Neural Processing Letters (2006) 24:1–18
DOI 10.1007/s11063-006-9007-8

Rule-Based Learning Systems for Support Vector Machines

HAYDEMAR NÚÑEZ[1], CECILIO ANGULO[2,⋆] and ANDREU CATALÀ[2]
[1]*Artificial Intelligence Laboratory, Central University of Venezuela, Caracas, Venezuela.*
[2]*Knowledge Engineering Research Group, Technical University of Catalonia, Barcelona, Spain. e-mail: cecilio.angulo@upc.edu*

Abstract. In this article, we propose some methods for deriving symbolic interpretation of data in the form of rule based learning systems by using Support Vector Machines (SVM). First, Radial Basis Function Neural Networks (RBFNN) learning techniques are explored, as is usual in the literature, since the local nature of this paradigm makes it a suitable platform for performing rule extraction. By using support vectors from a learned SVM it is possible in our approach to use any standard Radial Basis Function (RBF) learning technique for the rule extraction, whilst avoiding the overlapping between classes problem. We will show that merging node centers and support vectors explanation rules can be obtained in the form of ellipsoids and hyper-rectangles. Next, in a dual form, following the framework developed for RBFNN, we construct an algorithm for SVM. Taking SVM as the main paradigm, geometry in the input space is defined from a combination of support vectors and prototype vectors obtained from any clustering algorithm. Finally, randomness associated with clustering algorithms or RBF learning is avoided by using only a learned SVM to define the geometry of the studied region. The results obtained from a certain number of experiments on benchmarks in different domains are also given, leading to a conclusion on the viability of our proposal.

Abbreviations. RBF – Radial Basis Function; RBFNN – Radial Basis Function Neural Network; SVM – Support Vector Machine

Key words. rule based learning system, rule extraction, RBF neural networks, support vector machine

1. Introduction

When Artificial Neural Networks and general connectionist paradigms are used for constructing supervised classifiers, black-box models are produced, a significant drawback for a learning system even if the obtained performance is satisfactory for the user. With the aim of making the resulting classifier system interpretable, in the last 15 years rule extraction methods for trained neural networks have been developed [1, 5, 15, 22, 25]. In the case of Radial Basis Function Neural Networks (RBFNN) [12, 16], proposed rule extraction algorithms [8, 10, 14] usually impose restrictions during the training phase, to largely avoid overlapping between classes or categories and thus to facilitate the knowledge extraction process.

⋆Corresponding author.

Also, it has been demonstrated in the last decade that Support Vector Machines, (SVM) [4, 6, 12], which are derived from the statistical learning theory by V. N. Vapnik [23], have an excellent classification and approximation qualities on all kind of problems. However, as it is the case of artificial neural networks, models generated by these machines are difficult to understand from the point of view of the user.

In the research on rule extraction from neural networks, it has been recently argued that a fidelity-accuracy dilemma exists, hence it must be distinguished between rule extraction using neural networks and rule extraction for neural networks [24]. Main problem addressed in this work is how to translate the knowledge acquired for a learned RBFNN or SVM during a supervised training to a description in a new representing language. The new model should be functionally equivalent to the learned machine that it was derived for, so that the same outputs be obtained.

Interpretability using the geometry of learned SVM classifiers in the form of a rule-based learning system is our goal. Two novel rule extraction methods for supervised classification problems are presented based on both, prototype vectors and vectors lying on the boundaries of the classes. Prototype vectors will be obtained either, from centers of a RBFNN trained under any training regime (supervised or unsupervised) or from any clustering algorithm. They will be used as centers for the generated regions defining rules. Vectors on the frontiers of the classes will be selected from the set of support vectors obtained from a trained standard SVM for the supervised classification problem. These vectors will be used as delimiters of the regions defining rules and they will help to avoid overlapping between classes.

On the next section, the first novel algorithm based on a RBFNN generating prototype vectors is presented. RBFNN is first trained to obtain centers for the rules, next support vectors obtained from a trained standard SVM are used as classes delimiters, thus obtaining a new rule extraction method from RBFNN. This approach starting from a trained RBFNN is the most common in the literature, producing descriptions in the form of ellipsoid rules. Furthermore, the interpretation of the rules is facilitated through a second derivation in the form of hyper-rectangle rules, which are even more amenable for the user. The efficient use of the knowledge associated with the support vectors due to the geometry of the input space allows this method to solve the usual problematic of overlapping between classes when extracting rules, and it imposes no restrictions on the RBF network architecture, or its training regime.

On Section 3, in a dual form, starting with support vectors selected from a trained SVM and prototype vectors generated by any clustering algorithm or even by a RBFNN, a rule extraction procedure for SVM is proposed. This method is performed independently from the hyper-parameters and kernel selection in the SVM, and from the clustering algorithm that is implemented in order to obtain the prototype centers. The rule extraction method relies only on the geometry in the input space adopted from the learning functions. Like the previous one, it produces descriptions in the form of ellipsoid and hyper-rectangle rules.

Both proposed methods use a clustering algorithm or an unsupervised learned RBFNN to determine prototype vectors for the generation of rules during the rule extraction procedure for a supervised classification problem: for the whole input space in the first presented method, and for each separated class in the second one. High variability in the results is expected, and so is observed on several data sets, since it is well known that the performance of clustering algorithms varies when initial conditions of the training are modified. As a final extension, the second algorithm was modified for the exclusive use of the information supplied by a SVM, so variability is eliminated and it is showed that it is possible to cluster data from a learned SVM. Finally, some concluding remarks are presented and future work is discussed.

