \Sigma_{i=1}^{n}\alpha_i(y_i(w \cdot \mathbf{x}_i + b) - 1)$, where $\alpha_i \geq 0$ is known as the lagrange multiplier. With respect to w and b, minimizing J leads to $w = \Sigma_{i=1}^{sv}\alpha_i y_i \mathbf{x}_i$ and $\Sigma_{i=1}^{n}\alpha_i y_i = 0$ where sv is the number of support vectors [21]. By making some substitutions, we arrive at the hyperplane decision function $f(x) = sign(\Sigma_{i=1}^{sv}\alpha_i y_i \langle \mathbf{x}_i \cdot \mathbf{x}_j \rangle + b)$, $j = 1, ..., n$, where $\langle\rangle$ denotes inner product.

For nonlinear classification, the SVM has to map the data points into a high dimension feature space H in which the data can be linearly separated. Let $\Phi : R^m \to H$. By using kernel functions $K(x, x') = \langle \Phi(x) \cdot \Phi(x') \rangle$, all the necessary operations in the input space may be carried through, and the decision function can be $f(x) = sign(\Sigma_{i=1}^{sv} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}_j) + b)$. For more details on SVMs, see [21].

III. Geometric SVM Rule Extraction

Most rule extraction algorithms suffer from a lack of generality, a balance between correctness and fidelity, or both. In this section, we present a novel rule extraction algorithm called *Geometric and Oracle-Based Support Vector Machines Rule Extraction* (GSVMORC), which is designed to alleviate these limitations. GSVMORC utilizes the points on the SVM classification boundary and synthetic training instances to construct a set of optimized hypercube rules. The area covered by those rules is maximized and approximates the *area of interest*. The definitions of the *hypercube rule* and the *area of interest* are given as follows.

Definition 3.1: Hypercube Rule: It is said that an m-dimensional hypercube H characterizes a rule if every point in the scope of H falls into the same class classified by the SVM network. More precisely:

$$H = \left\{ \mathbf{x}_i \middle| \bigwedge l_j \leq x_{ij} \leq u_j \rightarrow A_k, 1 \leq j \leq m \right\}$$

where $\mathbf{x}_i = [x_{i1}, ..., x_{im}]$, l_j and u_j are the upper and lower bounds on H. A_k indicates a class label, and m is the dimension of the input space.

Definition 3.2: Area of Interest. This is the whole region covered by a class A_k in the input space.

$$I(A_k) = \left\{ \mathbf{x}_i \;\middle|\; the\ class\ of\ \mathbf{x}_i \ = A_k,\ l_j \leq x_{ij} \leq u_j, \right\},$$

where $1 \leq j \leq m, i \geq 1$, $\mathbf{x}_i$ is an m-dimensional input vector. u_j and l_j are the upper and lower bounds of x_{ij}. x_{ij} refers to the j^{th} dimension of $\mathbf{x}_i$. There is no vector $\mathbf{x}_i \in I(A_k)$ such that the classification of $\mathbf{x}_i$ equals the other class rather than A_k.

The aim of GSVMORC is to use the classification boundary and synthetic training instances to extract the hypercube rules without considering the inner structure and the support vectors of the SVM network. It treats the SVMs as *oracles*, and makes fewer assumptions about the architecture and training process, hopefully being applicable to other non-symbolic learning methods. All we assume is that an SVM is given which we can query and find the classification it gives for input vectors $\mathbf{x}_i$. After querying, a clustering process is imposed on those inputs $\mathbf{x}_i$ with the same y_i, in order to group them into a set of clusters. Then, by means of a binary search algorithm, we look for the points $\mathbf{P}$ that lie on the SVM classification boundaries. Subsequently, an initial optimal rule set can be extracted for the points in $\mathbf{P}$ and synthetic training instance set by solving an optimization problem whereby we attempt to find the largest consistent hypercubes in the input space. Finally, several post-processing measures are applied to this initial rule set in order to derive (a relatively small number of) generalized rules from. In what follows, we explain each of the above steps of our extraction algorithm.

Querying. The generality of GSVMORC is because it generates a subset of synthetic training inputs to query the SVMs, which then returns back the class labels of the inputs. We use a random data generator to produce a large amount of inputs. Only those inputs locate in the range of the input space, and their values of a density estimator $M(\mathbf{x})$ are larger than a user-defined constraint $c1$ are retained. Unlike TREPEN [13] which uses the *kernel density estimates of individual features*, GSVMORC uses *multivariate kernel density estimates* which takes into account the *relations within features*. $M(\mathbf{x})$ used by GSVMORC models the probability density function for inputs $\mathbf{x}$ as:

$$M(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{\prod_{j=1}^{m} h_j} [\frac{1}{(\sqrt{2\pi})^m} e^{-\frac{1}{2}(\| \frac{\mathbf{x} - \mathbf{x}_i}{h} \|)}]$$

where $\mathbf{X}_i$ are the given training samples, $1 \leq i \leq N$. $h = [h_1, h_2, ...h_m]$ is a vector of bandwidth such that $h = [\frac{4}{(m+2)N}]^{\frac{1}{m+4}} \sigma$ where σ is standard deviation of the training samples. Figure 1 contains the input generation algorithm.

DRAWINPUTS

Input: a constraint $c1$, the lower and upper bounds of the input space L and U, a random data generator $g(\mathbf{x})$, a density estimator $M(\mathbf{x})$ and the number of iteration
times T
Output: the input set $\mathbf{x} = \{\mathbf{x}_i : i \geq 1\}$

```
(1)  Initialize the input set $\mathbf{x}$
(2)      Generate $T$ random data $\{\mathbf{x}_i : 1 \leq i \leq T\}$ based on $g(\mathbf{x})$
(3)  for each iteration $i < T$ do
(4)      If $L \leq \mathbf{x}_i \leq U$ then
(5)          Calculate $M(\mathbf{x}_i)$
(6)          if $M(\mathbf{x}_i) > c1$ then
(7)              $\mathbf{x} := \mathbf{x} \bigcup \mathbf{x}_i$
```

Fig. 1. The DRAWINPUTS function: call a random data generator $g(\mathbf{x})$ to create uniformly distributed data; use a density estimator $M(\mathbf{x})$ to reserve those data whose probabilities are larger than an arbitrary number $c1$. Meanwhile, the outputs should lie between L and U

After obtaining those synthetic inputs $\{\mathbf{x}_i, i \geq 1\}$, we treat them as the inputs of the SVMs; the SVMs are considered as *oracles* [1]. Suppose we have an SVM computing function $f(\mathbf{x})$. For each input $\mathbf{x}_i$, we feed $\mathbf{x}_i$ into the SVM and get the corresponding output $y_i = f(\mathbf{x}_i)$. The instances $\{(\mathbf{x}_i, y_i), i \geq 1\}$ are created. Note that the distance d from $\mathbf{x}_i$ to the separating hyperplane is also obtained when SVM answer the queries from $\mathbf{x}_i$.

