Rule Extraction Based on Support and Prototype Vectors

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

[1] Artificial Intelligence Laboratory, School of Computing, Central University of Venezuela, Caracas, Venezuela
[2] Knowledge Engineering Research Group, Technical University of Catalonia, Rambla de l'Exposició s/n. E-08800 Vilanova i laGeltrú, Spain

The support vector machine (SVM) is a modelling technique based on the statistical learning theory (Cortes and Vapnik 1995; Cristianini and Shawe-Taylor 2000; Vapnik 1998), which has been successfully applied initially in classification problems and later extended in different domains to other kind of problems like regression or novel detection. As a learning tool, it has demonstrated its strength especially in the cases where a data set of reduced size is at hands and/or when input space is of a high dimensionality. Nevertheless, a possible limitation of the SVMs is, similarly to the neuronal networks case, that they are only able of generating results in the form of black box models; that is, the solution provided by them is difficult to be interpreted from the point of view of the user.

In the neuronal networks research area there has been a wide activity addressed to solve this situation by developing methods able to transfer the knowledge acquired by a neuronal network during the learning phase to a more amenable representation (Andrews et al. 1995; Craven and Shavlik 1997; Tickle et al. 1998; Tickle et al. 2000). The objective of these rule extraction methods is to use the neuronal networks like a tool for solving the problem, obtaining benefit from the advantages that offer these learning paradigms (like its good generalization property, its ability to process nonlinear relations and their high tolerance to the noise and the imprecision in the input data), as well as adding the possibility to open the black box, which would allow to obtain an explanatory result of the problem under study and a simpler solution to be understood by the user. Hence, the extraction of rules improves the adjustment of the neuronal networks to solve problems of data mining (Mitra et al. 2002; Witten and Frank 2005), when the primary target is to discover unknown and implicit relations in large databases that, in many cases, is necessary to be expressed in a comprehensible format.

Following this line of work, a solution to the lack of transparency of the models generated for the support vector machines would be the development of specific techniques of rule extraction directed to this kind of learning machines.

H. Núñez et al.: *Rule Extraction Based on Support and Prototype Vectors*, Studies in Computational Intelligence (SCI) **80**, 109–134 (2008)
www.springerlink.com
© Springer-Verlag Berlin Heidelberg 2008

An interesting property of the SVM is that the hypothesis that it generates is built on the basis of a subgroup of training vectors called support vectors. These vectors constitute the key elements of the learning set since they are the points nearest to the decision limit and, therefore, represents the most informative patterns for building the solution. This explicit dependency of the learned model on the support vectors will facilitate the work of its interpretation.

A rule extraction method for the interpretation of support vector machines is presented in this work which uses the support vectors, along with prototype vectors generated by any clustering algorithm, for building a set of regions that fit the limit of the decision function defined by the SVM, so that it can be transferred to interpretable rules by the user (Núñez et al. 2002a, 2003). These regions can be built in two types: ellipsoids, which will generate equation-type rules, and hyper-rectangles, built from ellipsoids parallel to the axes of the variables, rising to a more comprehensible language in the form of interval rules.

In Sect. 1, the algorithm for the extraction of rules from a SVM is presented in detail, starting with the explanation about how generating an ellipsoid of maximal coverage that adjusts to the decision function generated for the SVM. Next, it is described the generation of a set of rules for a class. This algorithm will be modified to derive interval rules. The description of the mechanism ends with the description of the most interesting features of the algorithm. Several experiments on standard databases are described for different domains in Sect. 2, in order to evaluate the performance of the proposed rule extraction method, in particular its ability to extract the knowledge retained in a trained SVM. Since it is proved that the algorithm is strongly dependent on the initial conditions of the clustering algorithm used to derive the prototype vectors, in Sect. 3 a different approach is described for obtaining these prototype vectors based on the support vectors, overcoming in this form all the randomness due to the nowadays avoided clustering algorithm. Finally, some conclusions are derived and future works are sketched.

1 Combining Support Vectors and Prototype Vectors to Extract Rules

The methods for the extraction of rules from neuronal networks can be classified according to three basic features (Andrews et al. 1995; Craven and Shavlik 1997):

- The representation language used to describe the model estimated by the network.
- The form how the method explores the network to derive the rules. In this sense, local methods analyse the structure of the network at level of both, the hidden units and the output units to extract the rules, whereas

global methods extract rules on the basis of the input–output mapping generated by the network, with not analysis of its internal structure. Hybrid techniques have also been settled out in the between of these two methods.
- The portability of the method; that is, whether the technique is applicable independently of the network architecture and its training regime.

These features could be also used to define a rule extraction method for support vector machines and, similar to the case of neuronal networks, the main problem to be solved is to transfer the knowledge acquired by a SVM during the learning to a description in a new and comprehensible representation language. A key point to be considered is the functional equivalence between the new model and the SVM model from which it was extracted, by providing the same predictions.

For the development of such a translation technique, it is important to identify how the knowledge is codified for the hypothesis generated by a SVM. The solution provided by these learning machines is an expansion of kernel functions on the basis of the number of support vectors,

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

It could be affirmed then that knowledge is expressed in the form of:

- A set of support vectors $SV = \{(\mathbf{x}_i, \mathrm{y}_i)_{i=1\ldots sv}\}$, which are the data from the learning set LS nearest the limit of separation between classes
- A set of values $A = \{\alpha_{i=1\ldots N}\}$ associated to the data, which indicate whether or not a pattern is a support vector without error ($0 < \alpha < C$), a support vector with error ($\alpha = C$), or a pattern not considered in building the decision function ($\alpha = 0$)
- A kernel function K and its associated parameters, such as degree in a polynomial function or width in a Gaussian kernel

The support vectors, although in general are a small group of patterns into the learning set, are the most informative samples for the classification task. Being these vectors the points nearest the limit of separation between classes, would turn out advantageous then to use them in the extraction technique in the form of a set of class delimiters establishing the borders of regions defined in the input space that can be transferred to rules. In the method detailed in this work, these regions are a form of ellipsoids and the new representational language is a rule equation of the type 'if-then' using like antecedent or premise to the mathematical equation of the ellipsoid and like consequent, the label of the class associated to the data covered by this one, as it is shown in Fig. 1.

Transferring to this new representation the knowledge captured by the SVM during the learning entails the determination of a set of ellipsoids that fit to the form of the decision limit, avoiding as much as possible the overlapping between classes in the new model. In this sense, it is beneficial for

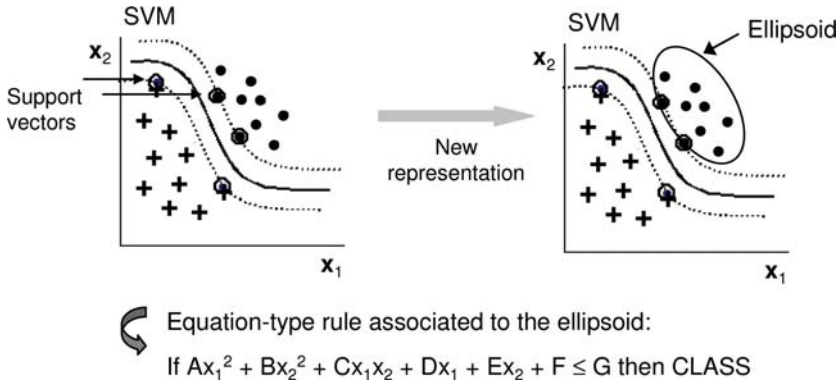

Fig. 1. Representational language used for the rule extraction

building these regions to use, in addition to the support vectors, the information provided by the vector of parameters α, since it indicates whether a support vector is in the region associated to another class or within the margin of separation between classes (those with a value of the associated parameter equal to C). In reference to the kernel functions, it is considered advisable that the method is independent of this characteristic in order that it can be applied to the widest range of SVM models.

Once defined the representational language of the extraction method, the problem to be approached is how to build these ellipsoids for a class from the information provided by a trained SVM. It will be two aspects that will be taken into account to perform this task:

- Ellipsoids fitting. The set of ellipsoids must fit the form of the discriminant function defined by the SVM, exhibiting as low overlapping between classes as possible to avoid problems of multiple instances in the set of rules. This premise will allow building rules with a high precision.
- Ellipsoids coverage. It is advisable to built ellipsoids covering as much data as possible with the purpose of producing a compact set of rules.