2. Interpretation of RBFNN with Support Vectors

The hypothesis space implanted by these learning machines is constituted by functions in the form,

$$f(\mathbf{x}, \mathbf{w}, \mathbf{v}) = \sum_{i=1}^{m} w_k \phi_k (\mathbf{x}, \mathbf{v_k}) + w_0. \tag{1}$$

Nonlinear activation function ϕ_k expresses similarity between any input pattern $\mathbf{x}$ and the center $\mathbf{v_k}$ by means of a distance measure. Each function ϕ_k defines a region in the input space, the receptive field, on which the node produces an appreciable activation value. In the common case, when the Gaussian function is used, the center $\mathbf{v_k}$ of the function ϕ_k defines the prototype of an input cluster k and the variance σ_k defines the size of the covered region in the input space.

A limited number of rule extraction methods directed to RBFNN have been developed [8, 10, 14] since the local nature of RBF neural networks makes them an interesting platform for performing rule extraction. However, basis functions overlap to some degree in order to give a relatively smooth representation of the distribution of training data [12, 16], representing a major drawback for rule extraction. To avoid the overlapping, most of the methods in the literature use special training regimes or special architectures in order to guarantee that RBF nodes are assigned and used by a single class.

The solution provided by a SVM for classification tasks is a summation of kernel functions, $k(\cdot, \cdot)$, constructed on the basis of a set of sv support vectors, $\mathbf{x_i}$,

$$f_a(\mathbf{x}) = \operatorname{sign}\left(\sum_{i=1}^{sv} \alpha_i y_i k(\mathbf{x}, \mathbf{x_i}) + b \right). \tag{2}$$

Support vectors are the training patterns nearest the separation limit between classes or they are misclassified data; in any case, they are the most informative samples for the classification task. Moreover, it was already recognized in [21] that the SVM relies only on distances in the Reproducing Kernel Hilbert Space.

Therefore, when Gaussian kernels or polynomial degree one or two kernels are used, the input space is endowed with a certain geometrical structure according to the kernel selected. This structure, based on standard distances, can be merged with those attached to the node centers and the RBF chosen, usually a Gaussian function.

2.1. RULE EXTRACTION METHODOLOGY

Node centers generated by RBFNN, independently of the used training regime, are located in zones representing a 'center of mass' for the patterns associated with this center, whereas support vectors lie on the geometrical border of the class. Hence, by limiting influence zones associated with RBF node centers with support vectors near the frontier, we will show that it is possible to construct regions in the input space, which will be later translated to if-then rules. These regions will be originally defined as ellipsoids, that is, we will generate rules whose antecedent is the mathematical equation of an ellipsoid. In order to increase the explanatory power of the generated rules for the user, hyper-rectangles will be defined from parallel ellipsoids to the axes in a later phase. They will generate rules whose premise is a set of restrictions on the values of each variable (Figure 1).

Two main features will be considered to evaluate the performance of the rule extraction task in the form of ellipsoids: (i) ellipsoids must adjust to the form of the decision limit defined by the SVM, exhibiting an overlapping between classes that is as low as possible; (ii) ellipsoids must be as large as possible to cover many data producing a small number of rules.

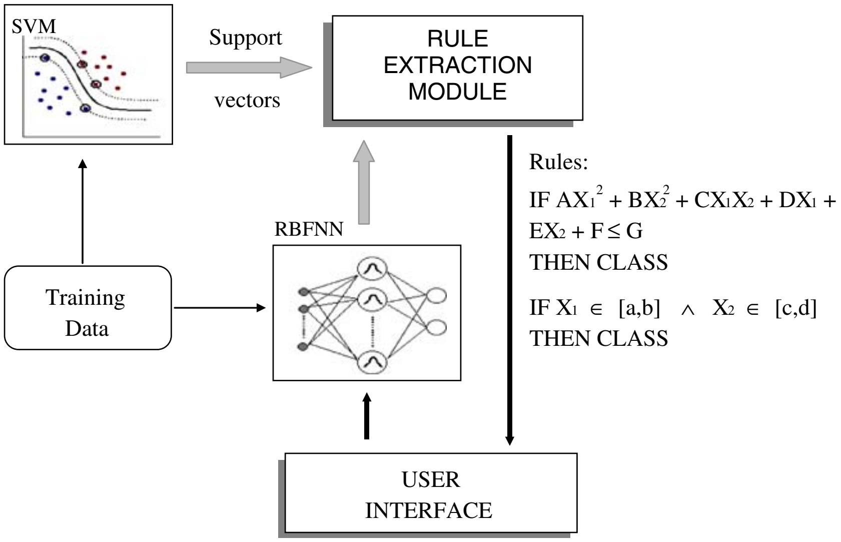

Figure 1. The rule extraction method from RBFNN and support vectors.

In our proposal, the rule extraction method does not involve any special RBFNN training algorithm or specially devoted architecture; it extracts rules from any ordinarily learned RBFNN. In order to solve the overlapping problem between rules, a SVM is used as a frontier pattern selector. In the following, we describe the proposed algorithm (Algorithm 1) for deriving a set of rules from a trained RBFNN.

