Rule Based Learning Systems from SVM and RBFNN

Haydemar Núñez[1], Cecilio Angulo[2] and Andreu Català[2]

[1] Laboratorio de Inteligencia Artificial, Universidad Central de Venezuela. Caracas, Venezuela
hnunez@strix.ciens.ucv.ve

[2] Grup de Recerca en Enginyeria del Coneixement, Universitat Politècnica de Catalunya. 08800 Vilanova i la Geltrú, España
{andreu.catala,cecilio.angulo}@upc.es

Abstract. Two methods for the symbolic interpretation of both, Support Vector Machines (SVM) and Radial Basis Function Neural Networks (RBFNN) are proposed. These schemes, based on the combination of support vectors and prototype vectors by means of geometry produce rules in the form of ellipsoids and hyper-rectangles. Results obtained from a certain number of experiments on artificial and real data bases in different domains, allow concluding on the suitability of our proposal. Moreover, schemes incorporating into the SVMs the available prior domain knowledge expressed like symbolic rules are explored, obtaining an excellent performance.

1 Introduction

When neural networks are used for constructing classifiers, the generated models are of type black box, even if the obtained performance is good. With the purpose of turning into interpretable the classifier system, in the last 10 years rule extraction methods for trained neural networks have been developed [1],[4],[12],[17]. In the case of radial basis function neural networks (RBFNN) [9],[13], the rule extraction algorithms proposed usually impose restrictions over training, to avoid overlapping between classes or categories and thus facilitate the extraction process [7],[8],[11].

On the other hand, in the last 7 years, it has been demonstrated that support vector machines (SVM) [3],[5],[9], which are derived from the Statistical Learning Theory of V. N. Vapnik [18], have an excellent classification and approximation qualities on all kind of problems. However, as the neural networks, the generated models by these machines are difficult to understand from the point of view of the user.

The present work studies deeply the classifier function structure of the SVM and the RBFNN, in order to do the interpretation of these models. Starting with the support vectors selected by a SVM and prototype vectors generated by any clustering algorithm or by RBFNN, a rule extraction method for SVM is proposed. This method produces descriptions in the form of ellipsoids and it is

independent from the type of the used training. Furthermore, the interpretation of the rules is facilitated through a second derivation in the form of hyper-rectangles. As extension and original contribution of the work, the algorithm is modified for the exclusive use of the information supplied by a SVM and thus, eliminating the intrinsic variability in the classification made by clustering algorithms or by RBFNN is achieved

Additionally, starting with the RBFNN centers and using support vectors as classes delimiters, a rule extraction method for RBFNN is proposed. As the previous one, it produces descriptions in the form of ellipsoids and hyper-rectangles. This method solves the overlapping between classes without imposing restrictions on the network architecture nor its training regime.

Finally, since the final information will be expressed in the form of interpretable rules, a third contribution of this work is the analysis of knowledge insertion schemes into SVMs, when this knowledge is available in form of rules. The studied alternatives allow us to conclude that the access to this information by the SVM improves its final performance.

This paper is organized as follows: the rule extraction method from trained SVM is described in the next section, along with experimental results. Section 3 presents the rule extraction method for RBFNN. Section 4 describes prior knowledge integration methods into SVM. Finally, we present the conclusions and the future work.

2 Interpretation of Support Vector Machines

The solution provided by a SVM is one summation of kernel functions constructed on the base of the support vectors

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

The support vectors, sv, are the data nearest the separation limit between classes. They are the most informative samples for the classification task. By using geometric methods, these vectors with prototype vectors (generated by any clustering algorithm or by RBFNN) are combined in order to construct regions in the input space, which are later translated to if-then rules. These regions can be of two types: ellipsoids and hyper-rectangles. Ellipsoids generate rules whose antecedent is the mathematical equation of the ellipsoid. Hyper-rectangles (defined from parallel ellipsoids to the axes) generate rules whose premise is a set of restrictions on the values of each variable (Figure 1).

