Chapter 18
Rough Support Vectors: Classification, Regression, Clustering

Pawan Lingras, Parag Bhalchandra, Cory Butz, and S. Asharaf

Abstract. Support vector techniques were proposed by Vapnik as an alternative to neural networks for solving non-linear problems. The concepts of margins in support vector techniques provides a natural relationship with the rough set theory. This chapter describes rough set theoretic extensions of support vector technologies for classification, prediction, and clustering. The theoretical formulations of rough support vector machines, rough support vector regression, and rough support vector clustering are supported with a summary of experimental results.

Keywords: Support vector machines, clustering, prediction, classification, rough sets, rough patterns, financial modeling, conservative and aggressive modeling, regression, ε-insensitive loss function.

18.1 Introduction

Support vector machines (SVMs) are based on the statistical learning theory, also called *Vapnik-Chervonenkis* (VC) theory, developed by Vapnik and colleagues [3, 7, 25, 26, 27, 28]. Classification based on SVM's extends single layer perceptrons for non-linearly separable data_sets with the help of kernel functions and optimization. The original proposal for classification was generalized to regression, followed by clustering. The concept of margin that was central to the support vector based classification is a natural conduit to its fusion with rough set theory. The margin can be

Pawan Lingras
Saint Mary's University Halifax, Canada

Pawan Lingras · Parag Bhalchandra
Swami Ramanand Teerth Marathwada University, Nanded, India

Cory Butz
University of Regina, Regina, Canada

S. Asharaf
Indian Institute of Management, Kozhikode, India

interpreted as the boundary region in rough set theory. Lingras and Butz [11, 12] described a rough set-based interpretation of support vector classification and proposed an efficient multi-classification extension based on rough set theory. A similar margin in support vector regression created with the help of ε-loss function led to the proposal for rough support vector regression by Lingras and Butz [13]. Ashraf et al. [1] extended the support vector clustering(SVC)[2] to propose a rough support vector clustering algorithm that can create irregular-shaped soft clusters with lower and upper bounds. This chapter provides a comprehensive introduction to the hybridization of rough set theory and support vector techniques for classification, regression, and clustering. Theoretical foundations are followed by algorithmic implementation as well as experimental results to provide reader with sufficient background to pursue further developments in rough support vector computing.

18.2 Rough Support Vector Machines for Multi-classification

Support vector machines (SVMs), proposed by Vapnik [25, 26, 28, 27], are used for creating functions from a set of labeled training data [21]. The function can be a classification function with binary outputs or it can be a general regression function. In this section, we will restrict ourselves to classification functions. For classification, SVMs operate by attempting to find a hypersurface in the space of possible inputs that splits the positive examples from the negative examples. The split will be chosen to have the largest distance from the hypersurface to the nearest of the positive and negative examples. Intuitively, this makes the classification correct for testing data that is near, but not identical, to the training data.

Let $\mathbf{x}$ be an input vector in the input space X. Let y be the output in $Y \in \{-1,1\}$. Let $S = \{(\mathbf{x}_1,y_1),(\mathbf{x}_2,y_2),\ldots,(\mathbf{x}_i,y_i),\ldots\}$ be the training set used for supervised prediction. Let us define the inner product of two vectors $\mathbf{x}$ and $\mathbf{w}$ as

$$\langle \mathbf{x}, \mathbf{w} \rangle = \sum_j x_j \times w_j, \tag{18.1}$$

where x_j and w_j are components of the vectors $\mathbf{x}$ and $\mathbf{w}$, respectively.

The linear separability restriction in perceptron is overcome by the use of a non-linear transformation Φ as shown in Fig. 18.1(a). Here, we are using the class labels $\{1,2\}$ instead of $\{-1,1\}$, since we will be generalizing the binary classification problem to multi-classification.

The choice of the hyperplane in perceptron algorithm was arbitrary. SVMs use the size of margin between two classes to search for an optimal hyperplane, which bifurcates the maximum margin. Fig. 18.1(b) shows the concept of margin between two classes. The line with intercepts of b_1 and b_2 enclose the maximum margin between two classes. The line with intercept b is the optimal line separating the two classes. The problem of maximizing the margin can be reduced to an optimization problem [5,28]:

$$\text{Minimize } \langle \Phi(\mathbf{w}), \Phi(\mathbf{w}) \rangle y \times \langle \Phi(\mathbf{x}), \Phi(\mathbf{w}) \rangle \geq 0, \forall (\mathbf{x}, y) \in S \tag{18.2}$$

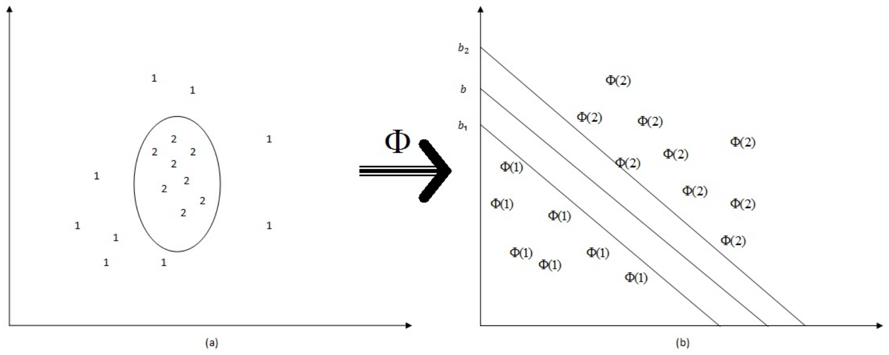


Fig. 18.1. Support Vector Machine Transformation to Linearly Separable Space

Usually, a high dimensional transformation is needed in order to obtain a reasonable prediction [30, 31]. Computational overhead can be reduced by not explicitly mapping the data to the feature space, but instead just working out the inner product in that space. In fact, SVMs use a kernel function K corresponding to the inner product in the transformed feature space as $K(\mathbf{x},\mathbf{w}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{w}) \rangle$.

Vapnik recognized the margin separating the two classes as an important issue in further theoretical developments. Lingras and Butz [12] described rough set-based interpretation of SVM binary classifications, where the margin between hyperplanes with intercept of b_1 and b_2 represent the boundary region between the two classes. Lingras and Butz described an algorithm for the calculation of b_1 and b_2 [12].