Initially, a partition of the input space is made by assigning each input pattern to its closest center, a RBF node, according to the Euclidean distance. It must be noted that when a pattern is assigned to its closest center, it will be assigned to the RBF node that will give the maximum activation value for that pattern, without taking into consideration the class applicable for this node.

```
Rules = Rule_Extraction{Learned RBFNN, Training set}
    Support_Vectors = train_SVM{Training set}
    assign patterns to the nearest RBF node
    assign class to RBF nodes using RBFNN
    for each RBF node
            for each RBF node with patterns having the same class
                    Region = Build_Region{RBF node, Patterns, Support_Vectors}
            end_for
            for other classes in the partition
                    determine midpoint of data in the class
                    Region = Build_Region{Midpoint, Patterns, Support_Vectors}
            end_for
    end_for
    translate each Region to a Rule
end_Rule_Extraction
```

Algorithm 1. General algorithm for rule extraction from RBFNN and support vectors.

Next, a class label is assigned for each center of RBF units. The output value of the RBF network for each center is used in order to determine this class label. Then, with data having the same class as the node in the partition, an ellipsoid is constructed for each node. Nevertheless, according to the proposed construction, data belonging to other classes could be present in the partition of RBF units. In order to divide these data, we determine the mid-point of each class in the region. Each mean is used as a center of its class to construct a new ellipsoid with the associated data. Once the ellipsoids have been determined, they are transferred to rules.

Ellipsoids exhibiting low overlapping between classes and covering data to produce a small number of rules are still the main features to be accomplished for the rule extraction procedure. In order to obtain ellipsoids with these characteristics, the support vectors are used to determine the axes and vertices (Build_Region{Midpoint or RBF node, Patterns, Support_Vectors} in Algorithm 1): first, the RBF node or the midpoint (prototype) of the class will be used as the center of the ellipsoid. Next, the support vector of the same class with both a value of the Lagrange multiplier α smaller than the C parameter and a maximum distance to the prototype or node is chosen. The straight line defined by these two points determines the first axis of the ellipsoid. The rest of the axes and the associated vertices are determined by geometry. To construct hyper-rectangles a similar procedure is followed. The only difference is that parallel lines to the axes are used to define the axes of the associated ellipsoid.

This procedure will generate several rules by each node, as many rules as the number of classes included in the partition. In Figure 2, we show two examples with both ellipsoids and hyper-rectangle type rules, generated by the extraction rules algorithm from a trained RBFNN on artificial data on the plane.

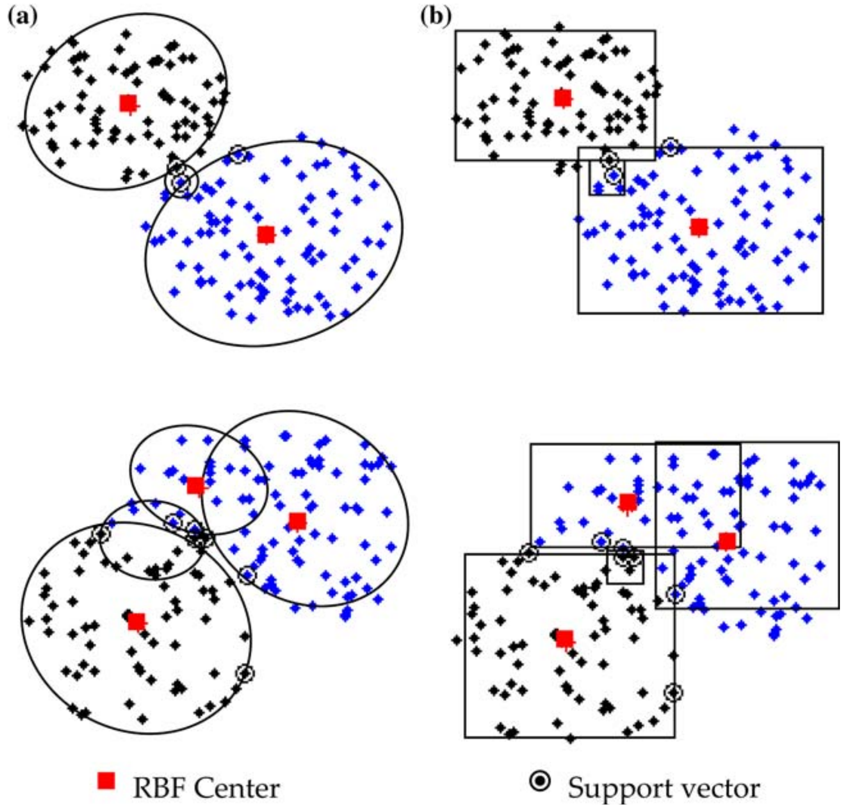

Figure 2. Two examples of (a) ellipsoids and (b) hyper-rectangles generated by the rule extraction method from a trained RBFNN on artificial data on the plane.

