Hybrid Learning Systems Based on Support Vector Machines and Radial Basis Function Neural Networks

Haydemar Núñez
Laboratorio de Inteligencia Artificial
Facultad de Ciencias. Universidad Central de Venezuela. Caracas, Venezuela
hnunez@strix.ciens.ucv.ve

Cecilio Angulo, Andreu Català,
ERIC – Engineering & Research in Computational Intelligence
Technical University of Catalonia. Vilanova i la Geltrú, Spain
cecilio.angulo@upc.es, andreu.catala@upc.es,

Abstract

Two methods are proposed for the symbolic interpretation of both Support Vector Machines (SVM) and Radial Basis Function Neural Networks (RBFNN). 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 databases in different domains allow conclusions to be drawn on the suitability of our proposal. Moreover, schemes that incorporate the available prior domain knowledge expressed as symbolic rules into SVMs are explored, with excellent performances being obtained.
Keywords: Artificial Intelligence, Support Vector Machines, Neural Networks, Hybrid Architectures

Resumen

En este trabajo se proponen dos métodos para la interpretación simbólica de máquinas de soporte vectorial (SVM) y redes neuronales de función de base radial (RBFNN), respectivamente. Ambos esquemas se basan en la combinación, mediante geometría, de los vectores de soporte generados por una SVM y vectores prototipos o centros de una RBFNN, para producir descripciones en la forma de elipsoides e hiper-rectángulos. Los resultados de los numerosos experimentos realizados sobre bases de datos artificiales y reales de diferentes dominios, nos permiten concluir sobre la viabilidad de la propuesta. También, se exploran esquemas para la inserción, en máquinas de soporte vectorial, del conocimiento previo disponible expresado como reglas simbólicas.
Palabras claves: Inteligencia Artificial, Máquinas de Soporte Vectorial, Redes Neuronales, Arquitecturas Híbridas

1. INTRODUCTION

When neural networks are used for constructing classifiers, black-box type models are generated, even if the obtained performance is good. With the aim of facilitating the interpretation of the classifier system, over the last 10 years rule extraction methods for trained neural networks have been developed [1],[6],[14],[21] and [22]. In the case of radial basis function neural networks (RBFNN) [11],[15],[19], the rule extraction algorithms proposed usually impose restrictions over training in order to avoid overlapping between classes or categories and thus facilitate the extraction process [4],[9],[10] and [13].

On the other hand, over the last 7 years it has been demonstrated that the support vector machines (SVM) [5],[7],[11] derived from V.N. Vapnik's Statistical Learning Theory [23], have excellent classification and approximation qualities for all kinds of problems. However, as in the case of the neural networks, the models generated by these machines are difficult to understand from a user's point of view.

In order to interpret these models, the present work studies the classifier function structure of the SVM and the RBFNN in depth [17]. 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 of the type of training used. Furthermore, the interpretation of the rules is facilitated through a second derivation in the form of hyper-rectangles. As an original contribution and extension of the work, the algorithm is modified for the exclusive use of the information supplied by a SVM, thus achieving the elimination of the intrinsic variability in the classification made by clustering algorithms or by RBFNN.

Additionally, starting with the RBFNN centres and using support vectors as class delimiters, a rule extraction method for RBFNN is proposed. As in the previous case, 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 or 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 schemes for inserting knowledge into SVMs, when this knowledge is available in form of rules. The alternatives studied allow us to conclude that access to this information by the SVM improves its final performance.

This paper is organized as follows: the rule extraction method from trained SVMs is described in the next section, along with the 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 future work.

2. INTERPRETATION OF SUPPORT VECTOR MACHINES

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

$$f_a(\boldsymbol{x}) = sign\left(\sum_{i=1}^{sv} \alpha_i y_i K(\boldsymbol{x}, \boldsymbol{x}_i) + b\right) \tag{1}$$

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 follows: first, the algorithm determines one prototype vector per class by using a clustering algorithm. This vector will be the centre of the ellipsoid. Then, a support vector of the same class with a value α smaller than parameter C and with the maximum distance to the prototype is chosen. The 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 lines parallel to the axes are used to define the axes of the associated ellipsoid.