It is explained in the next, in a detailed form, the extraction method for building the associated ellipsoids to a class. It will be presented in a two-stage incremental procedure: first it will be assumed that it is possible to represent a whole class with a single ellipsoid. Next, it will be described how to generate a set of rules when this premise is not fulfilled, as it is usually the case.

1.1 Building an Ellipsoid and Its Associated Rule Equation

It is possible to define an ellipsoid associated to an input data set by determining the covariance matrix of the set and finding their eigenvectors and eigenvalues (Strang 1998). In this form, a set of axes for an ellipsoid following the directions of greater variance of the data is obtained, the vertices

being determined from the own values. Nevertheless, the idea underlying in the proposed method is that the ellipsoids should adjust to the form of the limit of decision generated by the SVM. In order to obtain an ellipsoid with these characteristics, its orientation is defined by the support vectors, which are explicitly used for determining the associated set of axes of the ellipsoid and vertices by means of geometric methods. Hence, the problem associated to determining such an ellipsoid in the m-dimensional input space, can be defined as:

Given

- A set of support vectors $\boldsymbol{SV} = \left\{(\boldsymbol{x}_i, \boldsymbol{y}_i) \in \Re^{m+1}, \boldsymbol{i} = 1 \ldots \boldsymbol{sv}\right\}$ obtained by using some SVM training procedure
- A set of remaining training data $\boldsymbol{D} = \left\{\left(\boldsymbol{x}_j, \boldsymbol{y}_j\right) \in \Re^{m+1}, \left(\boldsymbol{x}_j, \boldsymbol{y}_j\right) \notin \boldsymbol{SV}\right\}$, defined by excluding support vectors from the original training data
- A set of parameters $\mathrm{A} = \{(\alpha_{\boldsymbol{k}}) \in \Re, k = 1 \ldots N\}$, obtained by the SVM training procedure, being different to zero those associated to the support vectors

Determine

- A centre $\boldsymbol{p}$
- A set of orthonormal vectors, $\boldsymbol{E} = \{(\boldsymbol{e}_i), i = 1 \ldots m\}$
- A set of pair of vertices, $\boldsymbol{V} = \{(\boldsymbol{v}_{i1}, \boldsymbol{v}_{i2}), i = 1 \ldots m\}$

To define an ellipsoid which orientation is explicitly determined by the support vectors.

The algorithm in pseudo-code to derive an ellipsoid is in Table 1. First step in the algorithm is the initialization of the output values. Output sets

Table 1. Algorithm for deriving an ellipsoid

```
{Input: SV, D, A}
Initialize p, E, V
Build_Ellipsoid
    {e1, v11} = Determine_First_Axis_Vertex
    v12 = Determine_Second_Vertex
    E = E ∪ e1, V = V ∪ {(v11, v12)}
    p = Update_Prototype
    For k = 2 to m
        {ei, vi1} = Determine_Next_Axis_Vertex
        vi2 = Determine_Second_Vertex
        E = E ∪ ei, V = V ∪ {(vi1, vi2)}
    End_For
End_Build_Ellipsoid
{Output: p, E, V}
```

are set to null and the first considered centre of the ellipsoid is defined like the centre of gravity of all the data in the class,

$$p = \frac{1}{N} \sum_{i=1}^{N} x_i, \boldsymbol{x}_i \in LS. \tag{2}$$

Once the initial prototype is calculated, now the algorithm to build the ellipsoid determines, using the Determine_First_Axis_Vertex procedure, the first axis of the ellipsoid as well as the first vertex in this axis. Using Determine_Second_Vertex, the second vertex along the first axis is defined. E is a matrix of column vectors containing the orthonormal vectors defining the axes of the ellipsoid. V holds for the vertices associated to each axis. Next, the algorithm enters into a loop in which, both, the axis and its first vertex are determined each iteration by means of the Determine_Next_Axis_Vertex procedure, as well as the second vertex through the Determine_Second_Vertex procedure. It will be now detailed each one of the procedures used in the algorithm.

Determine_First_Axis_Vertex Procedure

This procedure determines both, the vector that will be used like the first axis and the first vertex considered on this axis. The vector $e_1 \in E$ is built from the prototype $\boldsymbol{p}$ and the support vector without error, i.e. $\alpha < C$ having maximal distance to the prototype,

$$q_{11} = \{x | \, \|x - p\| = \operatorname{argmax} (\|x_j - p\|), x_j \in SV, \alpha_j < C\}. \tag{3}$$

As it were already indicated, an ellipsoid as general as possible should be built, but fitting the form of the decision limit defined by the SVM.

Whether more than a support vector fulfils this condition, one of them is randomly selected. In the opposite, it is also possible that a support vector without error does not exist. In this case, the pattern in the set D with maximal distance to the prototype is selected,

$$q_{11} = \{x | \, \|x - p\| = \operatorname{argmax} (\|x_j - p\|), x_j \in D\}. \tag{4}$$

The unitary vector is defined as,

$$e_1 = \frac{q_{11} - p}{\|q_{11} - p\|}. \tag{5}$$

And the first vertex along this axis is the selected end-point,

$$v_{11} = q_{11}. \tag{6}$$

Determine_Second_Vertex Procedure

Through this procedure the second vertex on the axis is determined in the following form: first, a search region for support vectors is built by considering the subset of support vectors accomplishing,

$$SV_{i2} = \left\{ x \in SV | \arccos \left(\frac{(x-p) \cdot e_i}{\|x-p\|} \right) \geq \frac{3\pi}{8} \right\}. \tag{7}$$

Support vectors in this zone are those such that its projection with respect to the prototype over the axis is greater that any other projection over orthogonal axes to this one. The support vector related with the second vertex is that in SV_{i2} without error and larger distance to the prototype $\boldsymbol{p}$,

$$q_{12} = \{x | \|x-p\| = \operatorname{argmax} (\|x_j - p\|), x_j \in SV_{i2}, \alpha_j < C\}. \tag{8}$$

Again, when more than a support vector exists fulfilling the specific condition, one of them is randomly selected. For the case that no support vector exists in the zone, the pattern in D that satisfies the established criterion is selected.

Since the more general ellipsoid is searched, the second vertex is determined as,

$$v_{i2} = p - \|q_{i2} - p\| e_i. \tag{9}$$

When it does not exist neither data in the SV_{i2} set nor in the set D, the vertex is defined by using the first vertex as,

$$v_{i2} = p - \|v_{i1} - p\| e_i. \tag{10}$$

Update_Prototype Procedure

Once vertices are both determined on the first axis, the prototype is updated to be its midpoint. In this form, the original averaged initial centre $\boldsymbol{p}$ is replaced through the geometric situation of the support vectors by,

$$p = \frac{v_{11} + v_{12}}{2}. \tag{11}$$

Determine_Next_Axis_Vertex Procedure

This procedure iteratively determines orthonormal vectors to the set E included in the $m - i + 1$ linear manifold containing the centre $\boldsymbol{p}$. They are selected guided by the support vectors. Hence, first the set of support vectors

$$SV_{i1} = \left\{ x \in SV | ang_x \leq \frac{\pi}{4} \right\}, \tag{12}$$

is firstly determined, where ang_x is the angle determined by the vector $(x-p)$ and the $m-i+1$ linear subspace,

$$ang_x = \arccos\left(\sqrt{1-\sum_{j=1}^{i-1}\left(\frac{(x-p)\cdot e_j}{\|x-p\|}\right)^2}\right). \tag{13}$$

From this set, the vector with greater distance to the prototype is selected,

$$q_{i1} = \{x|\,\|x-p\| = \operatorname{argmax}\left(\|x_j-p\|\right), x_j \in SV_{i1}, \alpha_j < C\}. \tag{14}$$

When no support vector exists in the searching zone, then the pattern in D that fulfils the maximal distance criterion is selected. In order to determine the unitary vector, a projection of the former vector with respect to the prototype on the $m-i+1$ linear subspace is used. It is built by using the projection matrix $\mathbf{M}_p$ onto the subspace generated by E,

$$\mathbf{M}_p = EE^T. \tag{15}$$

Hence, the projection is defined as,

$$pq_{i1} = (q_{i1}-p) - \mathbf{M}_p\,(q_{i1}-p) \tag{16}$$

the orthonormal vector is built as,

$$e_i = \frac{(pq_{i1}-p)}{\|pq_{i1}-p\|} \tag{17}$$

and the first vertex in this vector will be defined as,

$$v_{i1} = p + \|q_{i1}-p\|\,e_i. \tag{18}$$

A key point to be solved is related with the infeasibility to find in the searching region a support vector or a pattern for determining an orthogonal vector to the set E. When it is not possible to find such a pattern in a certain iteration k, then it will be also impossible to find a pattern accomplishing the criteria in the successive iterations, that is, when the linear subspace will be smaller. In this situation, it has not more sense to evaluate our proposed algorithm of finding for the axe defined from support vectors. The ellipsoid is projected in this case on the k-dimensional linear manifold and a spheroid is obtained in this subspace.