This rough set interpretation of SVM binary classification will allow us to create three equivalence classes, two equivalence classes for the lower bounds of each of the two classes, and third for the boundary region and define a rough set-based approximation space. This simple extension of an SVM classifier provides a basis for a more practical application, when the SVM transformation does not lead to a linear separable case. Cristianini [5] lists disadvantages of refining feature space to achieve linear separability. Often this will lead to high dimensions, which will significantly increase the computational requirements. Moreover, it is easy to overfit in high dimensional spaces, that is, regularities could be found in the training set that are accidental, which would not be found again in a test set. The soft margin classifiers [5] modify the optimization problem to allow for an error rate. In such a case, boundary region allows us to define an area of ambiguity. The error rate can also be used to determine the values of b_1 and b_2 in such a way that a large percentage of objects below the hyperplane given by b_1 belong to class 1, and an equally large percentage of objects below the hyperplane given by b_2 belong to class 2. This concept of soft margin was easily incorporated by Lingras and Butz in the algorithm for the calculation of b_1 and b_2. A rough set-based SVM binary classifier can then be defined by rules:

(R1) If $\langle \Phi(\mathbf{x}), \Phi(\mathbf{w}) \rangle + b_2 \geq 0$ then $\mathbf{x}$ belongs to class 2.
(R2) If $\langle \Phi(\mathbf{x}), \Phi(\mathbf{w}) \rangle + b_1 \leq 0$ then $\mathbf{x}$ belongs to class 1.
(R3) Otherwise, classification of $\mathbf{x}$ is uncertain

18.2.1 Extending Binary SVM's for Multi-classification

The SVM binary classification is extended to mulitclassification using two popular techniques, 1-v-1 and 1-v-r. The "one versus one" (1-v-1) strategy, also known as pair wise coupling, consists of constructing one SVM for each pair of classes. Thus, for a problem with n classes, $n \times (n-1)/2$ SVMs are trained to distinguish the samples of one class from the samples of another class. While doing so, the classification of an unknown pattern is done according to the maximum voting, where each SVM votes for one class. This makes 1-v-1 more accurate.

The "one versus rest" (1-v-r) strategy consists of constructing one SVM per class, which is trained to distinguish the samples of one class from the samples of all remaining classes. Usually, classification of an unknown pattern is done according to the maximum output among all SVMs. The total training time with the 1-v-1 strategy is larger than with the 1-v-r, because it is necessary to train more binary classifiers; but it is not necessarily true when the binary classifiers are SVMs. On the other hand, the individual binary classifiers in 1-v-1 are trained on a smaller sample set.

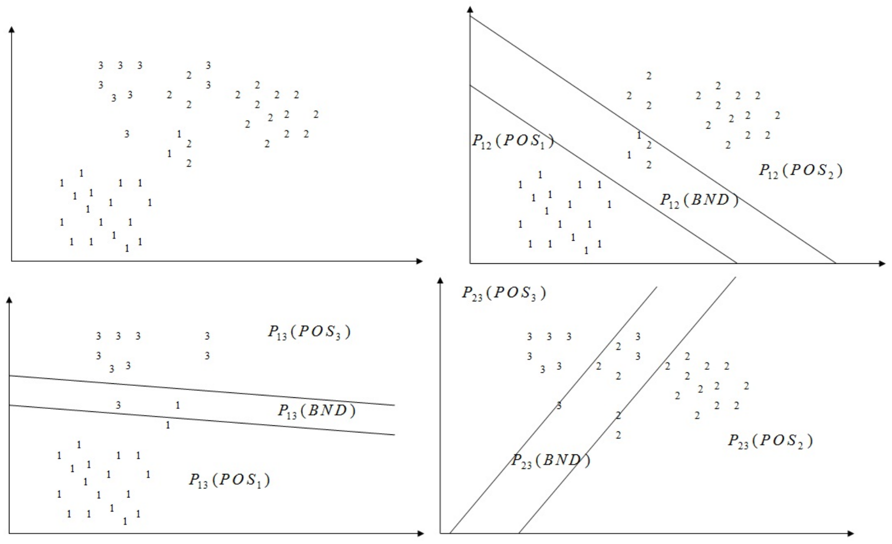


Fig. 18.2. Rough set-based 1-v-1 multi-classification

18.2.1.1 Rough Set-Based 1-v-1 Approach

Knerr et al. [8] suggested combining binary classifiers with an AND gate. While there are many other attempts to manage the large number of classifiers created by 1-v-1 approach, DAGSVM proposed by Platt et al. [22] is particularly interesting and popular. DAGSVM uses directed acyclic graphs to reduce the number of SVMs that need to be used during the testing and operational phase.

Lingras and Butz [12] provided a rough set interpretation of 1-v-1 classification that can lead to lower storage and lower computations in the operational phase. Fig. 18.2 depicts the approach for a three class problem. In order to understand the illustration, let us define some additional terms for pairwise classifications between class i and j in the 1-v-1 approach. For each pair, (i, j), that is separated by a binary SVM, we define three equivalence classes, let $P_{ij}(POS_i)$ and $P_{ij}(POS_j)$ be the lower bounds or positive regions of classes i and j, respectively. These two correspond to rules (R1) and (R2) Let $P_{ij}(BND)$ be the boundary region between the two classes corresponding to rule (R3). All the 1-v-1 classifiers can be combined to create lower bound and boundary regions for each class i as follows:

$$lower(class_i) = \bigcap_{j=1,\cdots,n,j\neq i} P_{ij}(POS_i) \tag{18.3}$$

$$upper(class_i) = \bigcup_{j=1,\cdots,n,j\neq i} (P_{ij}(BND) - lower(class_j)) \tag{18.4}$$

Detailed derivation of the above formulas can be found in [12].

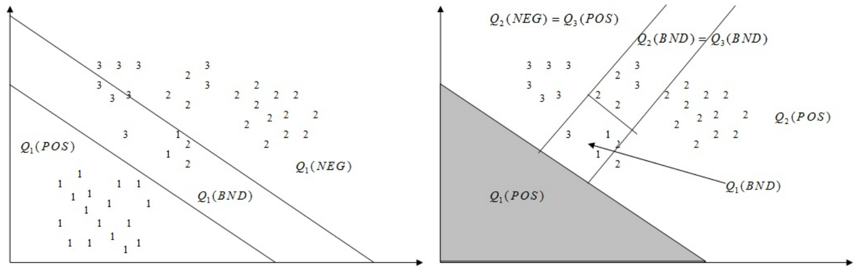
The rough set-based approach uses the same training time as the classical 1-v-1 or DAGSVM. It should perhaps be noted here that the upper bound given by eq. (18.4) will be polygonal, whose calculation needs additional computations. One of the advantages of rough set approach is that only two rules need to be stored for each class, one corresponding to the lower bound and another corresponding to the upper bound, as shown in the eq. (18.3) and eq. (18.4). Therefore, a total of $2 \times n$ rules are stored for the testing and operational phases, as opposed to $n \times (n-1)/2$ SVMs stored by 1-v-1 and DAGSVM approaches. Moreover, during the operational phase, the determination of membership of an object in a class will involve simply testing which lower bound the object belongs to. The time requirement for classification in the operational phase will be linear, that is, $O(n)$, the same as DAGSVM.