The ellipsoids must adjust to the form of the decision limit. In order to obtain one ellipsoid with these characteristics, the support vectors are used to determine the axes and vertices, as follow: First, the algorithm determines one prototype vector per class by using a clustering algorithm. This vector will be the center of ellipsoid. Then, the support vector of the same class with a value α smaller than C parameter and with the maximum distance to the prototype is chosen. The

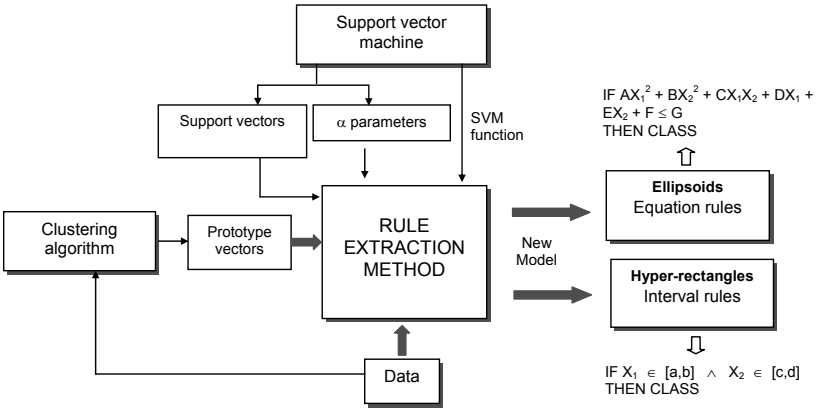

Fig. 1. Rule extraction method for SVMs.

straight line defined by these two points is the first axis of the ellipsoid. The rest of the axes and the associated vertices are determined by simple 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.

The number of regions necessary to describe to SVM model will depend on the form of the decision limit. It can happen that one ellipsoid is not sufficient to describe the data. Then, the rule base is determined by an iterative procedure that it begins with the construction of a general ellipsoid, which is divided in ellipsoids that adjust progressively to the form of the surface decision determined by SVM. To determine when divide an ellipsoid, a partition test is applied. If the test result is positive for one ellipsoid, the same will divide. We consider a partition test as positive 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.

Then, to define the number of rules per class the algorithm proceed 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 positive partition test and p regions with negative partition test. The latter ones 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 stopped once all partition tests are negative or the maximum number of iterations is reached. This process allows to control the number of generated rules.

After rules are extracted, the 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. Figure 2 shows an example of generated regions for each iteration.

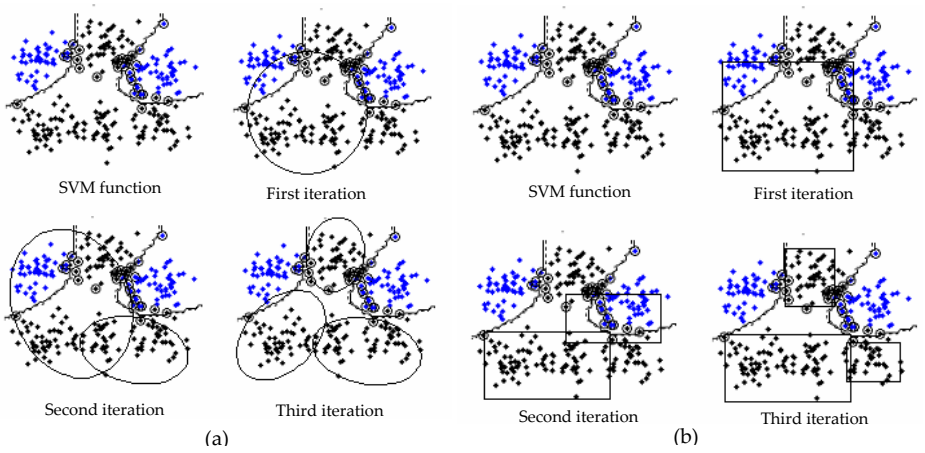

Fig. 2. Generated regions by rule extraction method. (a) Ellipsoids. (b) Hyper-rectangles.