In order to obtain a set of k orthonormal vectors in the k-dimensional subspace, an equation system can be solved and one of the infinite solutions be selected. Nevertheless, it is proposed to use a procedure based on the Gram-Schmidt orthogonalization (Strang 1998) which is described in the following.

Given a standard set of unitary vectors,

$$U = \{(u_i)\}_{i=1,\ldots,m} \tag{19}$$

an auxiliary set B_e, initialized as U is defined. Each time that a vector in E is determined, the nearest vector in B_e is searched,

$$proy_{\max} = \arg\max_{u \in B_e} (u \cdot e_i) \tag{20}$$

$$u_{e_i} = \{u_j \in B_e | u_j \cdot e_i = proy_{max}\} \tag{21}$$

and it is removed from the set B_e.

In the case that it is impossible to find a point (a support vector or a general pattern) that fulfils the criterion determining an axis, the k orthonormal vectors to the set E are determined from the set B_e in the following form,

For l $= m - k + 1$ to m
$\mathbf{M}_p = EE^T$
$j = 1$
$e_l = u_j - \mathbf{M}_p u_j; e_l = \frac{e_l}{\|e_l\|}; E = E + \{e_l\}$
$j = j + 1$
End_For

This procedure allows obtaining a fast unique solution without solving a system of equations, using basic vector operations and matrix product. In order to determine the vertices on each axis it is necessary to find the radius of the spheroid, which can be obtained from the radii associated to the axes in E. Several choices are possible, to use the greater radius, to use the smaller radius or the average of the radii. Since the most general ellipsoid is desired, the greater radius criterion will be used. When overlapping with data of other classes due to this generalist criterion appears, the ellipsoid can be specialized later by using a procedure for determining a set of rules that will be described later.

Another point to be solved is about the fact that, with the exception of the first axis, it will be usual that the vertices on the axes define a different radius, therefore one of them must be refined. Two criteria to decide which vertex to refine can be: using the radius defined by the vertex derived from a support vector or considering always the greater one from radii. Again, if overlapping with data of other classes appears when applying this heuristic, then the ellipsoid will specialize.

Generating the Rule

Once generated the ellipsoid, the rule equation can be derived from the centre $\boldsymbol{p}$, the set of orthonormal vectors E and the set of vertices V in the following form:

Let's suppose a 3-dimensional input space, then the ellipsoid is defined as,

$$\left(\frac{x_1'}{r_1}\right)^2 + \left(\frac{x_2'}{r_2}\right)^2 + \left(\frac{x_3'}{r_3}\right)^2 \leq 1 \tag{22}$$

with $r_i = \|v_{i1} - p\|$, $p = (p_1, p_2, p_3)$ and

$$\begin{bmatrix} x_1' \\ x_2' \\ x_3' \end{bmatrix} = E^T \cdot \begin{bmatrix} x_1 - p_1 \\ x_2 - p_2 \\ x_3 - p_3 \end{bmatrix}. \tag{23}$$

Developing (23), it is obtained an expression in the form

$$Ax_1^2 + Bx_2^2 + Cx_3^2 + Dx_1x_2 + Ex_1x_3 + Fx_2x_3 + Gx_1 + Hx_2 + Ix_3 + J \leq K. \tag{24}$$

Generating an equation-type rule with a form,

$IF\ Ax_1^2 + Bx_2^2 + Cx_3^2 + Dx_1x_2 + Ex_1x_3 + Fx_2x_3 + Gx_1 + Hx_2 + Ix_3 + J \leq K\ THEN$ CLASS.

1.2 Generating a Set of Rules

Is an ellipsoid enough to describe the distinctive zone of a class? A positive answer is possible for some cases, but it will not be thus for the general case. On the other hand, it is difficult to establish a priori the number of necessary rules to represent the model of a SVM.

Two basic premises exist on which the building of the new model is based: the generalization or covering of the rules and the accuracy or precision of the ellipsoids to fit the shape of the decision surface defined by the SVM. The proposed procedure to build an ellipsoid tries to satisfy them. However, when one ellipsoid is not enough to describe the data in a class, the question is how to determine a number of regions that exhibit these characteristics. For instance, it can be observed in Fig. 2a that the ellipsoid generated from the midpoint of the data invades the zone associated to the other class due to the curvature of the decision limit. By dividing the ellipsoid (as it is shown in Fig. 2b) overlapping is reduced and the two new regions fit better to the decision surface than the original region.

Therefore, to generate a set of rules, the extraction method will initially build one ellipsoid which will be divided (specialized) until a group of more specific ellipsoids covering the data in the class and fitting the shape of the separation surface defined by the SVM is obtained.

A mechanism for determining the centre of each ellipsoid must be available in order to be able executing our proposed procedure. Initially, the centre was built as the midpoint of the data in the class. Nevertheless, it must be defined how to find new centres and which data to use to build each ellipsoid when it is mandatory to divide the initial ellipsoid in two or more regions. The derivation of the centres or prototypes can be performed using a clustering algorithm (Duda et al. 2001; Kaufman and Rousseeuw 1990), which divides the data set of a class in a predetermined number of disjoint partitions and it determines a prototype or representative centre for each one of the partitions.

On the other hand, it is also necessary to establish conditions for determining when to divide an ellipsoid. Division criteria can be based on an index

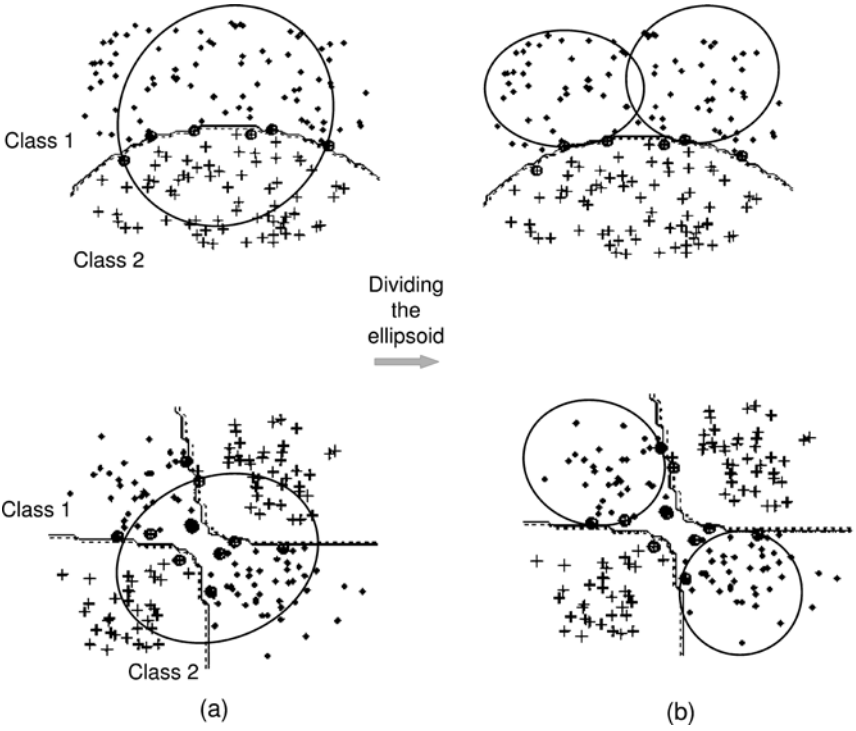

Fig. 2. (**a**) Generating one ellipsoid for a class. (**b**) By dividing the ellipsoid, two regions are obtained fitting better the decision function

of overlapping between classes, which could be implemented by means of geometric methods. It can be noticed in Fig. 2 that some support vectors exist belonging to another class within the defined ellipsoid. Since support vectors are the data nearest the decision function, they are the most informative about the shape of this separation surface. A first proposed criterion suggesting when to divide an ellipsoid is:

– *Criterion 1*: To divide an ellipsoid when support vectors of other classes exist in the region covered by it.