18.2.1.2 Rough Set-Based 1-v-r Approach

Without loss of generality, let us order the n classes such that class i contains at least as many objects as class $i+1$, where $1 \leq i < n$ in the training sample. Using the entire training sample and 1-v-r strategy, for $class_1$, create three equivalence classes $Q_1(POS)$ which is lower bound of $class_1$, $Q_1(NEG)$ is the lower bound of the collection of the rest of the classes, and $Q_1(BND)$ is the boundary region.

Since $Q_1(POS)$ definitely belongs to $class_1$, we can eliminate it from the next application of SVM classifier, that is, for the subsequent classes. For each subsequent class i, $1 < i < n$, refine $Q_{i-1}(BND) \cup Q_{i-1}(NEG)$ by creating $Q_i(POS)$, $Q_i(BND)$, $Q_i(NEG)$. For the last class, $Q_n(POS) = Q_{n-1}(NEG)$ and $Q_n(BND) = Q_{n-1}(BND)$. The upper and lower bounds for all n classes are defined as: $lower(class_i) = Q_i(POS)$ and $upper(class_i) = Q_i(POS) \cup Q_i(BND)$. As it is possible that some of the boundary regions may overlap with positive regions for subsequent classes, recalculate values of each of the boundary classes as:

$$Q_i(BND) = Q_i(BND) - \bigcup_{j=i}^{n} Q_j(POS). \tag{18.5}$$



(a) (b)

Fig. 18.3. Rough set-based 1-v-r multi-classification

Fig. 18.3 illustrates the rough-set based 1-v-r approach. For the $class_1$, $Q_1(POS)$, $Q_1(BND)$, $Q_1(NEG)$ are calculated using an SVM as shown in Fig. 18.3(a). As $Q_1(POS)$ only contains objects belonging to $class_1$, there is no need to further classify objects in $Q_1(POS)$. However, $Q_1(BND) \cup Q_1(NEG)$ should be further refined. Thus, for the next class ($class_2$), the resulting classification is shown in Fig. 18.3(b). Note that the shaded triangular area ($Q_1(POS)$) in Fig. 18.3(b) was eliminated from further classification, since it definitely belongs to $class_1$. 1-v-r classification allows us to identify the objects that definitely belong to $class_2$. In general case, the process will be further repeated until number of classes is reduced to two. In our example, we stop after applying the 1-v-r classification to class 2, since we have already classified $class_3$. That is, according to our algorithm, $Q_2(NEG) = Q_3(POS)$ and $Q_3(BND) = Q_2(BND)$. Fig. 18.3(b) shows the final classification.

An important feature of the rough set-based 1-v-r approach is that the sample size is reduced in each subsequent step. The reduction is optimized since the classes are ordered according to their cardinalities with the largest cardinality first. The lower bound (the positive region) of the largest class is eliminated from further classification, followed by the next largest class, and so on. By reducing the size of training set, this elimination process may increase the training performance over the traditional 1-v-r approach. Lingras and Butz [12] provide detailed calculations on the extent of the reduction in training time.

18.2.2 Experiments with 1-v-1 and 1-v-r Approach

Lingras and Butz [12] demonstrated the implementation of rough set-based 1-v-1 and 1-v-r approaches for a synthetic data_set shown in Fig. 18.4 consisting of 150 objects. These 150 objects belong to three classes with each class containing 50 objects. As can be seen from Fig. 18.4, it is possible to separate most of the objects from these three classes using lines, but there will be a certain percentage of false positives and negatives that should belong to the boundary regions.

The experiments were carried out using Gist (http://svm.sdsc.edu/cgi-bin/nph-SVMsubmit.cgi), which provides values of discriminants, which can be used as surrogates of the distances from the hyperplane separating the positives from negatives. Usually, a positive discriminant corresponds to the object above the hyperplane, and negative discriminant indicates an object below the hyperplane. Therefore, we do

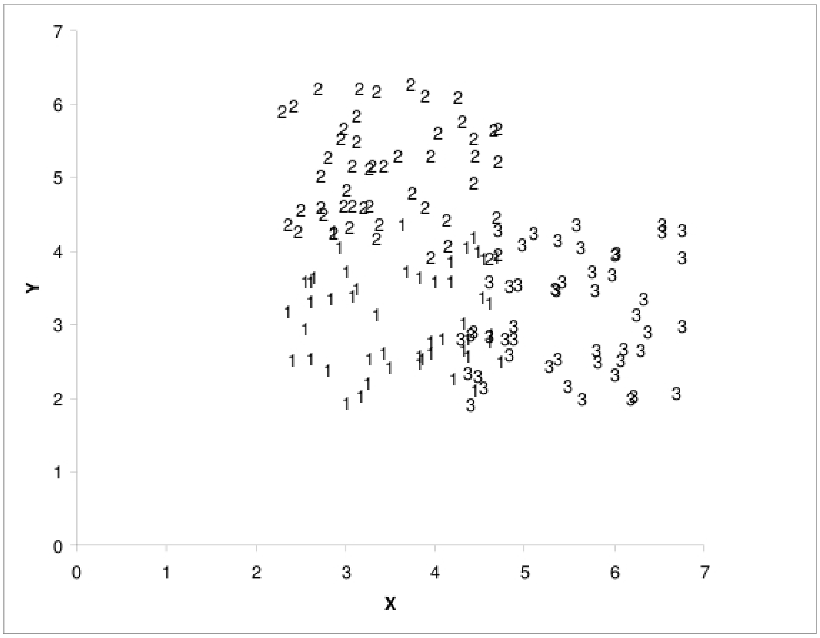


Fig. 18.4. Synthetic data for multi-classification

Table 18.1. Results of 1-v-1 classification

Region	Criteria	Cardinality
$P_{12}(POS_1)$	$0.2796 \leq d12$	35
$P_{12}(BND)$	$0.2796 > d12 > -0.3897$	32
$P_{12}(POS_2)$	$d12 \leq -0.3897$	33
$P_{13}(POS_1)$	$0.2767 \leq d13$	28
$P_{13}(BND)$	$0.2767 > d13 > -0.2105$	39
$P_{13}(POS_3)$	$d13 \leq -0.2105$	33
$P_{23}(POS_2)$	$0.083 \leq d23$	48
$P_{23}(BND)$	$0.083 > d23 > -0.2015$	3
$P_{23}(POS_3)$	$d23 \leq -0.2015$	49

not have to explicitly determine b_1 and b_2. More details about the use of discriminants can be found in [12].