The last step in querying is called SELECTINSTANCE. GSVMORC defines a factor n so that it is much more flexible in its ability to choose differing sizes of instances. In order to select the instances with much higher $M(\mathbf{x})$, those $\mathbf{x}_i$ $(i \geq 1)$

[1]Notice that the SVM is considered as a *black box*. All we need to know are key input-output patterns, rather than the inner structure. This complies with Thrun's desideratum for a general rule extraction method of *no training requirement*[4]. The querying process makes our approach independent of any special training data, neither does it make any assumption about the network's structure. It can be applied to any SVM classifier, regardless of the algorithms used to construct the classifier, including Sequential Minimal Optimization (SMO) [10] and DAGs-SVM [9].

with the same class label are sorted first. Suppose that there are CN classes involved in the classification problem and that the number of $\mathbf{x}_i$ for each class is $n(G)$. In this case, we choose the first n/CN instances, or the whole group of instances if $n(G) < n/CN$, to build up the synthetic training instance set S.

One key issue in querying is how to know the classes which are needed to be classified in a general way. GSVMORC still generates a large amount random data $\mathbf{Z} = \{\mathbf{z}_1, ..., \mathbf{z}_h\}$[2], which are uniformly distributed in the input space. These data $\mathbf{Z}$ are then input into the SVM network to acquire class labels for them. For large h, we believe that the data in $\mathbf{Z}$ are able to spread throughout the input space. Therefore, a set of classes ($A = \{A_k | 1 \leq k \leq CN, CN$ is the total number of classes $\}$) can be obtained after filtering the duplicates.

Clustering. Since there must be a classification boundary between different classes, we can find the points lying on the classification boundary between pairs of data for different classes. However, for a large number of training data, if we search each pair of data for different classes, this may lead to high complexity. Hence, we use clustering to create a balance between the complexity and prediction accuracy. A cluster C can be defined as a subset of training data $S = \{(\mathbf{x}_i, y_i)\}$, with the same class y_i.

We use hierarchical clustering on S. It starts by considering each individual point as a cluster, and it merges the clusters by measuring the distance between two clusters of data which have the same class labels. Because the mergence of the clusters is relevant only to those training instances that have the equivalent class, just the inputs $\mathbf{x}_i$ are involved in the distance calculation. Our approach uses one of the following linkage functions: *Single linkage*, uses the smallest distance between data $\mathbf{x}_i^r$ and $\mathbf{x}_j^s$ in the two clusters r and s. If the size of r and s are n_r and n_s then $d(r,s) = min(dist(\mathbf{x}_i^r, \mathbf{x}_j^s)), i \in (1, ..., n_r), j \in (1, ..., n_s)$; *Complete linkage*, uses the largest distance between data $\mathbf{x}_i^r$ and $\mathbf{x}_j^s$ in the two clusters r and s such that $d(r,s) = max(dist(\mathbf{x}_i^r, \mathbf{x}_j^s))$.

In order to reduce the randomness of the number of clusters, a *stopping criterion* is defined for the clustering process. Given q clusters $\{r_h,\ h = 1, 2, ..., q\}$, the classes of the clusters are identical, and the number of data in each cluster r_h is n^{r_h}. It is obvious that the mean and variance of each cluster relate to the data $\mathbf{x}_i, 1 \leq i \leq n^{r_h}$ inside this cluster. Hence, the mean m^{r_h} of each cluster r_h is:

$$m^{r_h} = \frac{1}{n^{r_h}} \sum_{i=1}^{n^{r_h}} \mathbf{x}_i$$

and the variance s^{r_h} is

$$s^{r_h} = \frac{1}{n^r} \sum_{i=1}^{n^{r_h}} (\mathbf{x}_i - m^{r_h})^2$$

[2]h is an arbitrary large integer.

Hence, the intra-cluster deviation is defined as follows:

$$s^{intra} = \sqrt{\sum_{h=1}^{q} (s^{r_h} * p(r_h))} \tag{1}$$

where $p(r_h) = \frac{n^{r_h}}{\sum_{h=1}^{q} n^{r_h}}$.

And the inter-cluster mean and deviation are specified in Equation 2.

$$m^{inter} = \sum_{h=1}^{q} (m^{r_h} * p(r_h)) \tag{2a}$$

$$s^{inter} = \sqrt{\sum_{h=1}^{q} [(m^{r_h} - m^{inter})^2 * p(r_h)]} \tag{2b}$$

Definition 3.3: The stopping criterion D is the rate between s^{intra} and s^{inter}.

If $\frac{s^{intra}}{s^{inter}} > \epsilon$, then GSVMORC will stop merging the data further. Note that ϵ is a user-defined parameter.

Searching. The searching step searches for and locates the points on the decision boundaries. Given clusters $P_1, P_2, ...P_a$ which fall into class A_+, and clusters $N_1, N_2, ...N_b$ which fall into class A_-, we use Zhang and Liu's measure [24] to automatically look for the points on an SVM's decision boundary[3].

We consider all pairs (p, n) s.t. $p \in P_j (1 \leq j \leq a)$ and $n \in N_k (1 \leq k \leq b)$. For each p, we find a corresponding point n whose distance to p is minimum. And for each n, we find a corresponding point p whose distance to n is minimum. As described in *querying* section, the distance from any point to the SVM hyperplane is one of the outputs by querying the SVM network. Let d_1 represent the distance from p to the hyperplane and d_2 represent the same for n. In order to find the point lying on the hyperplane, a binary search procedure is performed on (p, n). In other words, if $|d_1 - d_2| > \varepsilon$, the mid-point q between p and n is chosen. The SVM network classifies q and computes the distance between q and the hyperplane. If the class of q equals that of p, then p is replaced by q; otherwise, n is replaced by q. The process carries on until $|d_1 - d_2| < \varepsilon$ is achieved, where ε denotes an arbitrary small number.

Extracting. The main idea of our rule extraction approach is to find a set of optimal rules that 1) covers the maximum area of the *area of interest* and 2) covers the largest cardinality of synthetic instances at the same time.

Suppose that there are a set of points X lying on the SVM decision boundary, where X is the result of *searching*, a set of synthetic training instances S generated from *querying* for classes $A = \{A_p, 1 \leq p \leq CN\}$, and the SVM function $f(\mathbf{x})$.

To realize the first goal of the rule extraction algorithm, we try to solve the following optimization problem:

$$maximize \qquad \prod_{i=1}^{m} (x_i - \mathbf{x}_{0_i}) \tag{3a}$$

[3]Notice that for simplicity we have been considering $P = 2$ classes, but our extraction algorithm is applicable to any number of classes.

$$subject\ to \qquad l \leq \mathbf{x} \leq u \tag{3b}$$

$$\int_l^u (f(\mathbf{x}) - A_p) d\mathbf{x} = 0 \tag{3c}$$

where $\mathbf{x}_{0_i}$ denotes the i^{th} element of vector $\mathbf{x}_0 \in X$ ($\mathbf{x}_0$ indicates a starting point), x_i is the i^{th} element of $\mathbf{x}$, l and u are the m-dimensional vectors giving lower and upper bounds to this optimization problem.

The objective function (Equation 3a) aims to maximize the volume of the hypercube that a rule covers, and it has two constraints. One is a bound constraint to limit the optimal $\mathbf{x}^*$ in a given area, while the other is a nonlinear constraint that is used to exclude the points that have different class labels.