2.2. REFINING THE RULES BASE

In order to eliminate or to reduce the overlapping that could exist between ellipsoids of different classes, an overlapping test is applied. The overlapping test (Condition(i) = Overlapping_Test{Region(i), Support_Vectors} in Algorithm 2) checks when a support vector from another class exists within the ellipsoid. Since support vectors are the points nearest to the decision limit, their presence within an ellipsoid of different class is a reliable indicator of overlapping. When the overlapping test is positive, the ellipsoid is divided.

```
Region_Rules = Region_Partition{Region, Support_Vectors}
    Condition(1) = Overlapping_Test{Region, Support_Vectors}
    Data = Region_Data; Number_of_Regions = 2
    while any Condition(i) == True & Num_Iterations <= Max_Iterations
      {Prototypes, Partition} = Determine_Prototypes{Data, Number_of_Regions}
      for i=1:Number_of_Regions
           Region(i) = Build_Region{Prototypes(i), Partition(i), Support_Vectors}
           Condition(i) = Overlapping_Test{Region(i), Support_Vectors}
      end_for
      Number_of_New_Regions = 1
      New_Data = empty set
      for i=1:Number_of_Regions
           if Condition(i) == False | Num_Iterations == Max_Iterations
              Region_Rules = Region(i)
           else
              Number_of_New_Regions = Number_of_New_Regions + 1
              New_Data = New_Data + Partition(i)
           end_if
      end_for
      Data = New_Data; Number_of_Regions = Number_of_New_Regions
    end_while
end_Region_Partition
```

Algorithm 2. Procedure for refining the rule base in order to reduce the overlapping between classes.

This procedure will allow the rule base be refined in order to reduce the overlapping between classes. When the ellipsoids are divided, more specific rules are generated to exclude data from other classes. Algorithm 2 can be executed in an iterative form; depending on the number of iterations, two or more partitions by ellipsoid can be obtained. The user can establish the maximum number of iterations, thus controlling the number of generated rules for each RBF node.

According to the extracted rules, the generated system classifies an example by assigning it to the class of the nearest rule in the knowledge base, following the nearest neighbor philosophy, by using the Euclidian distance. If an example is covered by several rules, we choose the class of the most specific ellipsoid or hyper-rectangle containing the example; that is, the one with the smallest volume.

3. Interpretation of SVM with Prototype Vectors

From a geometrical perspective, support vectors are the most informative samples for the classification task, lying in the frontier between classes. On the other hand, prototype vectors representing classes can be generated by any clustering algorithm, and so a RBFNN will be not longer necessary to extract rules. Thus, a rule extraction procedure can be derived – as a dual one from the formerly presented one [18] – : first a standard SVM is learned on the training set for the supervised classification problem, generating a set of of support vectors, next a clustering algorithm determines prototype vectors for the classes, and finally regions of ellipsoids (hyper-rectangles) are built to be translated into if-then rules (Figure 3). The main differences of this methodology from the RBFNN-based rule extraction one are (Figure 4) as follows:

- When the classification task is performed: formerly, RBF node centers to be considered prototype vectors were determined having no consideration about classification of the patterns; now clustering implemented to find prototype vectors works for each separated class.
- Extracted rules from the SVM bi-classifier only cover one class, the remaining being space assigned to the another class.

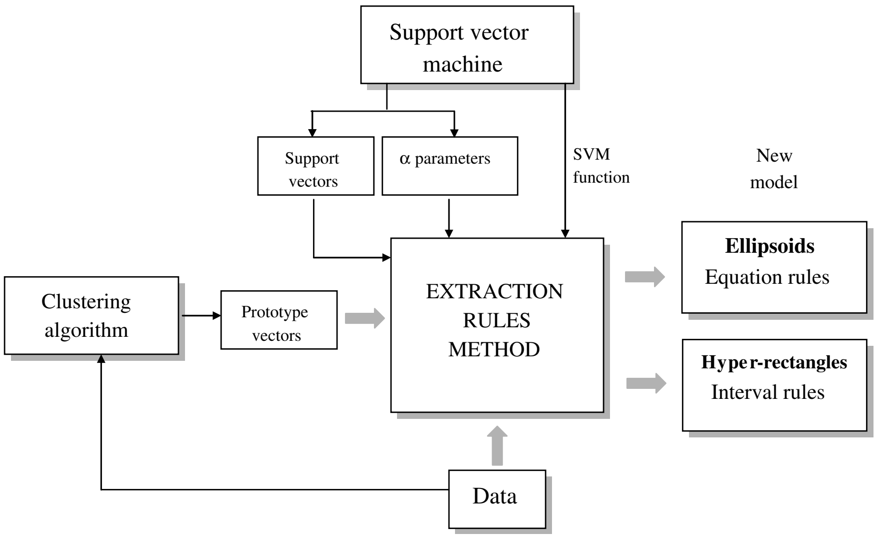

Figure 3. The rule extraction method from SVM and prototype vectors.

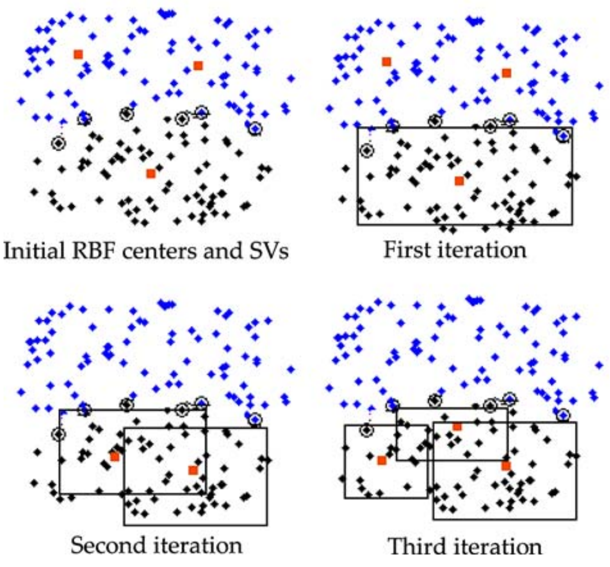

Figure 4. Hyper-rectangles generated by applying the iterative procedure of Algorithm 3.