18.2.2.1 Experimental Results Using 1-v-1

Let us use d12 to denote the discriminant for 1-v-1 classification of classes 1 and 2. Similarly, we also use d13 as discriminant for classification of classes 1 and 3, and d23 as discriminant for classification of classes 2 and 3. The results from application of rough set-based 1-v-1, shown in Table 18.1, can be combined using eq. 18.3 and eq. 18.4 to write the rules for upper and lower bounds of each class as follow:

- If $0.2796 \geq d12$ and $0.2767 \geq d13$ then, the object belongs to $lower(class_1)$
- If $d12 \leq 0.3897$ and $0.083 \geq d23$ then, the object belongs to $lower(class_2)$
- If $d13 \leq 0.2105$ and $d23 \leq 0.2015$ then, the object belongs to $lower(class_3)$

Similarly,

- If $d12 > 0.3897$ or $d13 > 0.2105$ then, the object belongs to $upper(class_1)$
- If $0.2796 < d12$ or $d23 > 0.2015$ then, the object belongs to $upper(class_2)$
- If $0.2767 < d13$ or $0.083 < d23$ then, the object belongs to $upper(class_3)$

As we can see, we only need to store six rules for the three classes. However, as mentioned before the rules can be complicated. Simplification and generalization of these rules is a promising research direction.

18.2.2.2 Experimental Results Using 1-v-1

Lingras and Butz [12] confirmed that 1-v-r classifications tend to produce less accurate results. Since all the classes had the same number of objects, we can choose any order for application of 1-v-r SVMs. We randomly choose the order $class_2, class_3$, $class_1$, The best results are found for $class_2$ as shown in Table 18.2, where d_{2r} is

Table 18.2. Results of 1-v-r classification

Region	Criteria	Cardinality
$Q_2(POS)$	$0.584 \leq d_{2r}$	33
$Q_2(BND)$	$0.584 > d_{2r} > -0.153$	32
$Q_2(NEG)$	$d_{2r} \leq -0.153$	85
$Q_3(POS)$	$0.3521 \leq d_{3r}$	28
$Q_3(BND)$	$0.3521 > d_{3r} > -0.2473$	44
$Q_3(NEG)$	$d_{3r} \leq -0.2473$	45

the discriminant of the 1-v-r classification to our synthetic data_set for $class_2$. Similarly, d_{3r} is the discriminant of the 1-v-r classification for $class_3$ after eliminating the positive region from $class_2$. We do not need to apply the 1-v-r classification to $class_1$, because the negative region of 1-v-r classification for $class_3$ can be used as the lower bound for $class_1$. Table 18.2 shows the regions, their criteria, and cardinalities which lead to the following rules:

- If $0.584 \geq d_{2r}$ then, the object belongs to $lower(class_2)$.
- If $0.3521 \geq d_{3r}$ then, the object belongs to $lower(class_3)$.
- If $d_{3r} < -0.2473$ then, the object belongs to $lower(class_1)$.

Similarly,

- If $d_{2r} > 0.153$ and $0.3521 > d_{3r} > -0.2473$ then, the object belongs to $upper(class2)$.
- If $0.584 < d_{2r}$ and $d_{3r} > 0.2473$ then, the object belongs to $upper(class3)$.
- If $0.584 < d_{2r}$ and $0.3521 < d_{3r}$ then, the object belongs to $upper(class_1)$.

It should be noted that the number of false positives and negatives in the lower bounds of all the classes are very high, compared with the 1-v-1 classification.

18.2.2.3 Semantics of Rough Set-Based Multi-classification

The rules corresponding to lower and upper bounds may provide better semantic interpretations of the multi-classification process than the other SVM approaches, which have been regarded as black-box models [23]. This is important, since da Rocha and Yager [23] advocate that describing the relationship between black-box approaches like SVMs with the logical rules approaches can lead to semantically enhanced network-based classifiers. It should be noted that the rules created by the application of Rough Set-based SVM cannot generally be written as a decision table, similar to conventional rough set approaches. However, the rules may still enable us to semantically analyze the classification process, especially if it were possible to associate semantic meaning to the discriminate values used in the rules developed in this section.

18.3 Dual Rough Support Vector Regression

Support vector regression (SVR) employs the margin concept for the regression problem with the help of ε-insensitive loss functions [24, 26]. SVR has been found especially useful in time series predictions [16, 17, 30, 31].

The following is a brief summary of SVR as described in detail by [30, 31]. We will use the same notations as SVM classifiers, except let $y \in \Re$. Furthermore, let $f(\mathbf{x})$ be a predicted value of y:

$$f(\mathbf{x}) = \langle \Phi(\mathbf{x}), \Phi(\mathbf{w}) \rangle + b. \tag{18.6}$$

The objective of SVR is to minimize the regression risk

$$R(f) = \frac{1}{2}\langle \Phi(\mathbf{w}), \Phi(\mathbf{w}) \rangle + C \sum_{i=1} l(f(\mathbf{x}_i), y_i), \tag{18.7}$$

where C is called the cost of error. The first term $\frac{1}{2}\langle \Phi(\mathbf{w}), \Phi(\mathbf{w}) \rangle$ can be seen as the margin in SVMs. The similarity between actual y and its prediction is given by the loss function $l(f(\mathbf{x}), y)$.

Vapnik [26] proposed an ε-insensitive loss function:

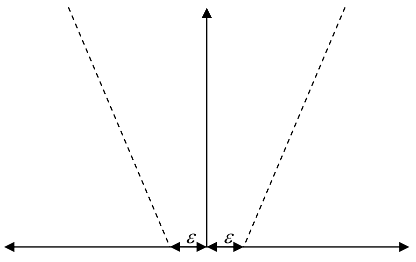
$$l_\varepsilon(f(\mathbf{x}), y) = \max(0, |y - f(\mathbf{x})| - \varepsilon) \tag{18.8}$$

shown in Fig. 18.5(a). The vertical axis denotes the loss. The horizontal axis corresponds to the value of $f(\mathbf{x})$. The two axes meet at $f(\mathbf{x}) = y$. If the predicted value is within $\pm\varepsilon$ of the actual value, the prediction is considered lossless. Fig. 18.5(b) shows how the actual values in the margin around the predicted function are considered acceptable or error-free. Increasing the ε value, reduces the number of support vectors. A large enough value of ε will lead to a constant regression function. The ε-insensitive loss function is ideally suited for modeling rough values as can be seen by the ε-tube around the prediction function in Fig. 18.5(b).