2.1 Experiments

In order to evaluate the performance of the rule extraction algorithm, we carried out two kinds of experiments: with artificial data sets and data bases obtained from the UCI repository. The algorithms associated to the extraction method were development on Matlab v5.3. The training of the SVMs on the artificial data sets was made with "“Matlab SVM/K-SVCR Toolbox” software [2]; for data bases of repository UCI we used “OSU Support Vector Machines Toolbox v3.00” software [10]. We used the k-means clustering algorithm [6] to generate the prototype vectors.

Because the space is limited, only the experiments on UCI data bases are described. Table 1 shows the characteristics of the data bases that were used. The performance of the generated rules was quantified with the following measures:

- Error (Err): This is the classification error provided by the rules on test set.
- Consistency (Cn): It is the percentage of the test set for which network and the rule base output agree.

– Coverage (Cv): The percentage of examples from a test set covered by the rule base.
– Overlapping (Ov): It is the percentage of examples from test set covered by several rules.
– Number of extracted rules (NR).

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 & 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
10	Heart	270	13	real, symbolic & binary	2

Table 1. Data bases and their characteristics

Table 2 shows the prediction error of the SVM and the performance values of the extracted rule base for each problem. The results were obtained by averaging over stratified ten-fold cross-validation. It should be emphasized that the consistency percentage between the rule base and the SVM is very high. This values indicate that the rule base captures most of the information embedded in the SVM.

On the other hand, it was observed that the quality of the solution depends on the initial values for the centers; the selection of prototypes affects the number and quality of the extracted rules. Therefore, it was necessary to apply k-means several times, starting with different initial conditions and then choosing the best solution. Although is well known the dependency of the clustering final result to the random way as the prototypes are selected, this characteristic is in any way desirable. The solution to this problem is an opened working area and new proposal as the following presented means alternatives in positive sense.

2.2 Overcoming the randomness of clustering algorithm

In order to eliminate the sensitivity of the rule extraction method to the initial conditions of clustering, we proposed determine the initial centers for the clustering algorithm from the support vectors. Thus, if m is the number of necessary prototypes for iteration, then the m initial conditions for k-means are determined in the following form: for each class,

ID	SVM	Equation rules					Interval rules				
database	% Error	% Err	Cn	Cv	Ov	NR	% Err	Cn	Cv	Ov	NR
1	3.3	4.0	98.00	72.00	0.67	7.0	4.0	99.33	68.00	0.00	4.7
2	3.1	3.4	98.52	89.15	0.30	4.0	3.7	98.24	93.26	0.73	5.1
3	2.2	1.7	98.30	67.49	0.55	5.9	2.3	96.07	69.89	2.28	8.2
4	0.0	0.0	100.0	33.00	0.00	6.3	2.0	98.00	75.00	0.00	6.4
5	3.2	3.2	97.21	80.07	0.00	7.1	3.2	96.30	70.58	2.72	9.2
6	12.7	13.3	93.60	65.70	2.86	18.4	13.7	93.20	87.66	3.18	21.6
7	10.2	11.7	96.26	21.39	0.53	14.0	11.2	96.26	40.11	0.00	22.0
8	5.1	9.1	85.18	33.56	0.00	24.0	5.6	92.59	59.49	0.00	33.0
9	17.8	21.1	76.38	32.87	0.46	60.0	21.9	75.95	63.19	5.78	84.0
10	2.3	3.4	97.45	27.55	0.00	7.0	2.7	99.07	100.0	0.00	4.0
11	4.5	4.5	99.09	31.45	0.74	9.8	5.2	97.07	74.97	0.00	9.4
12	15.9	13.7	97.04	56.67	0.74	4.5	16.3	96.67	60.01	0.00	20.4

Table 2. Obtained performance values for each data base.

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

Three criteria were used to select the support vectors:

- Partition scheme 1: select those vectors with the smallest average dissimilarity with respect to data.
- Partition scheme 2: select the support vectors nearest to each other.
- Partition scheme 3: order the support vectors in descendent way according to α parameter and select the m first vectors.

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

These schemes was evaluated on trained SVMs on same data bases of UCI repository. Table 3 shows the obtained results when partition scheme 3 was used. We can observe that the results are comparable with those obtained by using k-means. Thus, it is possible obtaining a good rule base with a single application of the rule extraction algorithm.