The values of l and u in Equation 3b can be calculated based on the lower and upper bounds of the input space. For example (see Figure 2(a)), suppose the scope of the input space is $[L_1, L_2] \leq \mathbf{x} \leq [U_1, U_2]$, and $\mathbf{x}_0 = [\mathbf{x}_{0_1}, \mathbf{x}_{0_2}]$ is a point lying on the SVM boundary. Note that when we change $\triangle\mathbf{x}_{0_1}$ on $\mathbf{x}_{0_1}$ or $-\triangle\mathbf{x}_{0_2}$ on $\mathbf{x}_{0_2}$ ($\triangle\mathbf{x}_{0_1}, \triangle\mathbf{x}_{0_2} \geq 0$) the SVM classification on $\mathbf{x}_0$ is class A_p. Hence, it is reasonable to assume that an optimal point can be found and that a rule for class A_p in a rectangle between points $\mathbf{x}_0$ and $[U_1, L_2]$ can be constructed. Here, $[U_1, L_2]$ is defined as an *orientation* for $\mathbf{x}_0$. l and u are then narrowed down to $l = [\mathbf{x}_{0_1}, L_2]$ and $u = [U_1, \mathbf{x}_{0_2}]$.

If more than one *orientation* is found for class A_p, then the principal orientations have to be selected. Let $\mathbf{v} = [v_1, v_2]$ stand for an orientation for $\mathbf{x}_0$. In the above example, $\mathbf{v} = [U_1, L_2]$. Selecting the orientations for $\mathbf{x}_0$ involves deciding how to compute the significance of each orientation. Equation 4 shows conditional probability estimation that GSVMORC uses to determine the significance of each orientation. The estimation represents the probability distribution of $\mathbf{x}$ lying in the area between $\mathbf{v}$ and $\mathbf{x}_0$, given class A_p. In Equation 4, $P(\min(\mathbf{v}, \mathbf{x}_0) \leq \mathbf{x} \leq \max(\mathbf{v}, \mathbf{x}_0) \bigcap class = A_p)$ indicates the probability of $\mathbf{x}$ falling into the area between $\mathbf{v}$ as well as belonging to class A_p, and $P(class = A_p)$ is the possibility that the classification of $\mathbf{x}$ equals A_p. Assume that the distribution of synthetic training examples is similar to that of the problem domain. Hence, the probabilities $P(\min(\mathbf{v}, \mathbf{x}_0) \leq \mathbf{x} \leq \max(\mathbf{v}, \mathbf{x}_0) \bigcap class = A_p)$ and $P(class = A_p)$ could be worked out from the synthetic data set. The value of $P(\min(\mathbf{v}, \mathbf{x}_0) \leq \mathbf{x} \leq \max(\mathbf{v}, \mathbf{x}_0) | class = A_p)$ is then calculated by dividing $P(\min(\mathbf{v}, \mathbf{x}_0) \leq \mathbf{x} \leq \max(\mathbf{v}, \mathbf{x}_0) \bigcap class = A_p)$ by $P(class = A_p)$. The end result is that those orientations that have the maximum probabilities are selected as the principal orientations on $\mathbf{x}$.

$$P(\min(\mathbf{v}, \mathbf{x}_0) \leq \mathbf{x} \leq \max(\mathbf{v}, \mathbf{x}_0) | class = A_p)$$

$$= \frac{P(\min(\mathbf{v}, \mathbf{x}_0) \leq \mathbf{x} \leq \max(\mathbf{v}, \mathbf{x}_0) \bigcap class = A_p)}{P(class = A_p)} \tag{4}$$

As presented in Equation 3c, the nonlinear constraint is a multi-dimension integral on a linear/nonlinear function. GSVMORC uses a quasi-Monte Carlo method [22] to approximate the integration because it is a superior method with many advantages such as improved convergence and more uniformity. Therefore, the hypercube H is considered to be composed of the points that are uniformly distributed as:

$$\frac{1}{n}\sum_{i=1}^{n} |f(\mathbf{a}_i) - A_p| \approx \int_l^u |f(\mathbf{x}) - A_p| \, d\mathbf{x}$$

where $\mathbf{a}_i$ is a low-discrepancy sequence inside the hypercube $[l, u]$, where $1 \leq i \leq n$, and n here means the number of points selected for approximation in the H. The estimation error then becomes,

$$\epsilon = \left| \int_l^u |f(\mathbf{x}) - A_p| \, dx - \frac{1}{n}\sum_{i=1}^{n} |f(\mathbf{a}_i) - A_p| \right|$$

From the above, it can be shown that the larger n is, the closer the approximation approaches the integral. It is clear that the complexity increases with the rise of n. Therefore, in order to strike a balance between error estimation, fidelity, accuracy prediction and complexity, a proper n has to be chosen. In the cross-validation experiments, we found $n = 1000$ as a suitable number for our benchmark datasets.

With this, the standard *pattern search algorithm* is applied to obtain a solution $\mathbf{x}^*$ to the optimization problem. Charles and Dennis analyze the generalization of the pattern search by evaluating the objective function [1]. After obtaining the optimal point $\mathbf{x}^*$, together with the starting point $\mathbf{x}_0$, the antecedents of a rule can be constructed by picking the minimum and maximum values of $\mathbf{x}^*$ and $\mathbf{x}_0$, as shown in Figure 3. Figure 2(a) gives an example of a hypercube rule with the starting point $\mathbf{x}_0$.

Finally, to find the set of rules covering all the synthetic training instances, Equation 3 is used again, and in it, $\mathbf{x}_0$ is replaced with $s \in S$ (see Figure 2(b)). The process is the same, which ensures the extracted rules cover most of the synthetic training instances as well as the maximum area of the *area of interest*.

Figure 3 summaries the rule extraction algorithm and its associated rule generation algorithm, as discussed above.

The rule set obtained from *extracting* may contain overlapping rules, for which a set of post-processing measures in the next section are employed to solve this problem.

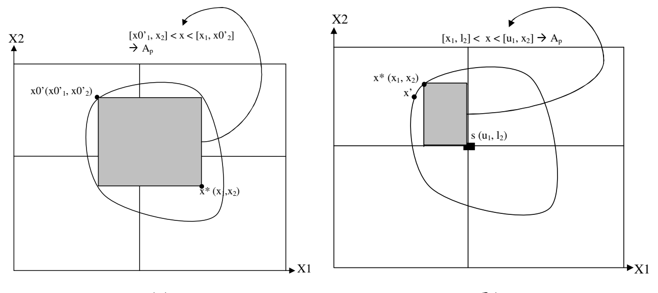


Fig. 2. *left:* extracting the rule from the starting point $[\mathbf{x}_{0_1}, \mathbf{x}_{0_2}]$ which is obtained from *Searching*; *right:* extracting the rule to cover the training data and approximate the area that A_p covers, the starting point is $[s_1, s_2]$.