The number of regions necessary to describe the SVM model will depend on the form of the decision limit. One ellipsoid may not be sufficient to describe the data. Then, the rule base is determined by an iterative procedure that begins 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. To determine when to divide an ellipsoid, a partition test is applied. If the test result is positive for one ellipsoid, the latter is divided. We consider a partition test to be positive if the generated prototype belongs to another class, if one of the vertices belongs to another class or if a support

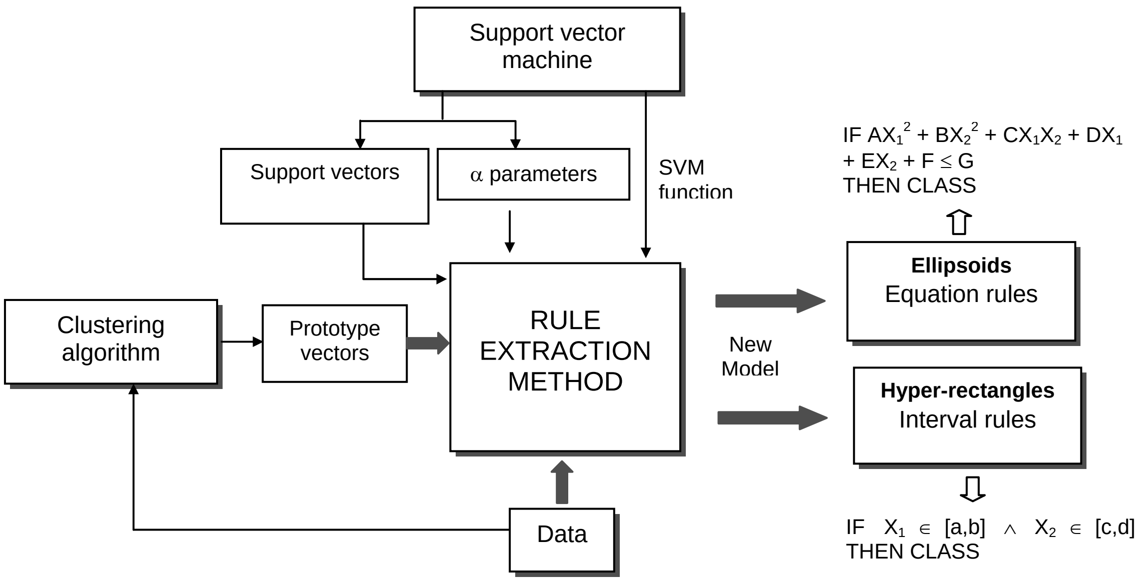


Figure 1. Rule extraction method for SVMs

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, in order to define the number of rules per class the algorithm proceeds as follows:

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 into 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 is stopped once all partition tests are negative or the maximum number of iterations has been reached. This process allows the number of rules generated to be controlled.

After the 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-neighbour 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 regions generated for each iteration.