In this form, support vectors are used, not only for defining the ellipsoids, but also to verify the overlapping between ellipsoids of different classes. Overlapping also appears when the generated prototypes belong to another class (see Fig. 3); this fact can be verified by using the SVM function to generate the class label for these artificial points, which leads to a second partition criterion:

– *Criterion 2*: To divide an ellipsoid when the generated prototype belongs to another class.

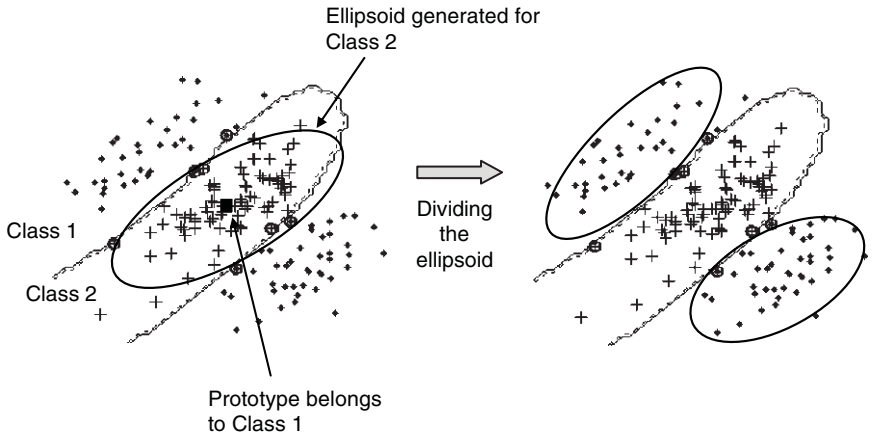

Fig. 3. The prototype generated like the midpoint of the data belongs to another class

Finally, overlapping between classes can also appear when at least one of the vertices generated by the algorithm to build an ellipsoid belongs to another class, which also can be verified using the trained SVM function. This third criterion of division is translated like:

– *Criterion 3*: To divide an ellipsoid when at least one of the vertices belongs to another class.

These three criteria can be formalized like a test of partition. Whether this test is positive when being applied to an ellipsoid, it indicates that it is very probable that it covers data from other classes.

It has been showed that overlapping can be reduced and fitting to the decision function increased by increasing the number of regions describing a class, however it should be now addressed how many regions are needed to describe data in a class. This number of regions could be user defined, providing in this form control on the size of the set of rules. Also it could be considered to divide ellipsoids until some established criterion is reached, for instance a good level of prediction.

According to this analysis, the extraction method for generating the set of ellipsoids associated to a class follows an iterative scheme. Starting by the prototype of the class (midpoint of the data) the initial ellipsoid is derived. Next, the partition test is applied on this region; when the answer to the test is negative the ellipsoid is transferred to a rule. Otherwise, a clustering algorithm is applied for determining two new prototypes with the data of the initial partition (data of the class); one ellipsoid is built for each one of them using the data of the respective partitions. The partition test is applied to these two new regions and the process is repeated. In this form, in the k-th iteration, tp regions have a positive test of partition and tn will be with negative answer. These last ones are transferred to rules. In the next iteration

$k + 1$, data from the tp regions are used to generate $tp + 1$ new prototype vectors which will lead to $tp + 1$ new regions. The procedure ends when all the partition tests have a negative answer or when the maximum number of iterations is reached (externally defined to control the number of generated rules). In Table 2 the complete algorithm to be used for deriving a set of rules is described, where:

– Determine_Prototypes (Data, Number_regions): It is a function for determining an equal number of prototypes to Number_regions for the data, using a clustering algorithm. For each prototype it also returns the

Table 2. Algorithm for deriving a set of rules

```
{Input: SV, D, SVM function}
Initialize_Generating_Rules
Do for each Class
    Number_regions = 1
    Data = Data_class
    [prototypes, partition] = Determine_Prototypes (Data, Number_regions)
    Ellipsoid = Build_Ellipsoid (prototypes, partition)
    Ellipsoid_rules = Ellipsoid
    Condition(1) = Partition_Test (Ellipsoid)
    Number_regions = 2
    While ((Condition(i) = 1) ∧ Iterations < max_iterations)
        [prototypes, partition] = Determine_Prototypes (Data,
        number_regions)
        For i = 1 to number_regions
          Ellipsoid(i) = Build_Ellipsoid (prototypes(i), partition(i))
          Condition(i) = Partition_Test (Ellipsoid (i))
        End_For
        Number_new_regions = 1
        New_data = [ ]
        For i = 1 to Number_regions
          If Condition(i) = 0 ∨ Iterations = max_iterations
            ellipsoid_rules = ellipsoid(i)
          Else
            Number_new_regions = Number_new_regions + 1
            New_data = New_data + Partition(i)
          End_If
        End_For
        Data = New_data
        Number_regions = Number_new_regions
    End_While
End_Do
{Output: rules}
```

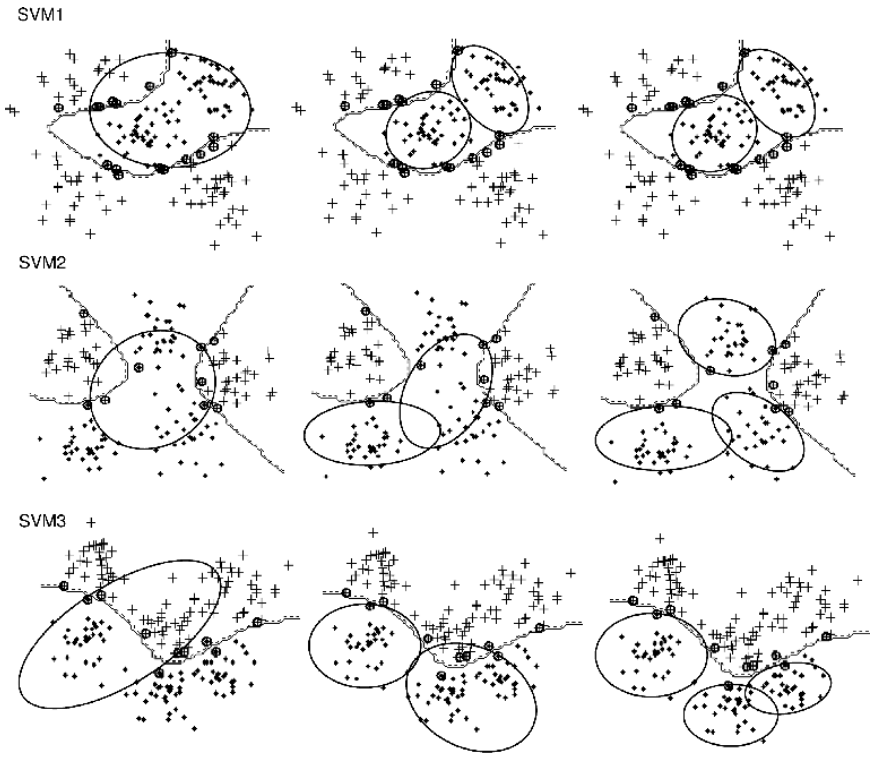

Fig. 4. Several examples of ellipsoids generated for learned SVM. Iteration *left* to *right*

respective partition. In the first iteration, the prototype will be the midpoint of the data.
- Build_Ellipsoid (prototypes, partition): This function builds an ellipsoid from a prototype and data from a partition.
- Partition_Test (Ellipsoid): It returns the logical answer of the partition test on an ellipsoid.