Extracting

Input: A set of points X on the SVM boundary obtained from the *searching* step; a set of training data S obtained from *querying* step and the class label A_p.
Output: A set of rules $R = \{r\}$,
where $r = \bigwedge l_i \le x_i \le u_i \rightarrow A_p$, $1 \le i \le m$

```
(1) for each $t \in X$
(2)     Construct the lower and upper bound $l$ and $u$
        by finding the orientations of $\mathbf{x}$
(3)     Apply pattern search algorithm [1]with $\mathbf{x}_0 = t$ to
        obtain $\mathbf{x}^*$
(4)     Call rule generation algorithm with parameters $\mathbf{x}^*$
        and $\mathbf{x}_0$ to construct a rule $r$ and make $R := \cup r$
(5) for each $s \in S$ and its corresponding $t \in X$
(6)     Construct the lower and upper bound $l$ and $u$
        by finding the orientations of $s$
(7)     Apply pattern search algorithm [1] with $\mathbf{x}_0 = s$ to
        obtain $\mathbf{x}^{*'}$.
(8)     Call rule generation algorithm with parameters $\mathbf{x}^{*'}$ and $s$
        to construct a rule $r'$ and make $R := R \cup r'$
```

Rule Generation Algorithm

Input: m-dimensional points $\mathbf{x}^*$ and $\mathbf{x}_0$
Output: a rule r

```
(1) Let lower bound $l = [min(\mathbf{x}_1^*, \mathbf{x}_{0_1}), ..., min(\mathbf{x}_m^*, \mathbf{x}_{0_m})]$
(2) Let upper bound $u = [max(\mathbf{x}_1^*, \mathbf{x}_{0_1}), ..., max(\mathbf{x}_m^*, \mathbf{x}_{0_m})]$
(3)     Generate $r = \bigwedge l_i \le x_i \le u_i \rightarrow A_p$, $1 \le i \le m$
```

Fig. 3. Rule extraction algorithm

Post-Processing. The purposes of these post-processing measures are to detect generalized rules, to prune rules with high error estimation and to construct non-overlapping rules with high coverage rate.

The notions of *non-overlapping* and *coverage rate* are defined as follows.

Definition 3.4: Non-overlapping Rule: Given two rules, $r_1 = \bigwedge a_i \le x_i \le b_i \rightarrow A_p$ and $r_2 = \bigwedge c_i \le x_i \le d_i \rightarrow A_p$, r_1 and r_2 are said to be non-overlapping iff $b_i \le c_i$ or $a_i \ge d_i$, for any i, $1 \le i \le m$.

Definition 3.5: Coverage rate is the rate between the number of testing data that are predicted correctly by a rule and the entire testing data.

1) Rule Extending: Given that the input space of a problem domain is from $[L_1, ..., L_m]$ to $[U_1, ..., U_m]$ and that of a rule is $r = [l_1, ..., l_m] \le \mathbf{x} \le [u_1, ...u_m] \rightarrow A_p$, the rule-extending step attempts to extend r into a larger scope. At the same time, the new rule r' still satisfies the constraint that the area covered by r' belongs to the same class. To exhaustively find all the potential rules in an extended scope, a topology is used to achieve this.

Let the original value of r be 0 and the new value of a rule be 1. For example, if the 1^{st} dimension of r is extended to L, then r becomes $l' = [L_1, l_2, ..., l_m] \le \mathbf{x} \le u' = [u_1, ..., u_m] \rightarrow A_p$. Hence, the new value $[L_1, u_1]$ on dimension one is regarded as 1. The definition of topology is defined as follows.

Definition 3.6: Topology: An arrangement in which each element means that the value of every dimension is mapped to 0 or 1 according to the above regulations.

$$\begin{tikzcd}
 & 111 & \\
110 \arrow[ur] & 101 \arrow[u] & 011 \arrow[ul] \\
100 \arrow[u] \arrow[ur] & 010 \arrow[ul] \arrow[ur] & 001 \arrow[u] \arrow[ul] \\
 & 000 \arrow[ul] \arrow[u] \arrow[ur] &
\end{tikzcd}$$

Fig. 4. Ordering on extending to the edge of the problem domain.

Figure 4 shows an example of a topology where the dimension of the input space is 3.

We assume that the rule r initially constructs a rule set R. The function has $m - 1$ iterations, and for each iteration, r_j in R is picked, and every dimension of r_j is extended, where $1 \le j \le n$ and n is the number of rules in R. At the first iteration, there is only one rule in R that is $r_j = r$, $j = 1$. Subsequently, each dimension i is extended to L_i, and the new value $[L_i, u_i]$ is verified if it satisfies the constraint $\frac{1}{n}\sum_{i=1}^{n}(|f(\mathbf{x}_i) - A_p|) = 0$. Next, for the same dimension, r is then extended to U_i, and a similar verification is performed on this new value. If there is any extension on the value of the i^{th} dimension of r, the new rule r' is kept for the next iteration. After going through each dimension of r_j, all the new rules, r', are put together for a new rule set R.

Finally, if the value of the i^{th} dimension equals the scale of the input space, which is believed to be applicable throughout the total range of the i^{th} dimension, GSVMORC then filters this dimension from the antecedents of the rule.

The complexity becomes exponential if the algorithm goes through every element in the topology. It increases with the rise of the dimensionality of the input space and even grows to be intractable in the worst case. Hence, in practice, an optimizing measure known as the *cracking of topology* has been adopted.

Firstly, the definition of a *clash* of the topology is given. A *clash* is an occurrence when the new region of a rule consists of the points for another class. The rule can be represented as an element in the topology.

When a *clash* is identified on a certain element, the rule-extending process would not continue on with the remaining elements that have connections with the element that has the *clash*. This is called *cracking of topology*.

Consider a three-dimensional problem. The antecedents of an initial rule are interpreted as 000. It is then easy to make a structure in the order of Figure 4. Given such an ordering, some conclusions can be drawn. If an element in Figure 4 deviates from $\frac{1}{n}\sum_{i=1}^{n}|f(\mathbf{a}_i) - A_p| = 0$, then a clash would be detected, which indicates that no other element along the ordering of this element would satisfy $\frac{1}{n}\sum_{i=1}^{n}|f(\mathbf{a}_i) - A_p| = 0$.

2) Rule Pruning: The *rule pruning* stage aims to prune those rules that have a relatively large estimated error. GSVMORC uses a *t-test* to analyze the null hypothesis that the mean of the estimated value and the expected value of the integral of a rule r are equal, that is the mean of the estimated value equals 0.

As GSVMORC uses quasi-Monte Carlo method to approximate the integration function in Equation 3c, there is a potential error between the approximation value and the integral. Most existing studies use the Koksma-Hlawka inequality [3] to state the limit of the integration error.

$$\left| \frac{1}{n} \sum_{i=1}^{n} |f(\mathbf{x}_i) - A_p| - \int_L^U |f(\mathbf{x}) - A_p| \right| \leq V(f) D_N^*$$

However, it is usually difficult to calculate the total variation $V(f)$, which makes it problematic to estimate error through Koksma-Hlawka inequality.

Morohosi and Fushimi [7] introduce a statistical method for quasi-Monte Carlo error estimation. The rule pruning of GSVMORC is based on their method. The general scheme of the method is as follows.