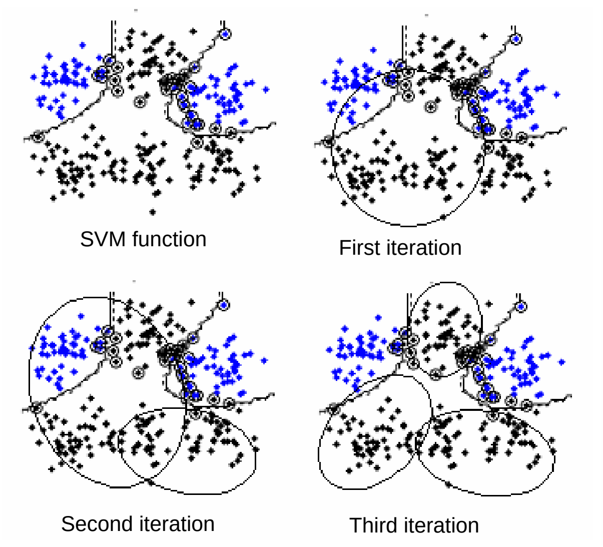
2.1 Experiments

In order to evaluate the performance of the rule extraction algorithm, we carried out two kinds of experiments: with artificial data sets and databases obtained from the UCI repository [3]. The algorithms associated to the extraction method were developed on Matlab v5.3. The training of the SVMs on the artificial data sets was carried out with "Matlab SVM/K-SVCR Toolbox" software [2]; for databases from the repository UCI we used "OSU Support Vector Machines Toolbox v3.00" software [12]. We used the k-means clustering algorithm [8] to generate the prototype vectors.

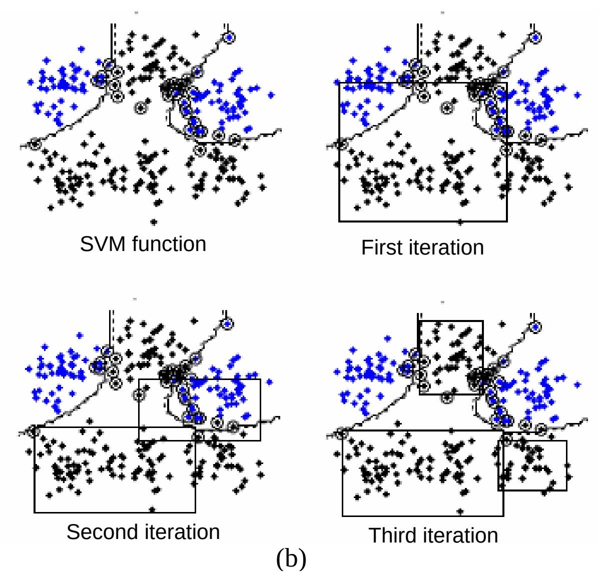
Because the space is limited, only the experiments on UCI databases are described. Table 1 shows the characteristics of the databases that were used. The performance of the rules generated was quantified using the following measures:

- *Error (Err)*: This is the classification error provided by the rules on the test set.
- *Consistency (Cn)*: This is the percentage of the test set for which the network and the rule base output agree.
- *Coverage (Cv)*: This is the percentage of examples from the test set covered by the rule base.
- *Overlapping (Ov)*: This is the percentage of examples from the test set covered by several rules.
- *Number of extracted rules (NR).*

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. These values indicate that the rule base captures most of the information embedded in the SVM.



(a)



(b)

Figure 2. Regions generated by the rule extraction method
(a) Ellipsoids. (b) Hyper-rectangles

Table 1. Databases and their characteristics

ID	Database	Data	Attributes	Type of attributes	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

On the other hand, it was observed that the quality of the solution depends on the initial values for the centres; 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 the dependency of the clustering final result on the random way the prototypes are selected is well known, this characteristic is never desirable. The solution to this problem is an opened working area and a new proposal such as the following one presented here means positive alternatives.

2.2 Overcoming the Randomness of Clustering Algorithms

In order to eliminate the sensitivity of the rule extraction method to the initial conditions of clustering, we proposed the initial centres for the clustering algorithms from the support vectors should be determined. 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:

By each class

- Select *m* support vectors with same class label.
- Assign examples to their closest support vector according to the Euclidean distance function.

Table 2. Performance values obtained for each database.