Figure 4 shows several examples of the iterative application of the rule extraction algorithm.

1.3 Simplified Representational Language for the Model

The ellipsoids and their equation-type rules define a representational language obtained by the rule extraction method to describe the model generated by the trained SVM. However, it will be showed that it is possible to derive more interpretable type rules by using like premise a set of constraints over the values of each one of the variables, to be satisfied so that the consequent one

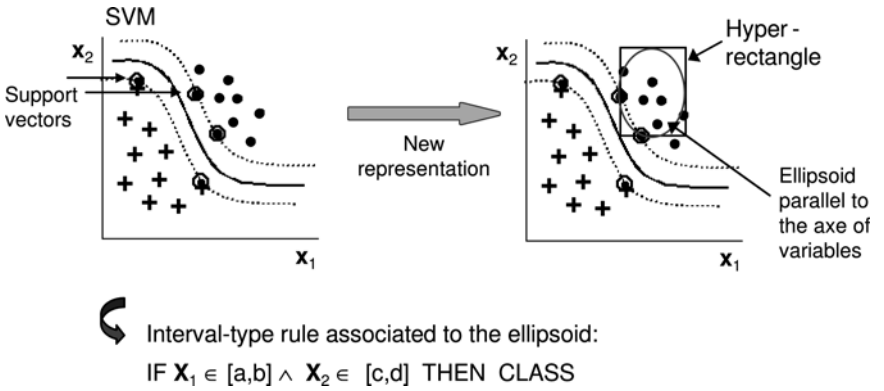

Fig. 5. Interval-type rule for SVM

is fulfilled (class label). This second representational language, called interval-type rules, can be observed in Fig. 5: it is associated to a convex region in the form of a hyper-rectangle generated from an ellipsoid parallel to the axes of the variables.

In order to derive the interval-type rules, the procedures Determine_First_Axis_Vertex and Determine_Next_Axis_Vertex of the Algorithm shown in Table 1 are modified. Key difference is about how the axes of the ellipsoid rising to a hyper-rectangle are built, by using the standard set U of unitary vectors. Support vectors are used to establish the order in selecting unitary vectors of the set B_e according to a criterion of proximity with the end points, as well as to build the vertices on the axes. This heuristic will allow that the ellipsoids parallel to the axes of the variables fit the shape of the separation surface defined by the SVM, without overlapping with the distinctive zone of other classes. These two modified procedures are described in the following.

Determine_First_Axis_Vertex Procedure

Given the standard set of unitary vectors U, it is initially defined $B_e = \mathrm{U}$, and $\boldsymbol{q}_{11}$ is determined as defined in Sect. 1.1. Next, it is calculated the projection of the vector $(\boldsymbol{q}_{11} - \boldsymbol{p})$ onto any vector in B_e. First selected unitary vector will be that in this set with higher projection, selecting in this form the axis with higher information about the support vector, that is,

$$proy_{max} = \arg\max_{u \in B_e} \left((q_{11} - p) \cdot u \right) \tag{25}$$

$$e_1 = \left\{ u_i \in B_e | \, (\boldsymbol{q_{11}} - p) \cdot u_i = proy_{max} \right\}. \tag{26}$$

The selected vector is removed from the set B_e. The vertex $\boldsymbol{v}_{11}$ is the projection of the vector $(q_{11} - p)$ on the next axis $\boldsymbol{e}_1$,

$$v_{11} = p + (e_1 \cdot (q_{11} - p))\, e_1. \tag{27}$$

Determine_Next_Axis_Vertex Procedure

In order to determine the next axe, the support vectors to be considered are those forming with the centre an angle lower than 45° with respect to the $m-i+1$ linear manifold containing the centre and that is orthogonal to the set of vectors E, which leads to a subset defined as,

$$SV_{i1} = \left\{ x \in SV | ang_x \leq \frac{\pi}{4} \right\}, \tag{28}$$

where

$$ang_x = \arccos\left(\sqrt{1 - \sum_{j=1}^{i-1} \left(\frac{(x-p)\cdot e_j}{\|x-p\|} \right)^2} \right). \tag{29}$$

From this subset, the support vector without error with maximal distance to the prototype is selected,

$$q_{\mathrm{i1}} = \{ x | \, \|x-p\| = \operatorname{argmax}\left(\|x_j - p\| \right), x_j \in SV_{i1}, \alpha_j < C \}. \tag{30}$$

When there are no support vectors in the searching zone, the pattern in D satisfying the already established criterion for these vectors is selected. In order to determine the unitary vector, the usual projection onto each vector of B_e is considered, and the higher projected one is selected,

$$proy_{max} = \arg\max_{u \in B_e} \left((q_{\mathrm{i1}} - p) \cdot u \right) \tag{31}$$

$$e_{\mathrm{i}} = \{ u_k \in B_e | \, (q_{i\mathbf{1}} - p) \cdot u_k = proy_{max} \}. \tag{32}$$

The selected vector is removed from B_e. The vertex is calculated as usual,

$$v_{i1} = p + \left(e_i \cdot (q_{i1} - p) \right) e_i. \tag{33}$$

When it is impossible to find a point (a support vector or a general pattern) that fulfils the criterion determining an axis, the k unitary vectors to the set E are determined from the set B_e in the following form,

$$\boldsymbol{r}_{\boldsymbol{esferoide}} = \underset{(i=1\ldots(m-k))(j=1,2)}{argmax} \left(\|\boldsymbol{v_{ij}} - \boldsymbol{p}\| \right).$$

For $l = (m-k+1)$ *to* m
  $\boldsymbol{j} = \mathbf{1}$
  $\boldsymbol{e_l} = \boldsymbol{u_j}$
  $\boldsymbol{v_{l1}} = \boldsymbol{p} + \boldsymbol{r}_{\boldsymbol{esferoide}}\, \boldsymbol{e_l}$
  $\boldsymbol{v_{l2}} = \boldsymbol{p} - \boldsymbol{r}_{\boldsymbol{esferoide}}\, \boldsymbol{e_l}$
  $\boldsymbol{E} = \boldsymbol{E} + \{\boldsymbol{e_l}\}$
  $\boldsymbol{j} = \boldsymbol{j} + \mathbf{1}$
End_For

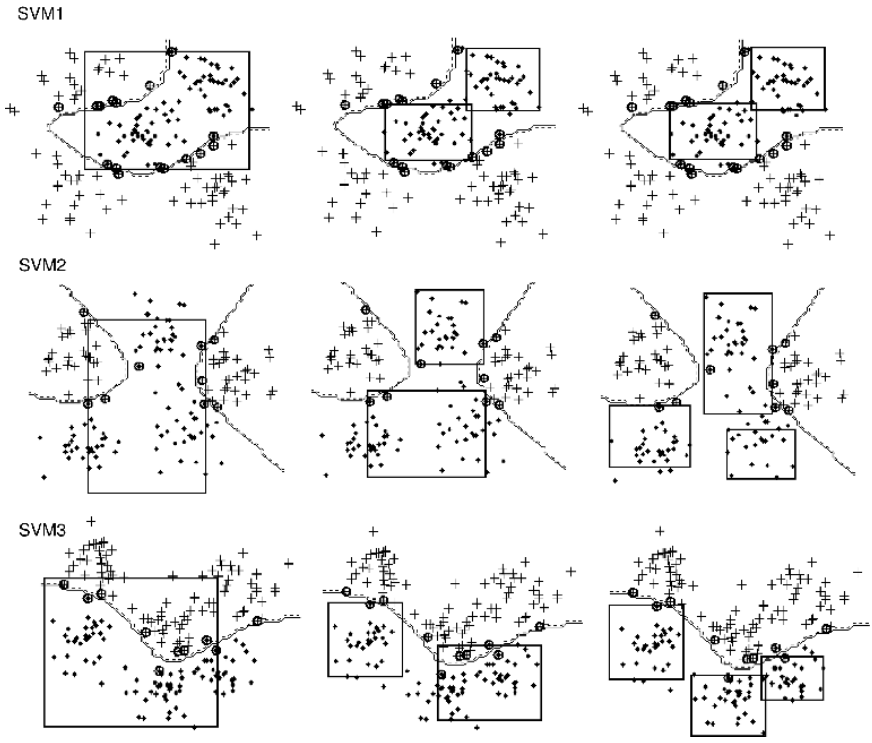

Fig. 6. Several examples of hyper-rectangles generated for learned SVM. Iteration *left* to *right*

The greater radius criterion is used for obtaining an as general ellipsoid as possible. Figure 6 shows some examples for hyper-rectangles associated to ellipsoids generated using the described iterative procedure.