The corresponding SVR is called an ε-SVR. The minimization of Eq. (18.7) is reduced to a quadratic programming problem. The details of the formulation can be found in [30, 31]. Lingras and Butz [14] proposed dual extensions of SVR for modeling the rough patterns.

18.3.1 Rough Patterns

Pawlak proposed the concept of rough real functions which can be useful for rough controllers [19]. The notion of rough real functions was defined as an approximate value in case exact values cannot be represented by the given set of values. However, the notion can be used in a broader context. Lingras [9] used the rough values



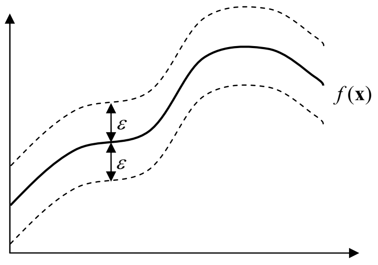


Fig. 18.5. Support Vector Regression

to develop supervised and unsupervised neural networks [10, 20] and genetic algorithms [15]. This section describes rough values and patterns.

In some cases, a precise value of an entity may not be available. For example, one may estimate the current temperature to be between 20 and 25° C. In other cases, it may not even be possible to state a precise value. Many spatial (rainfall in Nova Scotia) or temporal (daily temperature) variables fall in this category. We cannot associate a precise value for daily temperature, only a range of value using the highest and lowest temperatures recorded on that day. We use rough or interval values to measure such quantities. For continuous variables, rough values are special cases of intervals as they focus on the end points. However, unlike intervals, rough values can also be used to represent a set of discrete values using the minimum and maximum values in the set. Let $Y = \{y_1, y_2, ..., y_n\}$ be a set of values collected for a variable such as daily temperature or stock market index. Each *rough value y* is denoted by a pair $(\underline{y}, \overline{y})$:

$$\underline{y} = \inf\{y \in Y\},$$

and

$$\overline{y} = \sup\{y \in Y\}.$$

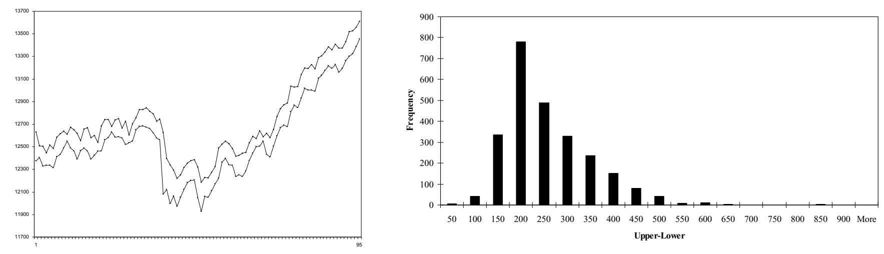
Here, sup is defined as the maximum value from the set, while inf corresponds to the minimum value. The definitions of inf and sup can be modified to exclude the outliers. For example, one could use the bottom 5th percentile value for $\underline{y}$ and top 5th percentile value for $\overline{y}$. The above definition by Pawlak accommodates sets with continuous as well as discrete values. If the values are continuous, the set will be infinite and the resulting rough values correspond to the conventional notion of interval.

Rough patterns are sequences of rough or interval values [9]. We will look at a real world example of a rough pattern using a stock market index.

The *Dow Jones Industrial Average* (DJIA) is an index based on stock prices of the 30 most prominent companies listed on U.S. stock exchanges such as NYSE and NASDAQ. It is one of the most closely watched stock market indices in the world. The data used in this study was obtained from Yahoo! (www.yahoo.com). It consisted of the date, opening value, closing value, high and low values, as well as the number of shares traded on the exchange. Data is only available for the trading days, that is, when the New York stock exchange was open. For example, in the ten years from May 21, 1997 to May 20, 2007, there were 2514 trading days.

Most of the prediction models are based on the closing values. The closing value of a stock or stock market index has impact on secondary investment instruments, such as mutual fund values and overseas markets. However, the traders on the New York stock exchange are reacting to minute-by-minute changes to stock prices, in addition to key indices like the DJIA. The stock exchanges are open from 10 a.m. to 3:30 p.m. from Monday through Friday with the exception of holidays. During these five and a half hours, one can get minute-by-minute updates on the values of DJIA. That will mean a total of 330 values per day. It will be difficult to manage such a large amount of data in any financial modeling. It is neither possible nor necessary to model/predict minute-by-minute values of the index. The traders, however, are interested in knowing how high or low a stock or index may go on a given day. For example, a trader who is looking to sell a stock or DJIA based financial derivative may wait until the high for the day is reached. Conversely, a trader who is looking to buy a stock or DJIA based financial derivative may wait until the low for the day is reached. Therefore, accurate prediction of trading range given by the rough pattern for a stock or stock index is an important part of stock market analysis.

Fig. 18.6(a) shows the rough pattern for the daily values of the DJIA from January 1 to May 20, 2007. The DJIA rough pattern consists of two curves. The top curve corresponds to the daily high's and the bottom one corresponds to the daily low values.



(a) High and Low patterns

(b) Frequency of differences between High's and Low's

Fig. 18.6. Dow Jones Industrial Average (DJIA) for early 2007

It is important to realize that there can be a considerable variation in the difference between high and low values, even though the general trend of the high and low values is essentially the same. Analysis of ten years of data from May 21, 1997 to May 20, 2007 shows the minimum difference to be 34.42, with a maximum value of 848.52. Fig. 18.6(b) shows the distribution of differences between highs and lows to be more or less normal. The average of the difference is 232, which is close to the median and mode values. This analysis suggests that the high and low values should be separately analyzed.