ID database	Error SVM	Equation rules					Interval rules				
		Error	Cn.	Cv.	Ov.	NR	Error	Cn.	Cv.	Ovl.	NR
1	0.033	0.040	98.00	72.00	0.67	7.0	0.040	99.33	68.00	0.00	4.7
2	0.031	0.034	98.52	89.15	0.30	4.0	0.037	98.24	93.26	0.73	5.1
3	0.022	0.017	98.30	67.49	0.55	5.9	0.023	96.07	69.89	2.28	8.2
4	0.000	0.000	100.00	33.00	0.00	6.3	0.020	98.00	75.00	0.00	6.4
5	0.032	0.032	97.21	80.07	0.00	7.1	0.032	96.30	70.58	2.72	9.2
6	0.127	0.133	93.60	65.70	2.86	18.4	0.137	93.20	87.66	3.18	21.6
7	0.102	0.117	96.26	21.39	0.53	14.0	0.112	96.26	40.11	0.00	22.0
8	0.051	0.091	85.18	33.56	0.00	24.0	0.056	92.59	59.49	0.00	33.0
9	0.178	0.211	76.38	32.87	0.46	60.0	0.219	75.95	63.19	5.78	84.0
10	0.023	0.034	97.45	27.55	0.00	7.0	0.027	99.07	100.00	0.00	4.0
11	0.045	0.045	99.09	31.45	0.74	9.8	0.052	97.07	74.97	0.00	9.4
12	0.159	0.137	97.04	56.67	0.74	4.5	0.163	96.67	60.01	0.00	20.4

- Once the initial partitions have been established, the mean of all instances in each partition is calculated.
- 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 the data.
- *Partition scheme 2*: select the support vectors nearest to each other.
- *Partition scheme 3*: organize the support vectors in descending order according to α parameter and select the m first vectors.

In all 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 were evaluated on trained SVMs on the same databases from the UCI repository. Tables 3, 4 and 5 show the obtained results when these partition schemes were used. We can observe that the results are comparable with those obtained by using k-means. Thus, it is possible to obtain a good rule base with a single application of the rule extraction algorithm.

Table 3. Performance values for each database using partition scheme 1

ID database	Error SVM	Equation rules					Interval rules				
		Error	Cn.	Cv.	Ov.	NR	Error	Con.	Cub.	Sol.	NR
1	0.033	0.040	99.33	67.33	1.33	7.0	0.047	98.67	67.33	1.33	4.7
2	0.031	0.032	98.07	87.83	0.14	4.4	0.032	98.53	89.74	1.02	10.0
3	0.022	0.023	96.63	67.91	1.11	6.0	0.028	97.18	75.34	3.89	10.9
4	0.000	0.020	98.00	25.00	0.00	4.9	0.020	98.00	70.00	2.00	7.7
5	0.032	0.028	96.73	76.32	2.74	8.0	0.042	96.30	72.03	3.18	10.2
6	0.127	0.131	91.00	59.98	2.75	21.1	0.146	92.75	84.91	4.47	24.0
7	0.102	0.182	88.77	19.79	5.88	12.0	0.118	90.37	71.23	1.60	34.0
8	0.051	0.123	86.34	52.91	0.23	28.0	0.141	86.34	69.91	6.48	34.0
9	0.178	0.201	78.24	34.03	1.38	64.0	0.231	78.00	56.71	2.31	72.0
10	0.023	0.025	96.99	40.05	0.00	8.0	0.018	99.53	95.60	0.00	8.0
11	0.045	0.046	99.09	29.23	0.00	10.2	0.052	95.96	74.97	0.00	9.5
12	0.159	0.151	96.30	52.22	0.37	4.6	0.167	96.30	57.04	0.37	21.0

Table 4. Performance values for each database using partition scheme 2

ID database	Error SVM	Equation rules					Interval rules				
		Error	Cn.	Cv.	Ov.	NR	Error	Con.	Cub.	Sol.	NR
1	0.033	0.047	98.67	69.33	0.67	6.7	0.040	99.33	68.67	0.67	4.8
2	0.031	0.035	98.38	89.15	0.29	4.3	0.035	98.09	91.36	0.73	9.3
3	0.022	0.028	98.06	63.36	0.55	7.2	0.028	97.22	73.64	1.11	11.7
4	0.000	0.020	98.00	25.00	0.00	4.9	0.020	98.00	68.00	2.00	8.2
5	0.032	0.032	96.28	80.52	0.93	7.9	0.042	95.37	69.16	3.65	10.6
6	0.127	0.149	90.85	65.16	3.90	23.2	0.157	90.01	88.39	7.63	25.9
7	0.102	0.176	90.37	17.65	0.00	20.0	0.197	89.30	36.89	3.21	22.0
8	0.051	0.171	84.25	59.03	3.24	12.0	0.078	90.74	90.51	13.4	15.0
9	0.178	0.215	74.53	33.10	1.62	70.0	0.231	78.00	56.71	2.31	72.0
10	0.023	0.048	96.99	36.00	0.00	9.0	0.018	99.54	94.90	0.00	9.0
11	0.045	0.046	89.09	27.11	0.00	10.1	0.052	95.96	74.97	0.00	9.5
12	0.159	0.137	95.56	54.44	0.74	4.8	0.166	95.18	57.78	0.74	20.3