Once derived the ellipsoid, its translation to an interval-type rule is performed by using a set of ordered vertexes $\boldsymbol{V} = \{(\boldsymbol{v}_{i1}, \boldsymbol{v}_{i2}), i = 1 \ldots m\}$ as follows,

$$IF\ x_1 \in [v_{11}, v_{12}] \wedge x_2 \in [v_{21}, v_{22}] \wedge \cdots \wedge x_m \in [v_{m1}, v_{m2}]\ \ THEN\ Class.$$

Determination of the set of interval-type rules for a class is processed similarly to that procedure described in Table 2; nevertheless, some modifications should be considered:

- The function Build_Ellipsoid is now based on the algorithm for building an ellipsoid parallel to the axis of the variables.
- Verification of the Criterion 1 in the function Partition_Test is performed on the hyper-rectangle associated to the ellipsoid in order to reduce the overlapping when deriving the interval-type rule.

1.4 Classification by Using the Set of Rules

Once obtained the model of the learned SVM in the new representation (equation-type or interval-type rules) it must be considered how this description will be used to classify a new pattern. Several scenarios can appear for the new pattern being evaluated:

- *It is covered only for one rule.* This is the most favourable case, allowing classifying a new entry without ambiguity.
- *It is covered for no rule.* In this case it is possible to define a default rule for classifying all these cases not covered for any rule (Mitchell 1997; Witten and Frank 2005). Using a similarity measure for determining the proximity of a pattern to a rule is an alternative choice; the assigned class label to the data will be those associated to the nearest rule (Domingos 1991).
- *It is covered for more than a rule.* Several solutions can be considered in this overlapping situation. A first one is ordering the covering rules using some quality measure and classifying the new instance according to the first fired rule (Berthold and Hand 1999; Witten and Frank 2005). A second one is applying a weighted classification scheme using all the covering rules similarly to the fuzzy logic algorithms. Other solutions include using a frequency based scheme that assigns the class associated to the most active rule, or, inversely, assign the pattern to the most specific rule (Salzberg 1991). Finally, it can be attempted to avoid multiple covering by refining the rules until a disjoint partition is achieved.

The proposed method classify an instance assigning it the label associated to the nearest rule (Domingos 1991), following the nearest neighbour technique. A distance measure between a pattern and a rule is defined depending on the type of rule, equation or interval. The distance between an equation rule and an instance is defined as,

$$D\left(R, x\right) = EQ\left(x\right), \tag{34}$$

where $EQ(\boldsymbol{x})$ is the result of evaluating the mathematical equation of the ellipsoid on the pattern $\boldsymbol{x}$. For the interval-type rules, the distance definition is based on a distance component for each attribute defined as follows,

$$\delta_i = \begin{cases} 0 & if \quad l_{i,\mathrm{inf}} \leq x_i \leq l_{i,\mathrm{sup}}, \\ x_i - l_{i,\mathrm{sup}} & if \quad x_i > l_{i,\mathrm{sup}}, \\ l_{i,\mathrm{inf}} - x_i & if \quad x_i < l_{i,\mathrm{inf}}, \end{cases} \tag{35}$$

where $l_{i,inf}$ y $l_{i,sup}$ are the lower and upper bounds of the interval, respectively, of the i-th component. Hence,

$$D\left(R, x\right) = \sum_{i=1}^{m} \delta_i. \tag{36}$$

When an instance is covered for more than a rule, the following heuristic is used to solve the overlapping: the most specific ellipsoid or hyper-rectangle containing the instance is selected, that is, that with the lowest volume (Salzberg 1991). For interval-type rules this volume is calculated as,

$$V(R) = \prod_{i=1}^{m} (l_{i,\text{sup}} - l_{i,\text{inf}}). \tag{37}$$

For equation-type rules, the volume associated to rules is compared by using an approximated measure based on the radius for each axis of the ellipsoid, in the following form,

$$V(R) = \prod_{i=1}^{m} r_i. \tag{38}$$

2 Experiments

The proposed rule extraction methods have been evaluated through experimentation on ten databases from the UCI repository (Blake and Merz 1998), considered a standard benchmark for the machine learning community. Features defining these bases are showed in Table 3: number of input variables, type of variables, number of patterns and number of classes.

The algorithms associated to the rule extraction method were developed under the Matlab v6.5 programming environment. SVM's training was completed using the software package "OSU Support Vector Machines Toolbox" version 3.00 (Ma and Zhao 2002). For multi-class classification, the one-versus-rest technique was employed for determining the SVM decision function (Vapnik 1998), i.e. a classifier was trained for each class and obtained support vectors were stored to be used next in the rule extraction algorithm. For

Table 3. Features describing the ten databases used for experimentation

Code	Databases	No. patterns	No. attributes	Type attributes	No. classes
1	IRIS	150	4	Numerical (continuous)	3
2	WISCONSIN	699	9	Categorical	2
3	WINE	178	13	Numerical (continuous)	3
4	SOYBEAN	47	35	Numerical (discrete)	4
5	NewTHYROID	215	5	Numerical (continuous)	3
6	MUSHROOM	8,124	22	Categorical	2
7	SPECT	267	23	Binary	2
8	MONK3	432	6	Categorical	2
9	ZOO	101	16	Categorical	7
10	HEART	270	13	Mixed	2

all the experiments, the k-means algorithm (Duda et al. 2001) was used for determining the centres or prototypes of the ellipsoids.

A key point is to determine the performance indexes to be used for evaluating the rule extraction algorithm (Andrews et al. 1995; Mitra et al. 2002; Zhou 2004). Goal is the extraction of the embedded knowledge in the trained SVM and represent it in the language defined by the method, so an interesting parameter is to determine the functional equivalence between both methods, known like *fidelity* parameter, being calculated like the percentage of data where both, the SVM and the rule set produce the same results,

$$Fidelity = 100 \cdot \frac{N_{agreed}}{Data_{total}}, \tag{39}$$

where N_{agreed} is the number of times that both, SVM and rule set predicts the same result.

A second main feature to be considered is the generality or covering of the rules on the data set, as well as their accuracy, defined through the error. Hence, two more parameters will be measured to determine the performance of the algorithm:

- Covering: Percentage of samples covered by the set of rules

$$Covering = 100 \cdot \frac{Data_{covered}}{Data_{total}}, \tag{40}$$

  where $Data_{covered}$ is the number of samples covered by the rules and $Data_{total}$ is the size of the dada set
- Error: Mean quadratic error of the rule set on the data

In order to estimate these performance features, ten stratified cross-validation experiments on ten partitions were performed (Witten and Frank 2005) and the mean values on the test set were taken for the performance comparison. Obtained results for each database are shown in Table 4, for the equation-type and interval-type rules. It have been displayed the accuracy (Error), the percentage for the features measuring the equivalence or fidelity (Equ.), the covering (Cov.) and the number of obtained rules (NR).

Results on the application of the rule extraction method to support vector machines trained with real data databases in different domains, shown in general a high percentage of equivalence between the SVM and the extracted set of rules (upper to 90%). This high level indicates that the proposed methods are able to capture the embedded knowledge in the support vector machine.

It was also observed a high dependency of both, the quality and the quantity of the rules generated by the extraction method on the initial conditions for the k-means clustering algorithm. It is well-known that the final result of a clustering algorithm highly depends on the random choice of the prototypes (Duda et al. 2001), so it was a predictable result, but it is not a desirable behaviour. In this sense, a new proposal for reducing this randomness will be explained below, and it is still an open research area.