18.3.2 Conservative and Aggressive Modeling of Rough Patterns

The formulation of rough ε-SVR to model rough patterns needs a definition of $\pm$ in the context of rough values. Let us assume that y is a rough value given by $(\underline{y}, \overline{y})$. Lingras and Butz [14] described dual approaches termed as conservative and aggressive. In the conservative approach, the tube of predicted values will be tucked inside the tube of actual values. That is, the actual lower value $\underline{y}$ will not be higher than $\underline{f}(\mathbf{x})$ and the actual upper value $\overline{y}$ will not be lower than $\overline{f}(\mathbf{x})$. Let $fc(\mathbf{x}) = (\underline{fc}(\mathbf{x}), \overline{fc}(\mathbf{x}))$ be the conservative prediction of y such that

$$y \;=\; (\underline{fc}(\mathbf{x}) - \varepsilon, \overline{fc}(\mathbf{x}) + \varepsilon). \tag{18.9}$$

The aggressive model, on the other hand, will tell us how much lower the lower value can drop and how much higher the upper value can rise. The *aggressive* prediction, denoted fa, for a rough value $fa(\mathbf{x}) = (\underline{fa}(\mathbf{x}), \overline{fa}(\mathbf{x}))$ is more formally described as:

$$fa(\mathbf{x}) \;=\; (\underline{y} - \varepsilon, \overline{y} + \varepsilon). \tag{18.10}$$

Eq. (18.10) indicates that the actual lower value $\underline{y}$ will be equal or higher than $\underline{fa}(\mathbf{x})$ and the actual upper value $\overline{y}$ will be equal or lower than $\overline{fa}(\mathbf{x})$.

The conservative rough ε-insensitive loss function, denoted $lc_{r\varepsilon}(fc(\mathbf{x}), y)$, is defined as:

$$lc_{r\varepsilon}(fc(\mathbf{x}), y) \;=\; \underline{lc_{r\varepsilon}}(\underline{fc}(\mathbf{x}), \underline{y}) \;+\; \overline{lc_{r\varepsilon}}(\overline{fc}(\mathbf{x}), \overline{y}), \tag{18.11}$$

where $\underline{lc_{r\varepsilon}}(\underline{fc}(\mathbf{x}), \underline{y})$ is the lower component of the loss defined by:

$$\underline{lc_{r\varepsilon}}(\underline{fc}(\mathbf{x}), \underline{y}) = \begin{cases} \underline{y} - \underline{fc}(\mathbf{x}) & \text{if } \underline{y} \geq \underline{fc}(\mathbf{x}) \\ max(0, \underline{fc}(\mathbf{x}) - \underline{y} - \varepsilon_d) & \text{otherwise} \end{cases}, \tag{18.12}$$

and $\overline{lc_{r\varepsilon}}(\overline{fc}(\mathbf{x}), \overline{y})$ is the upper component of the loss given by:

$$\overline{l_{r\varepsilon}}(\overline{fc}(\mathbf{x}), \overline{y}) = \begin{cases} \overline{fc}(\mathbf{x}) - \overline{y} & \text{if } \overline{fc}(\mathbf{x}) \geq \overline{y} \\ max(0, \overline{y} - \overline{fc}(\mathbf{x}) - \varepsilon_u) & \text{otherwise} \end{cases}. \tag{18.13}$$

The aggressive rough ε-insensitive loss function, denoted $la_{r\varepsilon}(fa(\mathbf{x}), y)$, is now defined as:

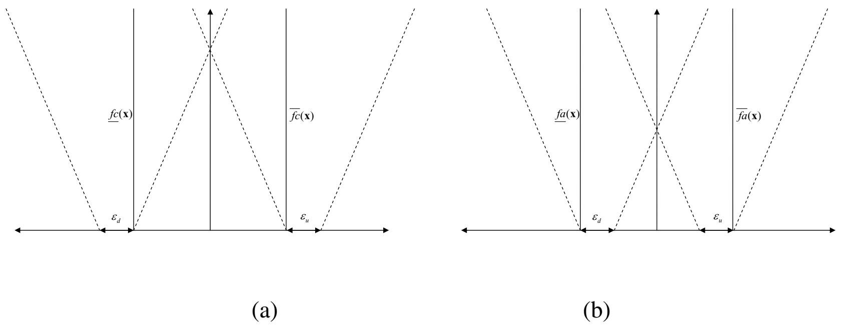
$$la_{r\varepsilon}(fa(\mathbf{x}), y) \;=\; \underline{la_{r\varepsilon}}(\underline{fa}(\mathbf{x}), \underline{y}) \;+\; \overline{la_{r\varepsilon}}(\overline{fa}(\mathbf{x}), \overline{y}), \tag{18.14}$$

where $\underline{la_{r\varepsilon}}(\underline{fa}(\mathbf{x}),\underline{y})$ is the lower component of the loss given by:

$$\underline{la_{r\varepsilon}}(\underline{fa}(\mathbf{x}),\underline{y}) = \begin{cases} max(0,\underline{fa}(\mathbf{x}) - \underline{y} - \varepsilon_d) & \text{if } \underline{y} \leq \underline{fa}(\mathbf{x}) \\ \underline{y} - \underline{fa}(\mathbf{x}) & \text{otherwise} \end{cases}, \tag{18.15}$$

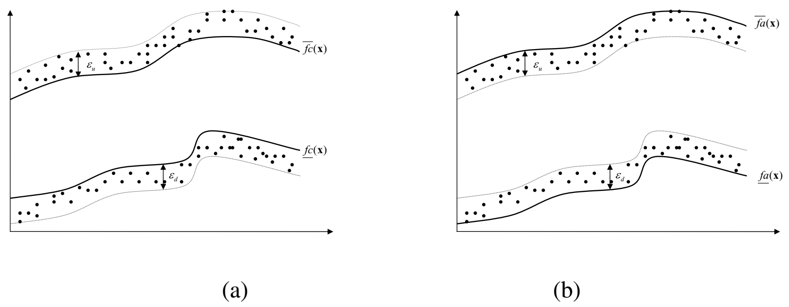
and $\overline{la_{r\varepsilon}}(\overline{fa}(\mathbf{x}),\overline{y})$ is the upper component of the loss defined as:

$$\overline{l_{r\varepsilon}}(\overline{fa}(\mathbf{x}),\overline{y}) = \begin{cases} max(0,\overline{y} - \overline{fa}(\mathbf{x}) - \varepsilon_u) & \text{if } \overline{fa}(\mathbf{x}) \leq \overline{y} \\ \overline{fa}(\mathbf{x}) - \overline{y} & \text{otherwise} \end{cases}. \tag{18.16}$$



$\underline{fc}(\mathbf{x})$
$\overline{fc}(\mathbf{x})$
ε_d
ε_u
(a)
$\underline{fa}(\mathbf{x})$
$\overline{fa}(\mathbf{x})$
ε_d
ε_u
(b)

Fig. 18.7. Conservative and aggressive rough ε-insensitive loss functions



$\overline{fc}(\mathbf{x})$
ε_u
$\underline{fc}(\mathbf{x})$
ε_d
(a)
$\overline{fa}(\mathbf{x})$
ε_u
$\underline{fa}(\mathbf{x})$
ε_d
(b)

Fig. 18.8. Conservative and aggressive modeling with dual rough ε-SVR

The conservative rough ε-insensitive loss function is shown in Fig. 18.7(a), which illustrates that we need to have an ε margin only on the outer side of the lower and upper prediction functions. The aggressive rough ε-insensitive loss function is illustrated in Fig. 18.7(b). As can be seen, we need only to have an ε margin on the inner side of the lower and upper prediction functions.