Table 5. Performance values for each database using partition scheme 3

ID database	Error SVM	Equation rules					Interval rules				
		Error	Cn.	Cv.	Ov.	NR	Error	Con.	Cub.	Sol.	NR
1	0.033	0.013	96.67	62.00	0.00	4.0	0.033	96.67	72.00	0.00	5.0
2	0.031	0.036	97.35	87.83	0.00	4.5	0.035	98.39	91.80	0.59	9.0
3	0.022	0.023	97.74	64.62	0.56	7.0	0.025	97.75	69.18	0.56	14.6
4	0.000	0.00	100.00	15.50	0.00	5.8	0.00	100.00	70.50	6.50	6.8
5	0.032	0.032	97.21	78.27	0.95	8.6	0.033	96.30	71.69	0.93	11.2
6	0.127	0.142	92.74	63.42	2.89	21.3	0.142	90.43	81.31	3.91	34.5
7	0.102	0.123	95.72	17.11	0.53	16.0	0.117	92.51	33.12	2.14	28.0
8	0.051	0.129	86.11	37.13	0.23	12.0	0.044	92.82	88.99	6.48	15.0
9	0.178	0.213	78.34	31.48	0.00	61.0	0.196	78.70	62.26	2.31	65.0
10	0.023	0.056	94.67	44.90	0.00	5.0	0.023	99.07	90.74	0.00	10.0
11	0.045	0.053	98.32	38.88	0.00	10.2	0.059	96.10	73.09	0.00	9.3
12	0.159	0.167	97.78	51.11	0.00	5.2	0.167	95.56	60.00	0.00	21.4

3. INTERPRETATION OF RADIAL BASIS FUNCTION NEURAL NETWORKS

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

$$f(\boldsymbol{x},\boldsymbol{w},\boldsymbol{v})=\sum_{k=1}^{m} w_k \phi_k(\boldsymbol{x},\boldsymbol{v}_k)+w_0 \qquad (2)$$

The nonlinear activation function ϕ_k expresses the similarity between any input pattern **x** and the centre $\mathbf{v}_k$ by means of a distance measure. Each function ϕ_k defines a region in the input space (receptive field) on which the neuron 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 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 degree in order to give a relatively smooth representation of the distribution of training data [11],[15]. This overlapping is a shortcoming for rule extraction.

Few rule extraction methods for RBFNN have been developed [4],[9],[10] and [13]. In order avoid the overlapping, most of them use special training regimes or special architectures so as to guarantee that the RBF nodes are assigned and used by a single class. Our proposal for rule extraction does not suppose any training methods 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 of ellipsoids and hyper-rectangles. Therefore, the algorithm for constructing an ellipsoid for SVM is used; the RBF centres replace the prototypes. In this case, support vectors establish the boundaries between classes.

Initially, by assigning each input pattern to its closest centre of RBF node according to the Euclidean distance function, a partition of the input space is made. When assigning a pattern to its closest centre, the former 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 centre of RBF units. The output value of the RBF network for each centre is used in order to determine this class label.

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

This procedure will generate a rule for each node. Nevertheless, other classes of data could be present in the partition of the RBF unit. For these data we determine the mean of each class. Each mean is used as a centre of its class in order 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 tests verify whether a support vector from another class exits within the ellipsoid. Because the support vectors are the points nearest to the decision limit, the presence of these vectors within an ellipsoid of a different class is a good indicator of overlapping. If the overlapping test is positive, the ellipsoid is divided.