ID	SVM	Equation rules					Interval rules				
database	% Error	% Err	Cn	Cv	Ov	NR	% Err	Cn	Cv	Ov	NR
1	3.3	1.3	96.67	62.00	0.00	4.0	3.3	96.67	72.00	0.00	5.0
2	3.1	3.6	97.35	87.83	0.00	4.5	3.5	98.39	91.80	0.59	9.0
3	2.2	2.3	97.74	64.62	0.56	7.0	2.5	97.75	69.18	0.56	14.6
4	0.0	0.0	100.0	15.50	0.00	5.8	0.0	100.0	70.50	6.50	6.8
5	3.2	3.2	97.21	78.27	0.95	8.6	3.3	96.30	71.69	0.93	11.2
6	12.7	14.2	92.74	63.42	2.89	21.3	14.2	90.43	81.31	3.91	34.5
7	10.2	12.3	95.72	17.11	0.53	16.0	11.7	92.51	33.12	2.14	28.0
8	5.1	12.9	86.11	37.13	0.23	12.0	4.4	92.82	88.99	6.48	15.0
9	17.8	21.3	78.34	31.48	0.00	61.0	19.6	78.70	62.26	2.31	65.0
10	2.3	5.6	94.67	44.90	0.00	5.0	2.3	99.07	90.74	0.00	10.0
11	4.5	5.3	98.32	38.88	0.00	10.2	5.9	96.10	73.09	0.00	9.3
12	15.9	16.7	97.78	51.11	0.00	5.2	16.7	95.56	60.00	0.00	21.4

Table 3. Performance values for each data base using partition scheme 3.

3 Interpretation of Radial Basis Function Neural Networks

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

$$f(\mathbf{x}, \mathbf{w}, \mathbf{v}) = \sum_{i=1}^{m} w_k \phi_k(\mathbf{x}, \mathbf{v_k}) + w_0 \tag{2}$$

The nonlinear activation function ϕ_k expresses the 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 (receptive field) on which the neurone produces a appreciable activation value. If the common case when the gaussian function is used, the center$\mathbf{v_k}$ of the function ϕ_k defines the prototype of input cluster k and the variance σ_k the size of the covered region in the input space.

The local nature of RBF networks makes them an interesting platform for performing rule extraction. However, the basis functions overlap to some grade in order to give a relatively smooth representation of the distribution of training data [9],[13]. This overlapping is a shortcoming for rule extraction.

Few rule extraction methods directed to RBFNN have been developed [7],[8],[11]. To avoid the overlapping, most of them are using special training regimes or special architectures in order to guarantee that RBF nodes are assigned and used by a single class. Our proposal for rule extraction does not suppose any training method or special architecture; it extracts rules from an ordinary RBFNN. In order to solve the overlapping, a support vector machine (SVM) is used as a frontier pattern selector.

The rule extraction method for RBFNN derives descriptions in the form ellipsoids and hyper-rectangles. Therefore, the algorithm for constructing an

ellipsoid for SVM is used; the prototypes are replaced by the RBF centers. In this case, support vectors establish the boundaries between classes.

Initially, by assign each input pattern to their closest center of RBF node according to the Euclidean distance function a partition of the input space is made. When assigning a pattern to its closest center, this one will be assigned to the RBF node that will give the maximum activation value for that pattern. From these partitions the ellipsoids are constructed.

Next, a class label is assigned for each center of RBF units. Output value of the RBF network for each center is used in order to determine this class label.

Then, for each node an ellipsoid with the associated partition data is constructed. Once determined the ellipsoids, they are transferred to rules.

This procedure will generate a rule by each node. Nevertheless, data of another classes could be present in the partition of RBF unit. For these data, we determine the mean of each class. Each mean is used as center of its class to construct an ellipsoid with the associated data.

In order to eliminate or to reduce the overlapping that could exist between ellipsoids of different classes, an overlapping test is applied. Overlapping test verifies if a support vector from other class exits within the ellipsoid. Because the support vectors are the points nearest to decision limit, the presence of these within an ellipsoid of different class is a good indicator of overlapping. If the overlapping test is positive, the ellipsoid is divided.