Suppose a rule r with an area ranging from l to u, M data sets $\{\mathbf{x}_i^{(j)}\}_{i=1}^{n}$, where $j = 1, ..., M$, $l \leq \mathbf{x}_i^{(j)} \leq u$ and $\{\mathbf{x}_i^{(j)}\}_{i=1}^{n}$ is a set of pseudorandom data. For each data set, the value of Equation 5 is computed.

$$S^{(j)} = \frac{1}{n} \sum_{i=1}^{n} |f(\mathbf{x}_i) - A_P|, j = 1, ..., M \tag{5}$$

The estimate of the mean $\hat{I}$ is calculated by

$$\hat{I} = \frac{1}{M} \sum_{j=1}^{M} S^{(j)} \tag{6}$$

so that the error of the integral is estimated using the variance of the evaluated values.

$$\hat{\sigma^2} = \frac{1}{M(M-1)} \sum_{j=1}^{M} (S^{(j)} - \hat{I})^2 \tag{7}$$

Hence, the *t-test* turns out to be:

$$t = \frac{\hat{I}}{\frac{\hat{\sigma}}{\sqrt{M}}} \tag{8}$$

GSVMORC sets a significance level to specify how close the approximation value is to the expected value 0. If t is larger than the standard value at the significance level, the rule is rejected. Otherwise it is accepted.

Those rules rejecting the null hypothesis are removed. Therefore, GSVMORC's pruning is able to ensure that GSVMORC approximates the behavior of the SVM network.

3) Non-overlapping Rule Construction: As mentioned in the *extracting* section, there could exist overlapping rules. To remove the intersections between rules and improve the comprehensibility of rules, the characteristics of non-overlapping rules is identified, that is at least one dimension of each of two rules do not intersect with each other. For example, let r_1 be $[a_1, .., a_m] \leq \mathbf{x} \leq [b_1, ..., b_m] \rightarrow A_p$ and r_2 be $[c_1, ...c_m] \leq \mathbf{x} \leq [d_1, .., d_m] \rightarrow A_p$. If $a_i \leq c_i \leq b_i \leq d_i$, $1 \leq i \leq m$, then the overlap of r_1 and r_2 is $\{[c_1, .., c_m] \leq \mathbf{x} \leq [b_1, ..b_m]\}$. Suppose r_2 does not change and r_1 has to be divided. For each dimension i, a non-overlapping rule can be constructed in three steps:

Part 1. Keep the original value $a_j \leq x_j \leq b_j$ of r_1 for those dimensions $j < i$.

Part 2. Use the non-overlapping value $a_i \leq x_i \leq c_i$ for the dimension $j = i$.

Part 3. Use the overlapping values $c_j \leq x_j \leq b_j$ instead of the original values of r_1 for those dimensions $j > i$.

As a result, the non-overlapping rule is the concatenation of these three parts.

For example, given two rules $r_1 = \{[1, 4, 3] \leq \mathbf{x} \leq [4, 7, 6]\} \rightarrow A_p$ and $r_2 = \{[2, 5, 4] \leq \mathbf{x} \leq [5, 9, 8]\} \rightarrow A_p$, the intersection part of these rules is $[2, 5, 4] \leq \mathbf{x} \leq [5, 7, 6]$. If $i = 2$ and r_2 remains, then r_1 should be split into three parts:

part 1. $1 \leq x_1 \leq 4$;
part 2. $4 \leq x_2 \leq 5$;
part 3. $4 \leq x_3 \leq 6$.

Then the non-overlapping rule is $[1, 4, 4] \leq \mathbf{x} \leq [4, 5, 6] \rightarrow A_p$.

4) Rule Selection: The last step of post-processing is *rule selection*. This discards those rules with zero coverage rate.

The aim of this step is to extract those rules with extensive information. In GSVMORC, this means that the rules predicting no data in our experiments are removed. Note that the selection does not change the predictive behavior of GSVMORC, it simply deletes extraneous rules

IV. Proof of Algorithm

We now prove that the proposed algorithm is quasi-soundness and quasi-completeness.

Theorem 1: Each rule $R : r \rightarrow A_p$ extracted by GSVMORC approximates the classification obtained by SVM. Note that r refers to the area associated to class A_p.

Proof: The proof structure is similar to that given by Garcez et al 2001 [2].

First, we have to show that a rule R extracted either at the extracting stage or at the post-processing stage can be obtained by querying the SVM. This can be proven by contradiction.

Consider a set of m-dimensional input vectors and a SVM $f(\mathbf{x})$. If the extracted rule R is not obtainable by querying the network, then there must exist a point $\mathbf{x}_i$ in r such that the class output of $f(\mathbf{x}_i)$ is not equivalent to A_p. By the definition of the rule, all the points inside the area r covered by R should refer to the same class A_p. If a point $\mathbf{x}_i$ exists that belongs to the other class, this contradicts to the definition of the rule. Therefore, R must be obtainable by querying the network.

Subsequently, in order to guarantee all the points in r belonging to A_p, the constraint $\int_L^U |f(\mathbf{x}) - A_p| = 0$ must be satisfied, where L and U are the lower and upper bounds of the expected range of R.

At the implementation level, the quasi-Monte Carlo $\frac{1}{n}\sum_{i=1}^{n}|f(\mathbf{x}_i) - A_p| = 0$ is used to approximate the above integral process and obtain the rule R. There should then be a potential approximation error $E = \left|\int_{L'}^{U'}|f(\mathbf{x}) - A_p| - \int_{L}^{U}|f(\mathbf{x}) - A_p|\right|$, where L' and U' are the actual lower and upper bounds of R.

The *rule pruning* step utilizes a statistical method to check if the extracted rule has a small error E. In our implementation, the significant level at the *rule pruning* step is set to 95%. Only the rules whose *t-test* outcomes satisfy the standard value at the significant level are kept, which means the estimation of the integral of r is close to the expected value of 0. This ensures that the approximation error is $E = \left|\int_{L'}^{U'}|f(\mathbf{x}) - A_p| - \int_{L}^{U}|f(\mathbf{x}) - A_p|\right| \leq \varepsilon$. Therefore, it can be concluded that the SVM classification on most points is the same as that of R with a rather small difference, and R is said to approximate the classification of the SVM. ■

Theorem 2: With an increasing number of rules, the rule set approximates the behavior of the SVM. Let S denote the area covered by the non-overlapping rule set $R = \{r_i \to A_p, i \geq 1\}$ and V represent *the area of interest* $I(A_p)$. When the number of rules increases, S approximates V, that is $\frac{|V-S|}{V} \leq \epsilon$, where ϵ is an arbitrary small number. Note that r_i refers to an area in class A_p.

Proof: Give an input domain $X \subseteq \Re^m$, a set of classes $Y = \{A_p \mid 1 \leq p \leq CN\}$, where CN is the number of classes, and a classifier function $f : X \to Y$.

Firstly, we need to show that there is an upper bound on *the area of interest*. The definition of *the area of interest* (see Definition 3.2) clarifies that $I(A_p)$ has an upper bound equal to $\prod_{i=1}^{m}(u_i - l_i)$, where u_i and l_i are the upper and lower bounds for *the area of interest*.

Next, we need to show that any part of the *area of interest* can be approximated by a set of rules extracted by GSVMORC (lemma 3), and the more rules we have, the larger the area covered by the rules (lemma 4).