This procedure will allow the rule base to 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. This procedure can be executed in an iterative form; depending of the number of iterations, two or more partitions by ellipsoids can be obtained. The user can establish the maximum number of iterations. Thus, it is possible to control the number of rules generated by the 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 databases obtained from the UCI repository. The algorithms associated to the extraction method were developed on a Matlab v5.3. Again, only experiments on databases from the UCI repository are described.

We used 6 databases from this repository and the same performance parameters. With the purpose of validating the hypothesis of the rule extraction method independent of the training techniques used, two different training procedures were used: the Netlab software [16], which uses the EM algorithm to determine the RBF centres, and the Orr software [18], which uses forward selection.

Tables 6 and 7, 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 the Orr method needs to use more hidden units to obtain a better performance, it produces a greater rule base.

Table 6. Results obtained from data sets (with Netlab software)

ID	RBF nodes	RBF error	Equation rules					Interval rules				
			Err	Cn.	Cv.	Ov.	NR	Err	Cn.	Cv	Ov	NR
1	4.5	0.040	0.033	96.67	64.67	0.00	6.8	0.040	97.33	70.67	0.00	6.5
2	2.2	0.029	0.032	98.24	91.21	1.76	5.7	0.041	96.78	95.01	2.93	14.5
3	3.0	0.011	0.038	97.78	66.43	2.75	9.7	0.046	95.38	81.30	7.32	10.7
4	5.4	0.020	0.020	96.00	17.00	0.00	6.9	0.060	91.50	62.50	4.00	10.2
5	9.3	0.065	0.060	94.91	80.99	2.29	16.3	0.056	94.44	79.07	6.06	16.5
10	6.0	0.048	0.060	95.14	63.19	0.93	14.0	0.027	97.91	100.0	0.00	11.0

Table 7. Results obtained from data sets (with Orr software)

ID	RBF nodes	RBF error	Equation rules					Interval rules				
			Err	Cn.	Cv	Ov.	NR	Err	Cn.	Cv.	Ov.	NR
1	5.1	0.033	0.027	96.67	72.00	0.00	9.2	0.033	94.67	74.67	0.00	8.9
2	21.5	0.034	0.035	97.22	83.41	0.43	26.4	0.049	96.19	95.31	3.95	28.2
3	15.2	0.039	0.039	93.26	63.20	0.62	38.1	0.062	93.30	85.38	7.26	86.2
4	12.4	0.000	0.040	96.00	30.00	0.00	14.7	0.085	91.50	91.00	30.50	19.6
5	29.2	0.062	0.064	88.87	61.41	0.45	33.0	0.054	88.87	72.23	0.90	33.0
10	12.0	0.050	0.069	90.97	63.19	3.93	23.0	0.064	92.36	100.0	57.40	33.0

4. INSERTING PRIOR KNOWLEDGE IN SUPPORT VECTOR MACHINES

To insert prior knowledge into the support vector machines, different techniques have been used: by means of generated virtual examples from transformation functions applied to the data [26] or the support vectors [20], designing kernel functions that adapt to the problems [7], [24], and adding new restrictions of the optimization problem [25]. Nevertheless, sometimes the prior knowledge is difficult to formalize as a transformation or kernel function; this knowledge can be expressed as a set of symbolic rules, which experts give as follows:

$$\textit{IF } x_1 \in [a_1, b_1] \wedge x_2 \in [a_2, b_2] \wedge \ldots \wedge x_m \in [a_m, b_m] \textit{ THEN class}$$

How could we integrate this knowledge in a SVM in this case? We propose doing it by means of a strategy similar to the 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 added to the learning set. However, these samples would be treated specially because the knowledge that they provide is correct. Then, given the training set $D = \{(\mathbf{x}_i, y_i) i = 1..n\}$ and the virtual example set $V = \left\{(\mathbf{x}_j, y_j) j = 1..d, d = 2^m\right\}$, three schemes are proposed for incorporating this knowledge into a SVM.