The conservative modeling of rough patterns is depicted in Fig. 18.8(a). As can be seen, the predicted rough pattern will be inside the actual rough values of the variables as suggested by Eq. (18.9). Fig. 18.8(b) depicts the aggressive modeling of rough patterns. As suggested by Eq. (18.10), it can be observed that the predicted rough pattern will be outside the actual rough values of the variables.

The objective of conservative modeling is to construct a $fc(\mathbf{x})$ such that $lc_{r\varepsilon}$ is minimized. Aggressive modeling, on the other hand, attempts to minimize $la_{r\varepsilon}$. Lingras and Butz [14] describe the optimization process for both of these modeling approaches.

18.3.3 Empirical Analysis of Dual RSVR

The study data used in Lingras and Butz's [14] experiments consisted of the daily high and low values of the DJIA from May 21st to May 20th, 2007. There were a total of 250 trading days during the study period.

Conservative and aggressive RSVR described in the previous subsection were applied to model the DJIA rough pattern. The input vector consisted of the previous ten days high's and low's. Therefore, the model was applied to $250 - 10 = 240$ days. The output was the rough value for the next day.

Lingras and Butz [14] experimented with linear and polynomial Kernels and different values of $\varepsilon = 150, 75, 50$. The results seemed to significantly improve when ε was reduced from 150 to 75. The performance gain was not obvious when ε was further reduced to 50.

Error distribution for the two models is shown in Fig. 18.9. The error is calculated as $actual - predicted$. That means negative errors correspond to over-prediction and positive errors correspond to under-prediction. Fig. 18.9 (a) shows the frequency of errors using three types of bars for conservative modeling. The hollow bars for the lower prediction means that the actual value is less than 75 points (the value of ε_d) below the predicted value for the conservative model, and hence is acceptable according to the loss function $lc_{r\varepsilon}$. The striped bars for lower predictions mean that the values are over-predicted leading to a lower loss (because ε_d will be deducted for over-predictions of lower values). The solid bars indicate under-prediction of lower values. The reverse is true for upper predictions, that is, solid bars indicate over-prediction, striped bars are under-predictions leading to lower loss (ε_u will be deducted for under prediction of upper values), and hollow bars are under-predictions by less than 75 points (the value of ε_u) leading to zero loss. Based on Eq. (18.9) and the loss function $lc_{r\varepsilon}$, hollow bars are the most desirable, followed by striped bars, while the solid bars are least desirable.

Fig. 18.9 (b) also shows the frequency of errors using three types of bars for aggressive modeling. However, the meaning of the bars for the aggressive modeling is a mirror image of that for the conservative modeling. The hollow bars for the

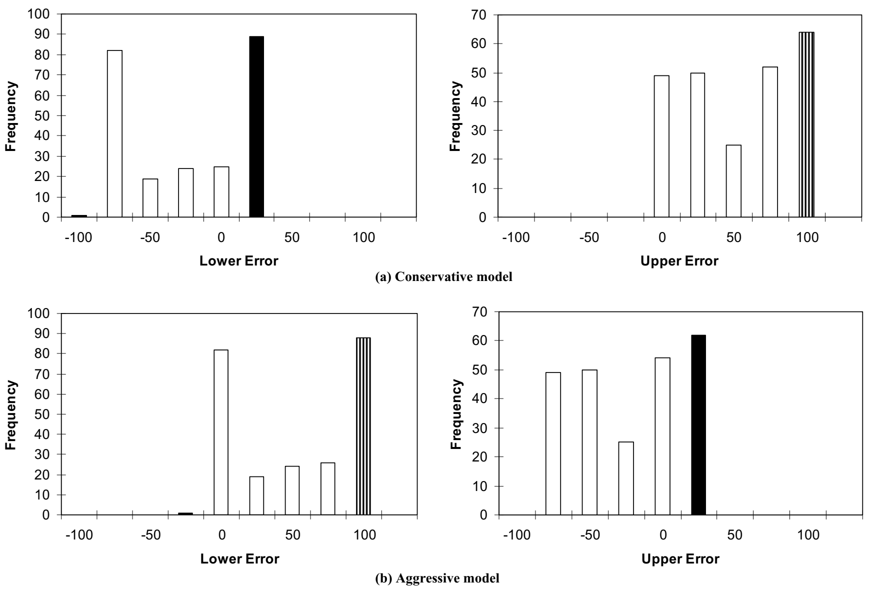


Fig. 18.9. Error distribution for conservative (top) and aggressive (bottom) rough ε-SVR modeling of rough DJIA pattern with $\varepsilon = 75$

lower prediction means that the actual value is less than 75 points (the value of ε_d) above the predicted value for the conservative model, and hence is acceptable according to the loss function $la_{r\varepsilon}$. The striped bars for lower predictions mean that the values are under-predicted leading to a lower loss (because ε_d will be deducted for under-predictions of lower values). The solid bars indicate over-prediction of lower values. The opposite is true for upper predictions, that is, solid bars indicate under-prediction, striped bars are over-predictions leading to lower loss (as ε_u will be deducted for over prediction of upper values), and hollow bars are over-predictions by less than 75 points (the value of ε_u) leading to zero loss. Similar to conservative modeling, based on Eq. (18.10) and the loss function $la_{r\varepsilon}$, the hollow bars are the most desirable, followed by striped bars, leaving solid bars serving as least desirable.

The abundance of hollow and striped bars, for both conservative and aggressive models, means that both approaches performed as expected. The errors for conservative modeling are on the outer sides of 0 (negative for lower values and positive for upper values), while they are on the inner side of 0 (positive for lower values and negative for upper values) for aggressive modeling. This observation clearly underscores the difference between the two philosphies. One can also notice a similarity between Fig. 18.9 (a) and the conservative loss function $lc_{r\varepsilon}$ given in Fig. 18.7(a). Similar correspondence can be drawn between Fig. 18.9 (b) and the aggressive loss function $la_{r\varepsilon}$ given in Fig. 18.7(b).