The number of regions necessary to describe the SVM model will depend on the form of the decision limit. It usually happens that only one ellipsoid is not sufficient to describe the data. Then, the rule base is determined by an iterative procedure beginning with the construction of a general ellipsoid, which is divided into ellipsoids that adjust progressively to the form of the surface decision determined by SVM (Algorithm 3). To determine the right moment to divide an ellipsoid, a partition test is applied. When the test result is positive for one ellipsoid, it will be divided. A partition test is considered as positive (Condition(i) = Partition_Test{Region(i), Support_Vectors} in Algorithm 3) if the generated prototype belongs to another class, if one of the vertices belongs to another class or if a support vector from another class exits within the region. To determine the class label of the prototypes and the class label of the vertices the SVM function is used.

```
Rules = Rules_Generation{Learned SVM, Training set}

    Number_of_Regions = 1
    for each Class
        Data = Class_Data
        {Prototypes(0), Partition(0)} = ...
        Determine_Prototypes{Data, Number_of_Regions}
        Region(0) = Build_Region{Prototypes(0), Partition(0), Support_Vectors}
        Condition(1) = Partition_Test{Region(0), Support_Vectors}
        Number_of_Regions = 2
        while any Condition(i) == True & Num_Iterations <= Max_Iterations
            {Prototypes(i), Partition(i)} = ...
            Determine_Prototypes{Data, Number_of_Regions}
            Region(i) = Build_Region{Prototypes(i), Partition(i), Support_Vectors}
            Condition(i) = Partition_Test{Region(i), Support_Vectors}
            Number_of_New_Regions = 1
            New_Data = empty set
            for i=1:Number_of_Regions
                if Condition(i) == False | Num_Iterations == Max_Iterations
                    Region_Rules = Region(i)
                else
                    Number_of_New_Regions = Number_of_New_Regions + 1
                    New_Data = New_Data + Partition(i)
                end_if
            end_for
            Data = New_Data; Number_of_Regions = Number_of_New_Regions
        end_while
    end_for
    translate each Region to a Rule
end_Rules_Generation
```

Algorithm 3. Iterative procedure for extracting a rule base from a learned SVM with prototype vectors.

Then, to define the number of rules per class the algorithm proceeds as follow: beginning with a single prototype, the associated ellipsoid is generated. Next, the partition test is applied on this region. If it is negative, the region is translated to a rule. Otherwise, new regions are generated. In this form, each iteration produces m regions with a positive partition test and p regions with a negative partition test. The latter are translated to rules. In the next iteration, the data of the m regions are used to determine $m+1$ new prototypes and to generate $m+1$ new ellipsoids. This procedure collapses once all partition tests are negative or the maximum number of iterations is reached. The process thus allows the number of generated rules to be controlled. Figure 5 shows an example of regions generated for each iteration.

4. Overcoming the Randomness of the Clustering Algorithm

Although the dependency of the final clustering result on the random way in which the prototypes are selected is well known [7], this behavior is in no way desirable. How to solve this randomness in the solution is an open research area. We propose a new framework for dealing with this problem by using only the SVM paradigm that represents an alternative.

In order to eliminate the sensitivity of the rule extraction method to the initial conditions of clustering when prototype vectors are being found, we propose

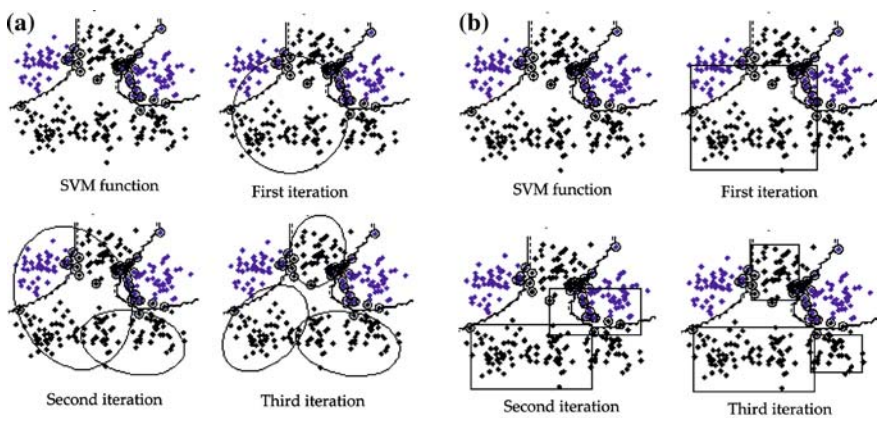

Figure 5. Regions generated in the plane by the rule extraction method, in the form of (a) ellipsoids, or (b) hyper-rectangles.

determining the initial centers for the clustering algorithm from the support vectors [19]. Since the SVM solution is univocally determined for fixed hyper-parameters, only one final base of rules is possible. Thus, if m is the number of necessary prototypes for iteration, then the m initial conditions for the k-means procedure are determined in the following form: for each class,

1. Select m support vectors with the same class label, according to some determined criteria.
2. Assign examples to their closest support vector according to the Euclidean distance.
3. Once the initial partitions have been established, the mean of all instances in each partition is calculated.
4. These points are the initial conditions for the clustering algorithm.

In order to accomplish the first step in the proposed algorithm, three criteria, named the partition schemes, were used to select the support vectors:

– Partition scheme 1: select those vectors with the smallest average dissimilarity with respect to data [11].
– Partition scheme 2: select the support vectors nearest to each other.
– Partition scheme 3: organize the support vectors in descending order according to the α parameters (Lagrange multipliers) and select the first m vectors. A higher value would indicate a more informative pattern [9].