Table 4. Performance values of the rules for each database

Database	Error	Equation-type rules				Interval-type rules			
	SVM	Error	*Equ.*	Cov.	NR	Error	Equ.	Cov.	NR
1	0.046	0.041	98.67	82.33	6.1	0.038	97.59	80.66	3.8
2	0.045	0.039	98.65	86.45	15.3	0.041	96.54	94.45	14.5
3	0.023	0.018	98.40	78.34	5.9	0.023	97.87	80.96	8.9
4	0.022	0.022	100.00	33.00	6.0	0.028	97.70	84.50	6.0
5	0.052	0.049	97.13	80.24	7.3	0.047	95.33	72.99	10.8
6	0.002	0.003	96.05	28.87	25.5	0.010	99.06	98.19	30.8
7	0.102	0.117	96.26	21.49	14.0	0.093	97.33	45.00	28.0
8	0.023	0.034	97.45	27.55	7.0	0.023	99.07	100.00	10.0
9	0.042	0.043	99.09	32.02	9.8	0.043	98.77	79.01	8.6
10	0.164	0.158	96.93	58.35	6.7	0.155	96.39	66.52	18.7

They are listed below, as a particular example, the set rules generated for the IRIS database using both rule extraction regimes

Equation-type rules

R1: **IF** $(6.16X_1{}^2 + 2.68X_2{}^2 + 9.84X_3{}^2 + 15.96X_4{}^2 - 3.63X_1X_2 - 3.62X_1X_3 - 1.63X_1X_4 - 0.47X_2X_3 + 2.31X_2X_4 - 0.14X_3X_4 + 43.6X_1 + 0.01X_2 - 8.78X_3 - 7.52X_4 + 116.49 \leq 2.64)$ **THEN** Iris-setosa

R2: **IF** $(1.29X_1{}^2 + 4.79X_2{}^2 + 3.31X_3{}^2 + 5.28X_4{}^2 + 1.69X_1X_2 - 2.02X_1X_3 + 0.97X_1X_4 - 2.50X_2X_3 - 1.84X_2X_4 - 2.09X_3X_4 - 11.77X_1 - 22.53X_2 - 6.34X_3 - 5.08X_4 + 78.30 \leq 0.84)$ **THEN** Iris-virsicolor

R3: **IF** $(4.65X_1{}^2 + 3.60X_2{}^2 + 6.75X_3{}^2 + 5.74X_4{}^2 + 0.67X_1X_2 - 1.46X_1X_3 + 1.14X_1X_4 - 0.04X_2X_3 - 0.48X_2X_4 - 2.89X_3X_4 - 56.84X_1 - 25.00X_2 - 47.34X_3 - 9.05X_4 + 333.49 \leq 1.81)$ **THEN** Iris-virsicolor

R4: **IF** $(9.91X_1{}^2 + 5.63X_2{}^2 + 12.13X_3{}^2 + 9.26X_4{}^2 - 3.21X_1X_2 - 3.89X_1X_3 + 7.69X_1X_4 + 4.90X_2X_3 + 0.03X_2X_4 + 0.57X_3X_4 - 127.63X_1 - 41.29X_2$ $137.18X_3 - 96.32X_4 + 1,052.49 \leq 3.54)$ **THEN** Iris-virginica

R5: **IF** $(16.25X_1{}^2 + 49.58X_2{}^2 + 19.01X_3{}^2 + 66.59X_4{}^2 - 11.35X_1X_2 - 8.13X_1X_3 - 5.03X_1X_4 + 4.98X_2X_3 - 54.77X_2X_4 - 10.81X_3X_4 - 109.50X_1 - 128.41X_2 - 140.12X_3 - 22.13X_4 + 886.57 \leq 18.80)$ **THEN** Iris-virginica

Interval-type rules

R1: **IF** $(X_1 \in [4.40, 5.80] \wedge X_2 \in [2.30, 4.40] \wedge X_3 \in [1.00, 1.95] \wedge X_4 \in [0.20, 0.51])$ **THEN** Iris-setosa

R2: **IF** $(X_1 \in [5.40, 6.10] \wedge X_2 \in [2.70, 3.00] \wedge X_3 \in [3.90, 4.57] \wedge X_4 \in [0.97, 1.64])$ **THEN** Iris-versicolor

R3: **IF** $(X_1 \in [4.90, 6.00] \wedge X_2 \in [1.90, 2.93] \wedge X_3 \in [2.99, 4.00] \wedge X_4 \in [1.00, 1.27])$ THEN Iris-versicolor

R4: **IF** ($X_1 \in [6.10, 7.00] \wedge X_2 \in [2.30, 3.30] \wedge X_3 \in [4.28, 4.95] \wedge X_4 \in [1.31, 1.50]$) **THEN** Iris-versicolor
R5: **IF** ($X_1 \in [4.90, 6.70] \wedge X_2 \in [2.02, 3.67] \wedge X_3 \in [4.90, 5.57] \wedge X_4 \in [1.29, 2.63]$) **THEN** Iris-virginica
R6: **IF** ($X_1 \in [6.30, 7.70] \wedge X_2 \in [2.50, 4.00] \wedge X_3 \in [5.40, 7.19] \wedge X_4 \in [1.60, 2.67]$) **THEN** Iris-virginica

A 100% equivalence is obtained with the trained SVM for these two sets of rules and no test data is classified wrong.

3 Eliminating Randomness from the Clustering Algorithm

It has been realized during the experimentation that the method is very sensible to the used prototype vectors and, therefore, the quality and amount of the obtained rules varies depending on the location of the centres of the ellipsoids. These centres have been obtained from an initial solution provided by the clustering algorithm based on k-means, which randomly depends on the ordination of the provided training points (Duda et al. 2001). From the point of view of the extraction method this randomness is an obstacle, especially if it is necessary to extract several rules by class. Hence, it is required to apply the method several times on the trained SVM (with different initial conditions for the clustering algorithm) to be able to obtain a good solution, because the set and performance of the extracted rules show a high variance from an experiment to another one.

This situation of randomness and dependency of the method on the centres is a problem to be solved. It would be possible to evaluate in an empirical form different clustering algorithms for each database and to select that providing a greater stability. Nevertheless, a novel direct technique will be proposed for the determination of unique initial conditions for the clustering algorithm based on the support vectors provided by the learning machine. In short, the algorithm works as follows: if m prototypes are needed for the j-labelled class in the k-th iteration of the extraction algorithm, the algorithm proposes clustering the available data around m support vectors selected according to some criterion; once established the disjoint partitions, the midpoint of each one of them would be an initial centre for the clustering algorithm (Núñez et al. 2002c).

Let SV_{jk} be the set of support vectors in class j for the iteration k and let D_{jk} be the set of data in class j for the iteration k, then:

- Select m support vectors from SV_{jk}
- Determine initial partitions P_i, assigning each instance in D_{jk} to the nearest support vector according to the Euclidean distance,

$$\mathrm{P_i} \leftarrow \mathbf{x},\ \text{if } {}^{\mathrm{d}}(\mathbf{x}, \mathbf{sv_i}) = \underset{\mathrm{p=1...m}}{\operatorname{argmin}} \left[\mathrm{d}\left(\mathbf{x}, \mathbf{sv_p}\right)\right] \quad \forall \mathbf{x} \in \mathrm{D_{jk}} \tag{41}$$

– Next, calculate the midpoint for each partition,

$$u_i = \frac{\sum_{r=1}^{n_i} x_r}{n_i}\ , \tag{42}$$

where $\mathrm{n_i}$ is the number of patterns in the partition $\mathrm{P_i}$. The points $\mathbf{u_i}$ will determine the initial conditions for the selected k-means algorithm

The criteria that could be used for the selection of the support vectors are the following ones (where it is only taken into account those vectors without error, i.e. with a value for the associated α lower than C):

- *Scheme of partition* EP_1: Support vectors are ordered in a descendent way according to its average similarity with data in the class (Kaufman and Rousseeuw 1990), so that the first m support vectors are selected.
- *Scheme of partition* EP_2: The m closest support vectors are chosen; it is aimed with this heuristic that the initial partitions are directed to the zones with greater curvature of the decision surface defined by the SVM.
- *Scheme of partition* EP_3: Support vectors are ordered in descendent form according to the value of the parameter α and the first m vectors are selected, on the hypothesis that larger is the value of the parameter, more informative is the associated pattern (Guyon et al. 1996).