Lemma 3: Consider V' as any part of the *area of interest*. V' should then be approximated by a set of rules extracted by GSVMORC whose area equals S_t. This can be represented as $\frac{|V'-S_t|}{V'} \leq \epsilon$.

Proof: This can be proven by contradiction. Suppose that the intersection between V' and S_t is V_t.

Assume that V' cannot be approximated by S_t extracted by GSVMORC; then the area in V' that is not covered by the rule set is large. It also means that the difference between V' and V_t is large.

Since the difference between V' and V_t is also an area, a set of uniformly distributed synthetic instances can be generated inside it, and GSVMORC is able to extract rules based on these instances. Hence, the size of S_t increases, and the difference between V' and V_t decreases. This process can be continued until $|V' - V_t| \leq \varepsilon'$ so that $\frac{|V'-V_t|}{V'} \leq \varepsilon$, where ε and ε' are arbitrary small numbers. Then,

$$\frac{|V' - S_t|}{V'} = \frac{|V' - V_t - (S_t - V_t)|}{V'} \leq \frac{|V' - V_t| + |S_t - V_t|}{V'}$$

By deduction, it can be worked out that $\frac{|S_t-V_t|}{V'} \leq \varepsilon$.

1) Suppose that the area covered by a rule is S_1 and that the intersection area between S_1 and V is V_1. The difference between S_1 and V_1 refers to the part in S_1 that is classified as the other classes by an SVM. With respect to Theorem 1, an extracted rule from GSVMORC is known to approximate the classification obtained by an SVM. Hence, if the points belonging to the other classes in S_1 exist, they occupy only a very small part of S_1 so that the deviation at this small part cannot influence the approximation value of S_1, which is $\frac{1}{n}\sum_{i=1}^{n}|f(\mathbf{x}) - A_p|$. Therefore, by comparing this with V_1, it can be concluded that $\frac{|S_1-V_1|}{V_1} \leq \varepsilon$, where ε is a arbitrary small number.
2) Let us assume that $\frac{|S_t-V_t|}{V_t} \leq \varepsilon$, where t is an arbitrary integer.
Then, for $S_{t+1} = S_t + S_1$ and $V_{t+1} = V_t + V_1$,

$$\begin{aligned}\frac{|S_{t+1} - V_{t+1}|}{V_{t+1}} &= \frac{|S_t + S_1 - V_t - V_1|}{V_{t+1}} \\ &\leq \frac{|S_t - V_t|}{V_{t+1}} + \frac{|S_1 - V_1|}{V_{t+1}} \\ &= \frac{\varepsilon(V_1 + V_t)}{V_{t+1}} = \varepsilon\end{aligned}$$

Therefore,

$$\begin{aligned}\frac{|V' - S_t|}{V'} &= \frac{|V' - V_t - (S_t - V_t)|}{V'} \\ &\leq \frac{|V' - V_t| + |S_t - V_t|}{V'} \\ &\leq 2 * \varepsilon = \epsilon\end{aligned}$$

where $\epsilon = 2 * \varepsilon$ is an arbitrary small number.

From the above, it can be demonstrated that V' can be approximated by a set of rules extracted by GSVMORC. ■

Lemma 4: The area S_{t+1} covered by $t + 1$ rules is larger than the area S_t covered by t rules (t is an integer).

Proof: [Proof] As the rule is defined to be non-overlapping (Section 4.6), this means that there is no intersection between the rules. The volume covered by $t + 1$ rules must then be larger than that covered by t rules.

In other words, if $S_{t+1} \leq S_t$, then there must exist at least two rules that overlap. This contradicts the definition of non-overlapping. Therefore, it can be concluded that $S_{t+1} > S_t$. ■

Lemma 3 demonstrates that any part of the *area of interest* can be approximated by a set of rules. Lemma 4 shows that the greater the number of rules extracted, the larger the non-overlapping area covered by the rules. Therefore, when the number of rules increases, the area covered by the rules can finally approximate *the area of interest*. Furthermore, the difference between the area covered by the rules (denoted by S) and the *area of interest* (denoted by V) satisfies $\frac{|V-S|}{V} \leq \epsilon$, where ϵ is an arbitrary small number.

Hence, by increasing the number of rules, the rule set extracted by GSVMORC can approximate the behavior of SVM networks. ■

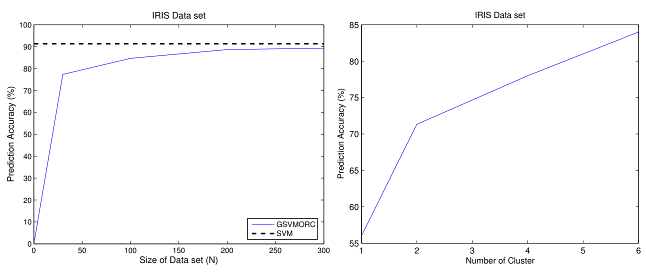
V. Experimental Results

We performed experiments in three real-world datasets, all obtained from the UCI Machine Learning repository: the Monk's problem, the Iris flower dataset and the Breast Cancer-Winsconsin dataset. All of the three Monk's problems have seven attributes, which include an Id feature for each instance. The other attributes are categorical, labelled as $a1, a2, a3, a4, a5, a6$. All instances in the Monk's problems are divided into two classes: $class1 = 0$ and $class2 = 1$. The Iris problem correlates four attributes (sepal length (SL), sepal width (SW), petal length (PL) and petal width (PW)) with three classes (Setosa, Versicolour and Virginica). For the Breast Cancer dataset, there are nine attributes (Clump Thickness (CT), Uniformity of Cell Size (UCSZ), Uniformity of Cell Shape (UCSP), Marginal Adhesion (MA), Single Epithelial Cell Size (SECS),Bare Nuclei (BN), Bland Chromatin (BC), Normal Nucleoli (NN) and Mitoses (MS)) and two classes (benign and malignant). We have used 5-fold cross validation in the experiments. For each fold:

1) We trained the SVM using different algorithms; for the Monk-2 and Iris datasets, we used DAGs-SVM [9], and for the Monk-1, Monk-3 and Breast Cancer dataset, we used SMO [10].
2) We generated a number of training data and queried the trained SVM to obtained the class label.
3) We applied the rule extraction algorithm to datasets of varying sizes.
4) We applied the rule extraction algorithm to datasets of varying number of clusters.
5) We measured rule accuracy with respect to the test set, rule fidelity to the SVM, and rule comprehensibility.

Accuracy measures the ability of the rules in predicting unseen cases according to a test set. The results show that when the data size increases, the accuracy of the rules increases, converging to that of the SVM as illustrated in the following Figures. For example, for the Iris dataset (see Figure 5(a)), when N equals 30, the accuracy is only 77.33%. However, 84.67% is achieved when N equals 100. It finally reaches 89.33% at $N = 300$, which is a value near to the accuracy of SVM. The same behavior is verified for the breast cancer dataset (see Figure 6(a)). When N equals 50, the accuracy is only around 60%. Although the rate of the increase reduces, it still causes the accuracy result to finally reach 90.14% at $N = 200$. While for Monk's problem (the results are shown in Figure 7(a), 8(a) and 9(a) , when N reaches 100, GSVMORC achieves 100% accuracy for Monk-1. For Monk-2, GSVMORC obtains 84.8% for a 200-size training set compared with the 85.7% accuracy classified by the SVM network. GSVMORC also achieves an average 95% correctness for a 100-size training set in the case of the Monk-3 problem, while the SVM obtains around 94% accuracy.