In all the cases, if more prototypes than available support vectors are required, the assembly of initial partitions is completed using those points with the smallest average dissimilarity with respect to the data.

5. Experiments

In order to evaluate the performance of the rule extraction algorithms in this article, we carried out several experiments on data bases obtained from the UCI repository [2]. Table 1 shows the characteristics of the data bases that were used in the experiments. SVM was implemented by using the 'OSU Support Vector Machines Toolbox v3.00' software [13]. In order to validate the hypothesis about the independence of the rule extraction method with respect to the RBFNN training techniques used, two different training procedures were used: the Netlab software [17], which uses the EM algorithm to determine the RBF centers, and the Orr software [20], which uses forward selection.

In the rule extraction community, the fidelity, accuracy, consistency, and comprehensibility (FACC) framework [1] is usually used to evaluate the extracted rules. Consistency has not been evaluated in this work because the presented approach in its final step eliminates any type of randomness in the algorithm and a total consistency exists, so rules extracted under differing training sessions will produce the same classifications of test examples. Instead, it has been measured the overlapping between rules, showing specificity of the rules, and the coverage, measuring the generalization obtained with the extracted rules. The accuracy–fidelity dilemma [24] has not been considered and both measures are presented evaluating the rules base. Therefore, performance of the generated rules is quantified with the following measures:

- Accuracy (Ac): percentatge of classification error provided by the rules on the test set.
- Fidelity (Fd): percentage of the test set for which SVM and rule base output agree.
- Coverage (Cv): percentage of the test set covered by the rule base.
- Overlapping (Ov): percentage of the test set covered by several rules.
- Comprehensibility measured as the number of extracted rules (NR).

Table 1. Data bases and their characteristics.

ID	Database	Data	Attributes	Type	Classes
1	Iris	150	4	Real	3
2	Wisconsin	699	9	Symbolic	2
3	Wine	178	13	Real	3
4	Soybean	47	35	Integer	4
5	New-Thyroid	215	5	Real	3
6	Australian	690	14	Real and symbolic	2
7	Spect	267	23	Binary	2
8	Monk1	432	6	Symbolic	2
9	Monk2	432	6	Symbolic	2
10	Monk3	432	6	Symbolic	2
11	Zoo	101	16	Symbolic	7
12	Heart	270	13	Real, symbolic, and binary	2

Tables 2–5 show the accuracy of the RBF network and the performance values of the extracted rule base obtained when interpreting RBFNN with support vectors. Results were obtained by averaging 10 runs over stratified 10-fold cross-validation when the test set was not provided. We can observe a high agreement between the results obtained from the rule base and those obtained from the RBF network. Further, because the Orr method uses more hidden units to that based on the Netlab software to obtain better performance, it produces a higher number of rules than the latter.

When interpreting SVM with prototype vectors, Tables 6 and 7 show the prediction error of the SVM and the performance values of the extracted rule base for each problem when ellipsoid and hyper-rectangle equations are used, respectively. Results were obtained by averaging over stratified 10-fold cross-validation when a test set was not available. Prototypes to be employed in the rule extraction procedure were determined using the *k*-means clustering algorithm [3]. Main point to be emphasized is the very high fidelity between the rule base and the initial learned SVM with a low overlapping degree. These values indicate that the generated rule base model captures most of the information embedded in the SVM, and it is functionally equivalent to the learned SVM.

As it was expected, during the experimentation process it was observed that the quality of the solution depends on the initial values for the centers; the selection

Table 2. Performance values for several data base when ellipsoids regions are used (with EM algorithm).

ID	RBF	RBF	Ellipsoid rules				
database	nodes	(%) error	(%) Ac	Fd	Cv	Ov	NR
1	4.5	4.0	3.3	96.67	64.67	0.00	6.8
2	2.2	2.9	3.2	98.24	91.21	1.76	5.7
3	3.0	1.1	3.8	97.78	66.43	2.75	9.7
4	5.4	2.0	2.0	96.00	17.00	0.00	6.9
5	9.3	6.5	6.0	94.91	80.99	2.29	16.3
10	6.0	4.8	6.0	95.14	63.19	0.93	14.0

Table 3. Performance values for several data base when user-friendly hyper-rectangle regions are used (with EM algorithm).

ID	RBF	RBF	Interval rules				
database	nodes	(%) error	(%) error	Fd	Cv	Ov	NR
1	4.5	4.0	4.0	97.33	70.67	0.00	6.5
2	2.2	2.9	4.1	96.78	95.01	2.93	14.5
3	3.0	1.1	4.6	95.38	81.30	7.32	10.7
4	5.4	2.0	6.0	91.50	62.50	4.00	10.2
5	9.3	6.5	5.6	94.44	79.07	6.06	16.5
10	6.0	4.8	2.7	97.91	100.0	0.00	11.0

Table 4. Performance values for several data base when ellipsoids regions are used (with Orr algorithm).

ID	RBF	RBF	Ellipsoid rules				
database	nodes	(%) error	(%) error	Fd	Cv	Ov	NR
1	5.1	3.3	2.7	96.67	72.00	0.00	9.2
2	21.5	3.4	3.5	97.22	83.41	0.43	26.4
3	15.2	3.9	3.9	93.26	63.20	0.62	38.1
4	12.4	0.0	4.0	96.00	30.00	0.00	14.7
5	29.32	6.2	6.4	88.87	61.41	0.45	33.0
10	12.0	5.0	6.9	90.97	63.19	3.93	23.0

Table 5. Performance values for several data base when user-friendly hyper-rectangle regions are used (with Orr algorithm).