For all these schemes of partition, in the case that only q support vectors are available with $q < m$, then the remaining vectors will be determined form the set $\mathrm{D_{jk}}$ by selecting those $m - q$ patterns with higher average similarity to the data.

One of these criteria has been empirically evaluated on trained support vector machines with the real databases of the UCI repository. In order to establish a direct comparison with the previously exposed results in Sect. 2, the same performance parameters were used, and they were identically calculated. Contrarily to the precedent results, now they are obtained by running a single iteration for the extraction algorithm, because SVM are unequivocally determined and so the centres from the proposed clustering procedure.

Table 5 shows the results obtained by using the scheme of partition EP_1. It can be observed that the performance is in average very similar to that obtained in Sect. 2. Therefore, using some of these schemes is a valid alternative route to be considered for generating the set of rules from a trained SVM trained in a determinist form. Possible extensions could be considered by defining some hybrid approaches that uses more than a partition scheme, with a decision module selecting the best set of generated rules.

Table 5. Performance values for the set of rules applied on each database using the scheme of partition EP_1

Database	Error SVM	Equation-type rules				Interval-type rules			
		Error	Equ.	Cov.	NR	Error	Equ.	Cov.	NR
1	0.046	0.041	98.00	80.40	6.4	0.037	97.00	81.40	4.5
2	0.045	0.039	97.85	86.03	16.1	0.042	96.00	90.33	15.9
3	0.023	0.020	98.33	78.40	6.2	0.024	92.90	79.42	9.8
4	0.022	0.022	100.00	25.00	6.6	0.020	98.00	73.00	6.2
5	0.052	0.048	97.04	78.46	8.0	0.045	94.82	72.03	11.3
6	0.002	0.003	96.51	29.81	25.4	0.009	99.10	97.56	31.4
7	0.102	0.143	90.78	20.01	12.0	0.112	96.01	56.34	32.00
8	0.023	0.025	96.99	40.05	8.0	0.022	99.53	95.60	11.00
9	0.042	0.044	99.02	29.33	10.3	0.045	98.16	79.00	8.8
10	0.164	0.160	97.13	57.40	6.5	0.167	97.30	62.24	19.3

4 Conclusions and Further Research

A method has been developed transforming the knowledge captured by a support vector machine during its learning in a representation based on rules, with the aim of equipping it with the capacity of explanation.

The algorithm proposed for the extraction of rules is based on the combination, using geometry elements, of the support vectors obtained from the SVM with prototype vectors derived from a clustering training regime to determine a set of ellipsoidal regions in the input space, later transferred to rules in the form of equation or interval rules. The hypothesis lying in this hybrid procedure is that, when using the support vectors, the defined regions adjust to the shape of the separation surface defined by the SVM with a minimal overlapping between classes.

An iterative procedure is followed for determining the set of rules, starting with the construction of a general ellipsoid that is successively specialized in more reduced ellipsoids in order to fit the shape of the decision function determined by the SVM. The partition criterion is also based on both, the support vectors and the decision function. The final number of rules, derived from the ellipsoids or hyper-rectangles, can be defined either, externally or through a stopping performance criterion.

Experimental results obtained when applying the rule extraction method on real databases from different domains shown a high degree of equivalence between the SVM and the extracted set of rules on test patterns. It can be so concluded that the proposed method is able to cope the acquired knowledge of the SVM during the learning phase.

None requirement is imposed in the initial method derivation about specific training regimes employed to train the SVM, kernel functions, nor clustering algorithm. Nevertheless, experimentation demonstrated that the quality and number of the generated rules with this method is highly dependent, due to

the randomness of the clustering algorithm, on the location of the prototype vectors to be used as centres of the regions. A good average solution is only provided by the algorithm after some iteration with different initial conditions for the clustering algorithm.

A totally novel solution has been proposed to this problem by determining unequivocally the initial conditions from the unique set of support vectors. Three associated schemes of partition have been proposed to initialize the proposed clustering algorithm and build in a deterministic form the set of rules from a trained SVM. Empirical results showed the opportunity of such schemes or a hybridization of them to reduce the sensibility of the method to the clustering algorithm.

Starting from these proposed solutions to increase the explicative power of a learned SVM in the form of a set of rules, it is possible to plan new developments. For example, it would be interesting to study a direct extension of the rule extraction method to regression problems. In reference to the representational language, it could be studied using another one to express the new model, for example generating fuzzy rules from the ellipsoids. It would be also profitable developing algorithms for rule simplification that can be applied to improve, when it is required, the understand ability of the knowledge that has been extracted of the SVM.

Finally, it should be realized that the method can be extended to extract rules of other models, such as radial basis function networks (RBFN). The extraction algorithm could be designed so that the prototype vectors would be replaced by the centres of the RBF nodes (Núñez et al. 2002b), and the borders of the rules or their activation rates would be determined by the support vectors.

References

Andrews R, Diederich J, Tickle AB (1995) A Survey and Critique of Techniques For Extracting Rules From Trained Artificial Neural Networks, Knowledge Based Systems, 8, pp. 373–389

Berthold M, Hand D (1999) Intelligent Data Analysis An Introduction. Springer-Verlag

Blake CL, Merz CJ (1998) UCI Repository of Machine Learning Data-Bases. University of California, Irvine. Dept. of Information and Computer Science. (http://www.ics.uci.edu/∼mlearn/MLRepository.html)

Cortes C, Vapnik V (1995) Support-Vector Networks. Machine Learning 20:273–297

Craven M, Shavlik J (1997) Using Neural Networks for Data Mining. Future Generation Computer Systems 13:211–229

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

Domingos P (1991) Unifying Instance-Based and Rule-Based Induction. Machine Learning 24:141–168

Duda R, Hart P, Stork D (2001) Pattern Recognition. 2nd edn. John Wiley & Sons, Inc

Guyon I, Martíc N, Vapnik V (1996) Discovery Information Patterns and Data Cleaning. In: Fayyad V, Piatetsky G, Smyth P, Uthurusamy R (eds). Advances in Knowledge Discovery and Data Mining. MIT Press

Kaufman L, Rousseeuw PJ (1990) Finding Groups in Data. An Introduction to Cluster Analysis. John Wiley & Sons, Inc

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

Mitchell T (1997). Machine Learning. McGraw-Hill

Mitra S, Pal SK, Mitra P (2002) Data Mining in Soft Computing Framework: A survey. IEEE Transactions on Neural Networks 13(1):3–14

Núñez H, Angulo C, Català A (2002a) Rule extraction from support vector machines. Proc. 10th European Symposium on Artificial Neural Networks, pp. 107–112

Núñez H, Angulo C, Català A (2002b) Rule Extraction from Radial Basis Function Networks by Using Support Vectors. Lecture Notes in Artificial Intelligence 2527:440–449

Núñez H, Angulo C, Català A (2002c) Support Vector Machines with Symbolic Interpretation. 7th Brazilian Symposium on Neural Networks, IEEE, pp. 142–147

Núñez H, Angulo C, Català A (2003) Hybrid Architecture based on Support Vector Machines. Lecture Notes in Computer Science 2686:646–653

Salzberg S (1991) A Nearest Hyper rectangle Learning Method. Machine Learning 6:251–276

Strang G (1998) Introduction to linear algebra. 3rd. edition. Wellesley-Cambridge Press

Tickle A, Andrews R, Mostefa G, Diederich J (1998) The Truth will come to light: Directions and Challenges in Extracting the Knowledge Embedded within Trained Artificial Neural Networks. IEEE Transactions on Neural Networks 9(6):1057–1068

Tickle A, Maire F, Bologna G, Andrews R, Diederich J (2000) Lessons from Past, Current Issues, and Future Research Directions in Extracting the Knowledge Embedded Artificial Neural Networks. In: Wermter S, Sun R (eds) Hybrid Neural Systems. Springer-Verlag

Vapnik V (1998) Statistical Learning Theory. John Wiley & Sons, Inc

Witten I, y Frank E (2005) Data Mining. Practical Machine Learning Tools and Techniques with Java Implementations. Second edition. Morgan Kaufmann Publishers

Zhou Z (2004) Rule Extraction: Using Neural Networks or For Neural Networks? Journal of Computer Science and Technology. 19(2):249–253