This procedure will allow refining the rule base in order to reduce the overlapping between classes. When the ellipsoids are divided, more specific rules are generated to exclude data from other classes. This procedure can be executed of iterative form; depending of the number of iterations, two or more partitions by ellipsoids can be obtained. User can establish the maximum number of iterations. Thus, its possible to control the number of generated rules by RBF node.

3.1 Experiments

In order to evaluate the performance of the rule extraction algorithm, we carried out two kinds of experiments: with artificial data sets and data bases obtained from the UCI repository. The algorithms associated to the extraction method were development on Matlab v5.3. Again, only experiments on data bases from UCI repository are described.

We used 6 databases of this repository and the same performance parameters. With the purpose to validate the hypothesis about of rule extraction method is independence of the used training techniques, two different training procedures are used: the Netlab software [14] which uses the EM algorithm to determine the RBF centers, and the Orr software [15], which uses forward selection.

Tables 4 and 5, show the prediction error of the RBF network and the performance values of the extracted rule base. Results were obtained by averaging over stratified ten-fold cross-validation. We can observe a high agreement between the results obtained from the rule base and those obtained from the RBF network.

However, because Orr method needs to use more hidden units to obtain better performance, it produces a greater rule base.

ID	RBF	RBF	Equation rules					Interval rules				
database	Nodes	% Error	% Err	Cn	Cv	Ov	NR	% Err	Cn	Cv	Ov	NR
1	4.5	4.0	3.3	96.67	64.67	0.00	6.8	4.0	97.33	70.67	0.00	6.5
2	2.2	2.9	3.2	98.24	91.21	1.76	5.7	4.1	96.78	95.01	2.93	14.5
3	3.0	1.1	3.8	97.78	66.43	2.75	9.7	4.6	95.38	81.30	7.32	10.7
4	5.4	2.0	2.0	96.00	17.00	0.00	6.9	6.0	91.50	62.50	4.00	10.2
5	9.3	6.5	6.0	94.91	80.99	2.29	16.3	5.6	94.44	79.07	6.06	16.5
10	6.0	4.8	6.0	95.14	63.19	0.93	14.0	2.7	97.91	100.0	0.00	11.0

Table 4. Obtained results from data sets (with Netlab software).

ID	RBF	RBF	Equation rules					Interval rules				
database	Nodes	% Error	% Err	Cn	Cv	Ov	NR	% Err	Cn	Cv	Ov	NR
1	5.1	3.3	2.7	96.67	72.00	0.00	9.2	3.3	94.67	74.67	0.00	8.9
2	21.5	3.4	3.5	97.22	83.41	0.43	26.4	4.9	96.19	95.31	3.95	28.2
3	15.2	3.9	3.9	93.26	63.20	0.62	38.1	6.2	93.30	85.38	7.26	86.2
4	12.4	0.0	4.0	96.00	30.00	0.00	14.7	8.5	91.50	91.00	30.50	19.6
5	29.32	6.2	6.4	88.87	61.41	0.45	33.0	5.4	88.87	72.23	0.90	33.0
10	12.0	5.0	6.9	90.97	63.19	3.93	23.0	6.4	92.36	100.0	57.40	33.0

Table 5. Obtained results from data sets (with Orr software).

4 Inserting Prior Knowledge in Support Vector Machines

To incorporate prior knowledge in the support vector machines, different techniques have been used: By means of generated virtual examples from transformation functions applied to the data [20] or the support vectors [16], designing kernel functions that adapt to the problems [5], and adding new restrictions of the optimization problem [19]. Nevertheless, sometimes the prior knowledge is difficult to formalize like a transformation or kernel function; this knowledge can be expressed as a set of symbolic rules giving by experts as follow,

$$if \quad x_1 \in [a_1, b_1] \wedge x_2 \in [a_2, b_2] \wedge \cdots \wedge x_m \in [a_m, b_m] \quad then \quad Class \qquad (3)$$

In this case, how could we integrate this knowledge in a SVM? We propose do it by means a strategy similar to virtual example method.