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

Minimize $\quad \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i=1}^{n}\xi_i$

Subject to $\quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i \qquad (\mathbf{x}_i, y_i) \in D, i = 1..n$

$y_j(\mathbf{w}\cdot\mathbf{x}_j + b) \geq 1 \qquad (\mathbf{x}_j, y_j) \in V, j = 1..d$ $\leftarrow$ **New restriction**

$\xi_i \geq 0 \quad \forall i$

- *Insertion method 2*: Two different parameters for error control are defined

Minimize $\quad \frac{1}{2}\|\mathbf{w}\|^2 + C_d\sum_{i=1}^{n}\xi_i^d + C_v\sum_{j=1}^{d}\xi_j^v$

Subject to $\quad y_i(\mathbf{w}\cdot\mathbf{x}_i + b) \geq 1 - \xi_i^d \qquad (\mathbf{x}_i, y_i) \in D, i = 1..n$

$y_j(\mathbf{w}\cdot\mathbf{x}_j + b) \geq 1 - \xi_j^v \qquad (\mathbf{x}_j, y_j) \in V, j = 1..d$ $\leftarrow$ **New restriction**

$\xi_i^d \geq 0 \quad \forall i$

$\xi_j^v \geq 0 \quad \forall i$

The C_d y C_v parameters determine a balance between the information from the 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 other classes. This pre-processing generates a set of virtual supports by class. Next, train a SVM with all the learning sets 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 from the UCI repository, because they have a defined domain theory. To verify whether it is possible to improve the SVM performance when inserting the domain knowledge, the following procedure was carried out: first, a SVM without added knowledge was trained. We then trained 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 (ErrEnt)
- Test set error (ErrTest)
- 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 to the virtual examples. Table 8 shows the results obtained. We can observe that the rule insertion methods improve the original SVM performance.

Table 8. Results obtained by applying the insertion methods on MONK databases.

Strategy	MONK1				MONK2				MONK3			
	Err Ent	Err Test	SV1	SV2	Err Ent	Err Test	SV1	SV2	Err Ent	Err Test	SV1	SV2
Within knowledge	0	0.059	28.24	23.22	0.008	0.146	34.22	29.74	0.023	0.054	14.04	14.44
Method 1	0	0.032	40.08	23.80	0.003	0.088	47.52	30.80	0.021	0.034	14.00	13.36
Method 2	0	0.032	40.08	23.80	0.003	0.088	47.84	30.72	0.020	0.033	13.04	13.70
Method 3	0	0.029	38.44	23.96	0.003	0.086	44.20	30.90	0.021	0.029	13.52	13.28

5. CONCLUSIONS AND FUTURE WORK

With the aim 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 with the rule extraction method on artificial data sets and with databases of different domains, show high levels of equivalence between the SVM and the extracted rule base.

Additionally, a rule extraction method for RBFNN was developed, which uses the algorithm for constructing an ellipsoid proposed for SVM as a core. Based on the results obtained, it can be concluded that the extraction technique derives consistent models with the RBFNN without any previous requirement from either the used training regime or 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 ways of extending the rule extraction methods to regression problems. If this were achieved, a more versatile technique would be available for a larger number of cases. Another question currently emerging is the study of the possibility of using another representation language to express the new model, such as fuzzy rules generated by ellipsoids.

References

[1] Andrews R., Diederich J. and Tickle A. (1995). A survey and Critique of Techniques for Extracting Rules from Trained Artificial Neural Networks. *Knowledge-Based Systems*. 8(6):373-389.

[2] Angulo, C. (2001). Matlab SVM/K-SVCR Toolbox. http://webesaii.upc.es/usr/cecilio/software.htm.

[3] Blake, C.L. and Merz, C.J. (1998). UCI Repository of Machine Learning Data-Bases. University of California, Irvine. http://www.ics.uci.edu/~mlearn/MLRepository.html.