ID	RBF	RBF	Interval rules				
database	nodes	(%) error	(%) error	Fd	Cv	Ov	NR
1	5.1	3.3	3.3	94.67	74.67	0.00	8.9
2	21.5	3.4	4.9	96.19	95.31	3.95	28.2
3	15.2	3.9	6.2	93.30	85.38	7.26	86.2
4	12.4	0.0	8.5	91.50	91.00	30.50	19.6
5	29.32	6.2	5.4	88.87	72.23	0.90	33.0
10	12.0	5.0	6.4	92.36	100.0	57.40	33.0

Table 6. Performance values obtained for each data base when ellipsoid regions are used.

ID	SVM	Ellipsoid rules				
database	(%) error	(%) error	Fd	Cv	Ov	NR
1	3.3	4.0	98.00	72.00	0.67	7.0
2	3.1	3.4	98.52	89.15	0.30	4.0
3	2.2	1.7	98.30	67.49	0.55	5.9
4	0.0	0.0	100.0	33.00	0.00	6.3
5	3.2	3.2	97.21	80.07	0.00	7.1
6	12.7	13.3	93.60	65.70	2.86	18.4
7	10.2	11.7	96.26	21.39	0.53	14.0
8	5.1	9.1	85.18	33.56	0.00	24.0
9	17.8	21.1	76.38	32.87	0.46	60.0
10	2.3	3.4	97.45	27.55	0.00	7.0
11	4.5	4.5	99.09	31.45	0.74	9.8
12	15.9	13.7	97.04	56.67	0.74	4.5

Table 7. Obtained performance values for each data base when more user-friendly hyper-rectangle regions are used.

ID	SVM	Interval rules				
database	(%) error	(%) error	Fd	Cv	Ov	NR
1	3.3	4.0	99.33	68.00	0.00	4.7
2	3.1	3.7	98.24	93.26	0.73	5.1
3	2.2	2.3	96.07	69.89	2.28	8.2
4	0.0	2.0	98.00	75.00	0.00	6.4
5	3.2	3.2	96.30	70.58	2.72	9.2
6	12.7	13.7	93.20	87.66	3.18	21.6
7	10.2	11.2	96.26	40.11	0.00	22.0
8	5.1	5.6	92.59	59.49	0.00	33.0
9	17.8	21.9	75.95	63.19	5.78	84.0
10	2.3	2.7	99.07	100.0	0.00	4.0
11	4.5	5.2	97.07	74.97	0.00	9.4
12	15.9	16.3	96.67	60.01	0.00	20.4

of the prototype vectors by the clustering algorithm directly affects the number and quality of the extracted rules. Therefore, it was necessary to apply the k-means algorithm several times, starting with different initial conditions, in order to chose the best cross-validation solution among all.

Partition schemes introduced in Section 4 for overcoming the randomness of the clustering algorithm were evaluated on trained SVMs on the data bases of the UCI repository. Tables 8 and 9 show the results obtained using partition scheme 3, the one with better performance on the benchmarks for ellipsoid and interval type equation rules, respectively. It can be observed that the results are comparable with those obtained using k-means for selecting prototypes, with the improvement that the mixed SVM – k-means clustering algorithm was applied only once. Thus, it has been demonstrated that it is possible to obtain an adequate rule base with a single application of the rule extraction algorithm.

6. Conclusions and Future Work

In this work, a rule extraction method for RBFNN, which uses as a core an algorithm for building ellipsoids based on support vectors was proposed. Based on the results obtained, it can be concluded that the extraction technique achieves consistent models with the RBFNN, without any previous requirement on either the training regime used or its architecture.

Furthermore, in order to provide SVMs with explanatory power, a method that converts the knowledge embedded in a trained SVM into a representation based on rules was developed. The experiments of the rule extraction method on data bases of different domains, show high levels of equivalence between the SVM and the extracted rule base.

Table 8. Performance values for each data base using partition scheme 3 when ellipsoid regions are used.

ID	SVM	Ellipsoid rules				
database	(%) error	(%) Ac	Fd	Cv	Ov	NR
1	3.3	1.3	96.67	62.00	0.00	4.0
2	3.1	3.6	97.35	87.83	0.00	4.5
3	2.2	2.3	97.74	64.62	0.56	7.0
4	0.0	0.0	100.0	15.50	0.00	5.8
5	3.2	3.2	97.21	78.27	0.95	8.6
6	12.7	14.2	92.74	63.42	2.89	21.3
7	10.2	12.3	95.72	17.11	0.53	16.0
8	5.1	12.9	86.11	37.13	0.23	12.0
9	17.8	21.3	78.34	31.48	0.00	61.0
10	2.3	5.6	94.67	44.90	0.00	5.0
11	4.5	5.3	98.32	38.88	0.00	10.2
12	15.9	16.7	97.78	51.11	0.00	5.2

Table 9. Performance values for each data base using partition scheme 3 when more user-friendly hyper-rectangle regions are used.