The prior knowledge expressed as a rule defines a convex region in the input space (in the form of a hyper-rectangle). This convex region is defined by a set of vertices, which provide information on the limits of the associated rule. These vertices can be used as virtual examples and add them to the learning set. However, these samples would have a special treatment because the knowledge that they provide is correct. Then, let the training set $D = \{(\mathbf{x_i}, y_i)\}_{i=1,\ldots,n}$ and let the virtual example set $V = \{(\mathbf{x}_j, y_j)\}_{j=1,\ldots,d=2^m}$, to insert this knowledge in a SVM three schemes are proposed

Insertion method 1. A new restriction is added to optimization problem which supposes that virtual examples can be classified without error:

$$\begin{aligned} &\min \frac{1}{2}\|\mathbf{w}\|^2 + C \cdot \sum_{i=1}^{n} \xi_i \\ &\text{s.t.} \begin{cases} y_i \ (\mathbf{w} \cdot \mathbf{x_i}) + b \geq 1 - \xi_i & (\mathbf{x_i}, y_i) \in D \\ y_j \ (\mathbf{w} \cdot \mathbf{x_j}) + b \geq 1 & (\mathbf{x_j}, y_j) \in V \\ \xi_i \geq 0 & (\mathbf{x_i}, y_i) \in D \end{cases} \end{aligned} \tag{4}$$

Insertion method 2. Two different parameters for error control are defined

$$\begin{aligned} &\min \frac{1}{2}\|\mathbf{w}\|^2 + C_d \cdot \sum_{i=1}^{n} \xi_i^d + C_v \cdot \sum_{j=1}^{d} \xi_j^v \\ &\text{s.t.} \begin{cases} y_i \ (\mathbf{w} \cdot \mathbf{x_i}) + b \geq 1 - \xi_i^d & (\mathbf{x_i}, y_i) \in D \\ y_j \ (\mathbf{w} \cdot \mathbf{x_j}) + b \geq 1 - \xi_j^v & (\mathbf{x_j}, y_j) \in V \\ \xi_i^d \geq 0 & (\mathbf{x_i}, y_i) \in D \\ \xi_j^v \geq 0 & (\mathbf{x_j}, y_j) \in V \end{cases} \end{aligned} \tag{5}$$

The C_d y C_v parameters determine a balance between the information from training data and the prior knowledge.

Insertion method 3. Train a SVM with the vertices of the rules of one class and the data from another classes. This pre-processing generates a set of virtual supports by class. Then, train a SVM with the all learning set and the generated virtual support vectors from this pre-processing. In this form, the part of the prior knowledge that would be more informative for the classification task is added to the data (the vertices nearest to the surface limit).

4.1 Experiments

In order to evaluate the rule insertion methods, we applied them to the three MONK problems of the UCI repository, because they have a defined domain theory. To verify if it is possible to improve the SVM performance when inserting the domain knowledge, the follow procedure was realised: First, a SVM without added knowledge was trained. Then, we train a SVM using the rule insertion methods. This procedure was repeated 50 times and we determined the average values on the following parameters:

– Training set error (EE)
– Test set error (ET)
– Number of support vectors from MONK class (Sv1)
– Number of support vectors from NO-MONK class (Sv2)

The used values of the C_d and C_v parameters guarantee a greater weight to the errors associated for the virtual examples. Table 6 shows the obtained results. We can observed that the rule insertion methods improve the original SVM performance,

Strategy	Monk1				Monk2				Monk3			
	EE	ET	SV1	SV2	EE	ET	SV1	SV2	EE	ET	SV1	SV2
No knowledge	0.00	5.90	28.24	23.22	0.80	14.60	34.22	29.74	2.30	5.40	14.04	14.44
Method 1	0.00	3.20	40.08	23.80	0.30	8.80	47.52	30.80	2.10	3.40	14.00	13.36
Method 2	0.00	3.20	40.08	23.80	0.30	8.80	47.84	30.72	2.00	3.30	13.04	13.70
Method 3	0.00	2.90	38.44	23.96	0.30	8.60	44.20	30.90	2.10	2.90	13.52	13.28

Table 6. Obtained results by applying the insertion methods on Monk databases.