[4] Brotherton, T., Chadderdon, G. and Grabill, P. (1999). Automated Rule Extraction for Engine Vibration Analysis. *Proc. IEEE Aerospace Conference*. 3:29-38.

[5] Cortes C. and Vapnik V. (1995). Support-Vector Networks. *Machine Learning*. 20:273-297.

[6] Craven M. and Shavlik J. (1997). Using Neural Networks for Data Mining. *Future Generation Computer Systems*. 13:211-229.

[7] Cristianini, N. and Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and other kernel-based learning methods*. Cambridge University Press.

[8] Duda, R., Hart, P. and Stork, D. (2001). *Pattern Recognition*. Second Edition. John Wiley & Sons, Inc.

[9] Fu, X. and Wang, L. (2001). Rule Extraction by Genetic Algorithms Based on a simplified RBF Neural Network. *Proc. of the Congress on Evolutionary Computation*. 2:753-758.

[10]Huber, K. and Berthold, M. (1995). Building Precise Classifiers with Automatic Rule Extraction. *Proc. IEEE International Conference on Neural Networks*. 3:1263-1268

[11]Kecman, V. (2001). *Learning and Soft Computing. Support Vector Machines, Neural Networks and Fuzzy Logic Models*. MIT Press.

[12]Ma, J. and Zhao, Y. (2002). OSU Support Vector Machines Toolbox, version 3.0. http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[13]McGarry, K., Wermter, S. and MacIntyre, J. (2001). Knowledge Extraction from Local Function Networks. *Proc. International Joint Conference on Neural Networks*. 765-770

[14]Mitra, S., Pal, S. K. and Mitra, P. (2002). Data Mining in Soft Computing Framework: A survey. *IEEE Transactions on Neural Networks*. 13 (1):3-14.

[15]Moody, J. and Darken, C.J. (1989). Fast Learning in Networks of Locally-Tuned Processing Units. *Neural Computation*. 1:281-294.

[16]Nabney, I. and Bishop, C. Netlab Neural Networks Software. http://www.ncrg.aston.ac.uk/netlab.

[17]Núñez, H. (2003). Hybrid Learning Systems based on Support Vector Machines and Radial Basis Function Neural Networks. Ph.D dissertation. Technical University of Catalonia. Spain.

[18]Orr, M. Radial Basis Function Networks. http://www.anc.ed.ac.uk/~mjo/rbf.html.

[19]Poggio, T. and Girosi, F. (1990). Networks for Approximation and Learning. *Proceedings of the IEEE*. 78:1481-1497.

[20]Schölkopf, B., Burges, C. and Vapnik, V. (1996). Incorporating Invariances in Support Vector Learning Machine. *Lecture Notes in Computer Science*. 1112:47-52.

[21]Setiono, R., Leow, W. and Zurada, J. (2002). Extraction Rules from Artificial Neural Networks for Nonlinear Regression. *IEEE Transactions on Neural Networks*. 13(3):564-577.

[22]Tickle, A., Maire, F., Bologna, G., Andrews, R.; Diederich, J.: Lessons from Past, Current Issues, and Future Research Directions in Extracting the Knowledge Embedded Artificial Neural Networks. In: Wermter, S. and Sun, R. (Eds): *Hybrid Neural Systems*. Springer-Verlag.

[23]Vapnik, V. (1998). *Statistical Learning Theory*. John Wiley & Sons, Inc.

[24]Zien, A., Rätsch, G., Mika, S., Schölkopf, B., Lengauer, T. and Müller, K.-R. (2000). Engineering Support Vector Machine kernels that Recognize Translation Initiation Sites. *Bioinformatics*. 16(9):799-807.

[25]Zhang, X. (1999). Using Class-Center vectors to build Support Vector Machines. *Proc. IEEE Conference on Neural Networks for Signal Processing*. 3-11.

[26]Zhao, Q. and Principe, J.C. (1999). Improving ATR Performance by Incorporating Virtual negative examples. *Proc. International Joint Conference on Neural Networks*. 5:3198-3203.