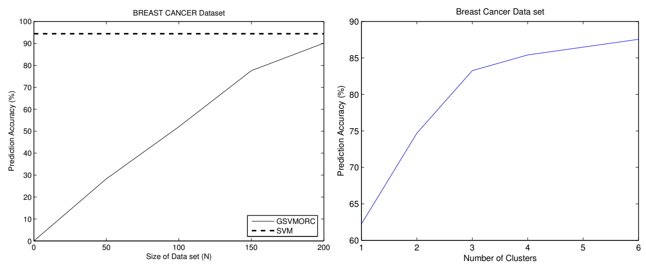
Figure 5(b), 6(b), 7(b), 8(b), 9(b) show that when the number of clusters increases, the accuracy increases as well. As an example, in the Iris dataset, GSVMORC classifies only 56% instances correctly when the cluster number is one.



(a) (b)

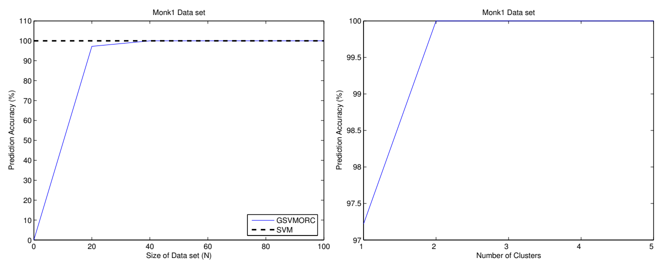
Fig. 5. Result of Iris dataset

But it predicts 84% instances correctly when the number of clusters goes to six. It is interesting to note that the value of 84%, closest to the SVM accuracy of 89.33%, is obtained when the training set contains 300 instances and has one cluster for each class. The same convergence movement occurs for the Breast Cancer dataset. An accuracy of only 62.23% is obtained when each class has one cluster, but 87.55% of instances are predicted correctly when the number of clusters goes to six. For the Monk's problem, the accuracy is only 97%, 62% and 58% respectively for Monk-1, Monk-2 and Monk-3, when the number of cluster is one. However, the accuracy increases to 100% for Monk-1, 78% for Monk-2 and 90% for Monk-3, when the number of clusters increases to 5. This is why we have chosen to use the stopping criterion of Section 3 to find an appropriate cluster value.



(a) (b)

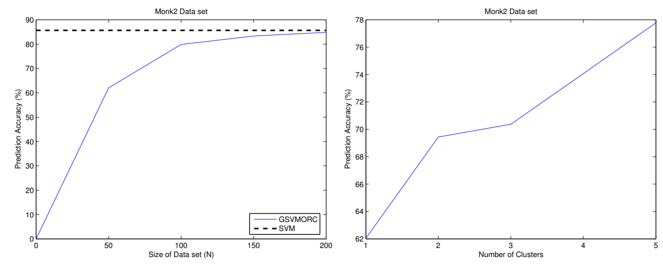
Fig. 6. Result of Breast Cancer dataset



(a) (b)

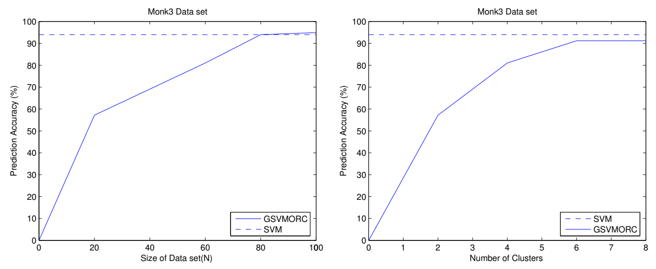
Fig. 7. Result of the Monk-1 problem

Fidelity measures how close the rules are to the actual behavior of the SVM, as opposed to its accuracy w.r.t a test set. The fidelity rate in Monk-1, Iris and Breast Cancer



(a) (b)

Fig. 8. Result of the Monk-2 problem



(a) (b)

Fig. 9. Result of the Monk-3 problem

problems has been 100%. Monk-2 and Monk-3 obtained 99.12% and 98.5% fidelity.

Comprehensibilitymeasures the number of rules and the number of conditions per rule. The following is an example of the extracted rules on the Iris dataset. GSVMORC obtains on average ten rules for each class, with four conditions per rule.

sepal length$= [4.3, 6.6] \bigwedge$ *sepal width*$= [2.0, 4.0] \bigwedge$ *petal length*$= [2.7, 5.0] \bigwedge$ *petal width*$= [0.4, 1.7] \rightarrow$ *Iris Versicolour*

The rule above correctly predicts 45 out of 150 instances in the data set. Overall, 93% training and test examples in the Iris data set are predicted correctly.

The following is an example of the extracted rules on Breast Cancer problem. GSVMORC obtained on average 26 rules for class 1 and 81 rules for class -1, with an average 7.2 conditions per rule. The final rule set classifies 90.14% of the test cases and 93% of the whole data set correctly.

$a_3 = [4, 9] \bigwedge a_5 = [3, 9] \bigwedge a_6 = [10, 10] \bigwedge a_7 = [5, 9] \rightarrow -1$

For Monk-1, GSVMORC obtained four rules for all classes. On average, each rule has 2.37 conditions. This is from the 100 synthetic training instances, which cover 100% of the test cases. For Monk-2, around 38 rules were extracted, with around 4.1 conditions per rule for class 1. For class 0, 24 rules with 5.8 conditions per rule were extracted. For Monk-3, 11 rules were extracted, with around 3.4 conditions per rule for class 1, while 6 rules with 2.7 conditions per rule were extracted for class 0 (for example $a_1 = 1 \bigwedge a_2 = 1 \rightarrow class = 1$; in this case, $a_1 = 1$ denotes that a_1 is $true$).

Discussion and Comparison with Related Work

We have shown how rules can be extracted from SVMs without needing to make many assumptions about the architecture, initial knowledge and training data set. We have also demonstrated that GSVMORC is able to approximate and simulate the behavior of SVM networks correctly.

The accuracy and the fidelity of our algorithm are better than those obtained by the SVM rule extraction approach proposed in [19], which is an important work on rule extraction from SVMs. GSVMORC obtains 100% accuracy for Monk-1, while the SVM+prototype [8] predicts only 59.49% of instances correctly in the test set. Compared with the 63.19% test-set performance by the rule base and the 82.2% SVM classification rate, GSVMORC achieves 84.8% accuracy on the test set while the classification of SVM is 85.7%. GSVMORC also obtains 100% fidelity, but the SVM+prototype has just 92.59% and 75.95% agreement with SVM networks in the respective data sets. (Note that the performance measures for the SVM+prototype and other techniques originate from published papers and not our own experiments [8].)