5 Conclusions and Future Work

With the purpose of providing SVMs with explanation 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 artificial data sets and with data bases of different domains, show high levels of equivalence between the SVM and the extracted rule base.

Additionally, a rule extraction method for RBFNN which uses as core the algorithm for constructing an ellipsoid proposed for SVM was developed. Based on the obtained results, can be conclude that the extraction technique achieves derive consistent models with the RBFNN, without any previous requirement over neither the used training regime nor its architecture.

The possibility of adding prior knowledge expressed as symbolic rules in a SVM was established, using schemes based on virtual examples which are generated from the associated vertices with hyper-rectangles related with the rules.

Given the achievements of this work, it is possible to raise new problems. For example, it would be interesting to study the ways to extend the rule extraction methods to regression problems. If it is reached, a more versatile technique would be available for a major number of cases. Another question coming up is the study of the possibility of using another representation language to express the new model, such as fuzzy rules generated from ellipsoids.

References

1. R. Andrews, J. Diederich, and A. Tickle. A survey and critique of techniques for extracting rules from trained artificial neural networks. *Knowledge-Based Systems*, 8(6):373–389, 1995.
2. Cecilio Angulo. Matlab svm/k-svcr toolbox. http://webesaii.upc.es/usr/cecilio/software.htm.
3. C. Cortes and V. Vapnik. Support vector networks. *Machine Learning*, 20:273–297, 1995.
4. M. Craven and Jude Shavlik. Using neural networks for data mining. *Future Generation Computer Systems*, 13:211–229, 1997.
5. N. Cristianini and J. Shawe-Taylor. *An introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University press 2000, 2000.
6. R. Duda, P. Hart, and D. Stork. *Pattern Recognition*. John Wiley & Sons, Inc, second edition, 2001.
7. X. Fu and L. Wang. Rule extraction by genetic algorithms based on a simplified rbf neural network. In *Proceedings of the Congress on Evolutionary Computation*, volume 2, pages 753–758, 2001.
8. K. Huber and M. Berthold. Building precise classifiers with automatic rule extraction. In *Proceedings of the IEEE International Conference on Neural Networks*, volume 3, pages 1263–1268, 1995.
9. V. Kecman. *Learning and Soft Computing. Support Vector Machines, Neural Networks and Fuzzy Logic Models*. MIT Press, 2001.
10. J. Ma and Y. Zhao. Osu support vector machines toolbox, version 3.0. http://www.csie.ntu.edu.tw/ cjlin/libsvm.
11. K. McGarry, S. Wermter, and J. MacIntyre. Knowledge extraction from local function networks. In *Proceedings of the International Joint Conference on Neural Networks*, pages 765–770, 2001.
12. S. Mitra, S.K. Pal, and P. Mitra. Data mining in soft computing framework: A survey. *IEEE Transactions on Neural Networks*, 13(1):3–14, 2002.
13. J. Moody and C. Darken. Fast learning in networks of locally-tuned processing units. *Neural computation*, 1(2):281–294, 1989.
14. I. Nabney and C. Bishop. Netlab neural networks software. http://www.ncrg.aston.ac.uk/netlab.
15. M. Orr. Radial basis function networks. http://www.anc.ed.ac.uk/ mjo/rbf.html.
16. B. Schölkopf, C. Burges, and V. Vapnik. Incorporatinig invariances in support vector learning machines. *Artificial Neural Networks- ICANN'96*, pages 47–52, 1996.
17. R. Seitono, W. Leow, and J. Zurada. Extraction rules from artificial neural networks for nonlinear regression. *IEEE Transactions on Neural Networks*, 13(3):564–577, 2002.
18. V.N. Vapnik. *Statistical Learning Theory*. John Wiley & Sons, Inc, 1998.
19. X. Zhang. Using class-center vectors to build support vector machines. In *Proceedings of the IEEE Conference on Neural Networks for Signal Processing*, pages 3–11, 1999.
20. Q. Zhao and J.C. Principe. Improving atr performance by incorporating virtual negative examples. In *Proceedings of the International Joint Conference on Neural Networks*, volume 5, pages 3198–3203, 1995.