ID	SVM	Interval rules				
database	(%) error	(%) Ac	Fd	Cv	Ov	NR
1	3.3	3.3	96.67	72.00	0.00	5.0
2	3.1	3.5	98.39	91.80	0.59	9.0
3	2.2	2.5	97.75	69.18	0.56	14.6
4	0.0	0.0	100.0	70.50	6.50	6.8
5	3.2	3.3	96.30	71.69	0.93	11.2
6	12.7	14.2	90.43	81.31	3.91	34.5
7	10.2	11.7	92.51	33.12	2.14	28.0
8	5.1	4.4	92.82	88.99	6.48	15.0
9	17.8	19.6	78.70	62.26	2.31	65.0
10	2.3	2.3	99.07	90.74	0.00	10.0
11	4.5	5.9	96.10	73.09	0.00	9.3
12	15.9	16.7	95.56	60.00	0.00	21.4

Additionally, a new proposal has been derived that avoids randomness in the extraction rules algorithm. From the results obtained it is possible to conclude that the proposed modification indeed avoids randomness, and a consistent base of rules is obtained without process iteration.

Given the achievements of this work, future lines of research arise. For example, it would be interesting to study the ways to extend the rule extraction methods to regression problems. If this were achieved, a more versatile technique would be available for a greater number of cases. Another question that arises is the study of the possibility of using another representation language to express the new model, such as fuzzy rules generated from ellipsoids.

Acknowledgment

This study was partially supported by the Spanish MCyT grant TIC2002-04371-C02-01.

References

1. Andrews, R., Diederich, J. and Tickle, A.: A survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems* **8**(6) (1995), 373–389.
2. Blake, C. and Merz, C.: UCI repository of machine learning databases (1998).
3. Cherkassky, V. and Mulier, F.: *Learning from Data*. Wiley, New York, 1998.
4. Cortes, C. and Vapnik, V.: Support vector networks'. *Machine Learning* **20** (1995), 273–297.
5. Craven, M. and Shavlik, J.: Using neural networks for data mining. *Future Generation Computer Systems* **13** (1997), 211–229.
6. Cristianini, N. and Shawe-Taylor, J.: *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press, Cambridge, 2000.
7. Duda, R., Hart, P. and Stork, D.: *Pattern Recognition*, 2nd edn. Wiley, New York, Inc, 2001.
8. Fu, X. and Wang, L.: Rule extraction by genetic algorithms based on a simplified RBF neural network. In: *Proceedings of the Congress on Evolutionary Computation*, vol. 2, pp. 753–758, 2001.
9. Guyon, I., Martić, N. and Vapnik, V.: *Advances in Knowledge Discovery and Data Mining, Chapt. Discovery Information Patterns and Data Cleaning*. MIT Press, Cambridge 1996.
10. Huber, K. and Berthold, M.: Building precise classifiers with automatic rule extraction. In: *Proceedings of the IEEE International Conference on Neural Networks*, vol. 3. pp. 1263–1268, 1995.
11. Kaufmand, L. and Rousseeuw, P.: *Finding Groups in Data. An Introduction to Cluster Analysis*. Wiley, New York Inc, 1990.
12. Kecman, V.: *Learning and Soft Computing. Support Vector Machines, Neural Networks and Fuzzy Logic Models*. MIT Press, Cambridge, 2001.
13. Ma, J. and Zhao, Y.: OSU support vector machines toolbox, version 3.0'. http://www.csie.ntu.edu.tw/ cjlin/libsvm, 2004.
14. McGarry, K., Wermter, S. and MacIntyre, J.: Knowledge extraction from local function networks. In: *Proceedings of the International Joint Conference on Neural Networks*. pp. 765–770, 2001.
15. Mitra, S., Pal, S. and Mitra, P.: Data mining in soft computing framework: a survey. *IEEE Transactions on Neural Networks* **13**(1) (2002), 3–14.
16. Moody, J. and Darken, C.: Fast learning in networks of locally-tuned processing units. *Neural Computation* **1**(2) (1989), 281–294.
17. Nabney, I. and Bishop, C.: Netlab neural networks software. http://www.ncrg.aston.ac.uk/netlab, 2002.
18. Núñez, H., Angulo, C. and Català, A.: Rule extraction from support vector machines. In: Verleysen, M., (ed.), *ESANN 2002, European Symposium on Artificial Neural Networks*. Bruges, Belgium, pp. 107–112, d-facto publications, 2002.
19. Núñez, H., Angulo, C. and Català, A.: Hybrid architecture based on support vector machines. In: *Proceedings of the IWANN 2003*, Vol. 2686 of *Lecture Notes in Computer Science*. Menorca, Spain, pp. 646–653, 2003.

20. Orr, M.: Radial basis function networks. http://www.anc.ed.ac.uk/ mjo/rbf.html, 1999.
21. Schölkopf, B.: The Kernel trick for distances. In: *Neural Information Processing Systems, NIPS*, vol. 13, 2000.
22. Seitono, R., Leow, W. and Zurada, J.: Extraction rules from artificial neural networks for nonlinear regression. *IEEE Transactions on Neural Networks* **13**(3) (2002), 564–577.
23. Vapnik, V.: (1998) *Statistical Learning Theory*. New York, Wiley Inc,
24. Zhou, Z.-H.: (2004) Rule extraction: using neural networks or for neural networks? *Journal of Computer Science and Technology* **19**(2) (2004), 249–253.
25. Zhou, Z.-H., Jiang, Y. and Chen, S.-F.: Extracting symbolic rules from trained neural network ensembles. *AI Communications* **16**(1) (2003), 3–15.