In the Iris problem, the SVM+prototype [19] reports an accuracy rate of 71% for interval rules and a fidelity rate of 97.33% compared with the 96% accuracy of RBF SVM networks [16]. Our algorithm achieves a maximum fidelity rate (100%) with a far higher accuracy (89.33%), while SVM accuracy is 91.33%.

In the Breast Cancer problem, the *ExtractRules-PCM* approach of [6] achieves an average accuracy of 98% compared with an SVM accuracy of 95%. In contrast, our extraction algorithm shows high agreement between the rules and the SVM. A good fidelity indicates that the rule extraction method mimics the behavior of SVM networks, and a better understanding of the learning process is therefore obtainable.

In the Breast Cancer problem, the eclectic approach of [15] achieved 82% accuracy compared with an SVM accuracy of 95%. Our approach performs better than that approach. It achieves 89% accuracy which is more closer to the SVM accuracy (94%).

In RuleExSVM [23], another vital algorithm for SVM rule extraction, rules are extracted based on the SVM classification boundary and support vectors. RuleExSVM has high rule accuracy and fidelity in the Iris and Breast Cancer problems. For example, for the Iris problem, RuleExSVM achieves 98% relative to the 97.5% of the SVM classification results, and in the Breast Cancer domain, 97.8% accuracy is obtained. The fidelity levels of these two domains range from 99.18% to 99.27%. However, RuleExSVM constructs the rules largely depending chiefly on the training samples and support vectors. It is difficult to apply to networks other than SVM networks. On the other hand, GSVMORC has a high level of generality in a wide array of networks.

Finally, on the issue of comprehensibility, *NeuroRule* [18], an approach to pruning neural networks and using decompositional extraction, produces five rules with 4.2 conditions per rule in the Breast Cancer domain compared to 107 rules with 7.2 conditions per rule in our approach. However, *NeuroRule* relies on special training procedures that facilitate the extraction of the rules. GSVMORC, on the other hand, is architecture-independent and has no special training requirements. It offers the highest fidelity rate and an interesting convergence property, as illustrated in the figures above.

VI. Conclusion and Future Work

We have presented an effective algorithm for extracting rules from SVMs so that results can be interpreted by humans more easily. A key feature of the extraction algorithm is the idea of trying to search for *optimal rules* with the use of expanding hypercubes, which characterize rules as constraints on a given classification. The main advantages of our approach are that we use synthetic training examples to extract accurate rules, and treat the SVM as an oracle so that the extraction does not depend on specific training requirements or given training data sets. Empirical results on real-world data sets indicate that the extraction method is correct, as it seems to converge to the true accuracy of the SVM as the number of training data increases and 100% fidelity rates are obtained in every experiment. In future work, we may consider using different shapes than hypercubes for the extraction of rules and compare results and further improve the comprehensibility of the extracted rules.

Support Vector Machines have been shown to provide excellent generalization results and better classification results than parametric methods or neural networks as a learning system in many application domains. In order to develop further the study of the area, we need to understand why this is so. Rule extraction offers a way of doing this by integrating the statistical nature of learning with the logical nature of symbolic artificial intelligence [20].

References

[1] Charles. A and Dennis. J. E. Analysis of generalized pattern searches. *SIAM Journal on Optimization*, 13(3):889–903, 2003.

[2] Garcez. A. A, Broda. K, and Gabbay. D. Symbolic knowledge extraction from trained neural networks: a sound approach. *Artificial Intelligence*, 125(1 - 2):155 – 207, January 2001.

[3] Karaivanova. A, Dimov. I, and Ivanovska. S. A quasi-monte carlo method for integration with improved convergence. In *LSSC '01: Proceedings of the Third International Conference on Large-Scale Scientific Computing-Revised Papers*, pages 158–165, London, UK, 2001. Springer-Verlag.

[4] Thrun. S. B. Extracting provably correct rules from artificial neural networks. Technical report, Institut für Informatik III, Universität Bonn, 1993.

[5] Pop. E, Hayward. R, and Diederich. J. Ruleneg: Extracting rules from a trained ann by stepwise negation. Technical report, December 1994.

[6] Fung. G, Sandilya. S, and Rao. B. R. Rule generation from linear support vector machines. *KDD'05*, pages 32 – 40, 2005.

[7] Morohosi. H and Fushimi. M. A practical approach to the error estimation of quasi-monte carlo integration. In Niederreiter. H and Spanier. J, editors, *Monte Carol and Quasi-Monte Carlo Methods*, pages 377–390, Berlin, 1998. Springer.

[8] Núñez. H, Cecilio Angulo, and Andreu Català. Hybrid architecture based on support vector machines. In *Proceedings of International Work Conference on Artificial Neural Networks*, pages 646–653, 2003.

[9] Platt. J, Cristianini. N, and Shawe-Taylor. J. Large margin dags for multiclass classification. *Advances in Neural Information Processing Systems*, 12:547–553, 2000.

[10] Platt. C. J. Sequential minimal optimization: A fast algorithm for training support vector machines. Technical Report MSR-TR-98-14, Microsoft Research, April 1998.

[11] Saito. K and Nakano. R. Law discovery using neural networks. *Proceeding NIPS'96 Rule Extraction From Trained Neural Networks Workshop*, pages 62 – 69, 1996.

[12] Craven. W. M and Shavlik. J. W. Using sampling and queries to extract rules from trained neural networks. In *International Conference on Machine Learning*, pages 37–45, 1994.

[13] Craven. W. M and Shavlik. J. W. Extracting tree-structured representations of trained networks. In David S. Touretzky, Michael C. Mozer, and Michael E. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 24–30. The MIT Press, 1996.

[14] Fu. L. M. Rule learning by searching on adapted nets. *Proceedings of the Ninth National Conference on Artificial Intelligence (Anaheim CA)*, pages 590–595, 1991.

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

[16] Nú nez. H, Angulo. C, and Català. Rule based learning systems for support vector machines. *Neural Processing Letters*, 24(1):1–18, August 2006.

[17] Filer. R, Sethi. I, and Austin. J. A comparison between two rule extraction methods for continuous input data. *Proceeding NIPS'97 Rule Extraction From Trained Artificial Neural Networks Workshop*, pages 38 – 45, 1994.

[18] Setiono. R. Extracting rules from neural networks by pruning and hidden unit splitting. *Neural Computation*, 9:205 – 225, 1997.

[19] Nũñez. N, Angulo. C, and Catalá. A. Rule extraction from support vector machines. *Proceeding of European Symposium on Artificial Neural Networks Bruges(Belgium)*, pages 107 – 112, 2003.

[20] Leslie G. V. Three problems in computer science. *J. ACM*, 50(1):96–99, 2003.

[21] Vapnik. N. V. *Statistical learning theory*. John Wiley and Sons, INC, 1998.

[22] Morokoff. J. W and Caflisch. R. E. Quasi-Monte Carlo integration. *J. Comp. Phys.*, 122:218–230, 1995.

[23] Fu. X-J, Ong. C-J, Keerthi. S, Gih G-H, and Goh L-P. Extracting the knowledge embedded in support vector machines. volume 1, pages 291–296, 2004.

[24] Zhang. J. Y and Liu. Y. X. Svm decision boundary based discriminative subspace induction. *Pattern Recognition*, 38(10):1746 – 1758, 2005.