18.4 Rough Support Vector Clustering

Support vector clustering[2] is a clustering method that uses “kernel trick” [4]. Here, the computation in a high dimensional feature space is achieved using a Kernel function without explicitly mapping data points to the high dimensional feature space. In SVC, we look for the smallest sphere in a feature space that encloses the image of the data such as shown in Fig. 18.10. If this sphere is mapped back to data space, it forms a set of contours that enclose the data points, such as shown in Fig. 18.11. These contours are interpreted as cluster boundaries. Points enclosed by each separate contour are associated with the same cluster. The kernel parameters can control the number of clusters. Here, the outliers are handled with the help of a soft margin formulation.

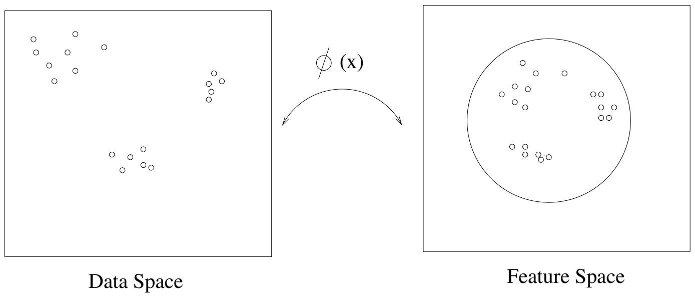


Fig. 18.10. Support Vector Clustering: data space to feature space mapping. Here, ϕ is the implicit non-linear transformation achieved by the kernel function.

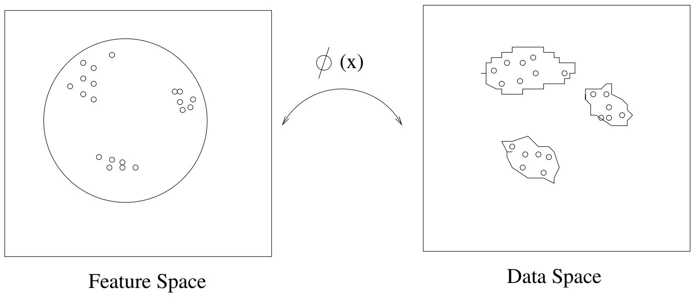


Fig. 18.11. Support Vector Clustering: the data space contours (clusters) obtained by the reverse mapping of the feature space sphere. Here, ϕ is the implicit non-linear transformation achieved by the kernel function.

To define the formulation, let $\{\mathbf{u}_i\} \subseteq U$ be an m-dimensional data set having n points, with $\mathbf{u}_i \in R^m$ being the data space. Now using a nonlinear transformation ϕ from U to some high dimensional feature space, we look for the smallest sphere of radius R enclosing all the points in U. Now the primal problem can be stated as:

$$\min_{R,\xi_i} R^2 + C\sum_{i=1}^{n}\xi_i$$

$$s.t.\ \ \|\phi(\mathbf{u}_i) - \mu\|^2 \le R^2 + \xi_i, \qquad \xi_i \ge 0\ \forall i \tag{18.17}$$

Here, $C\sum_{i=1}^{n}\xi_i$ is the penalty term for the patterns with distance from the center of the sphere in feature space being greater than R (patterns that lie outside the feature space sphere), μ is the center of the sphere in the high dimensional feature space, and $\|.\|$ is the L_2 norm.

Since this is a Convex Quadratic Programming problem, it is easy to solve its Wolfe Dual [6] form. The dual formulation is:

$$\min_{\alpha_i} \sum_{i,j=1}^{n} \alpha_i\alpha_j K(\mathbf{u}_i,\mathbf{u}_j) - \sum_{i=1}^{n}\alpha_i K(\mathbf{u}_i,\mathbf{u}_i)$$

$$s.t.\ \ 0 \le \alpha_i \le C \quad for\ \ i = 1....n, \qquad \sum_{i=1}^{n}\alpha_i = 1 \tag{18.18}$$

Here, $K(\mathbf{u}_i,\mathbf{u}_j)$ represents the Kernel function giving the dot product $\phi(\mathbf{u}_i)\cdot\phi(\mathbf{u}_j)$ in the high dimensional feature space, and α_is are the Lagrangian multipliers.

The value of α_i decides whether a point $\phi(\mathbf{u}_i)$ is inside, outside, or on the sphere. The points with $0 < \alpha_i < C$ form the support vectors. Hence the radius of the sphere enclosing the image of the data points is given by: $R = G(\mathbf{u}_i)$, where $0 < \alpha_i < C$, and where

$$\begin{aligned} G^2(\mathbf{u}_i) &= \|\phi(\mathbf{u}_i) - \mu\|^2 \\ &= K(\mathbf{u}_i,\mathbf{u}_i) - 2\sum_{j=1}^{n}\alpha_j K(\mathbf{u}_j,\mathbf{u}_i) + \sum_{j,k=1}^{n}\alpha_j\alpha_k K(\mathbf{u}_j,\mathbf{u}_k). \end{aligned} \tag{18.19}$$

Now the contours that enclose the points in data space are defined by: $\{\mathbf{u} : G(\mathbf{u}) = R\}$. Thus, the computation in high dimensional feature space and also reverse mapping to find the contours in data space are avoided with the help of Kernel function. Once these contours are found, the objects are assigned to different clusters. The cluster assignments employ a geometric method involving $G(\mathbf{u})$, based on the observation that given a pair of points that belong to different clusters, any path that connects them must exit from the sphere in the feature space. So we can define an adjacency matrix M by considering all pairs of points $\mathbf{u}_i$ and $\mathbf{u}_j$ whose images lie in or on the sphere in the feature space and then looking at the image of the path that connects them as:

$$M[i,j] = \begin{cases} 1 \text{ if } G(y) \leq R & \forall y \in [\mathbf{u}_i, \mathbf{u}_j] \\ 0 \text{ otherwise} \end{cases} . \qquad (18.20)$$

Clusters are now defined as the connected components of the graph induced by M. The points that lie outside the sphere, known as bounded support vectors, can be assigned to the closest clusters.

Rough Support Vector Clustering (RSVC)[1, 29] is an extension of the SVC paradigm that employs rough set theory to achieve soft clustering. To discuss the method formally, let us use the notion of rough sets to introduce a Rough Sphere. A *Rough Sphere* is defined as a sphere having an inner radius R defining its lower approximation and an outer radius $T > R$ defining its upper approximation. As in SVC, RSVC also uses a kernel function to achieve computation in a high dimensional feature space. It tries to find the smallest rough sphere in the high dimensional feature space enclosing the images of all the points in the data_set. Now those points whose images lie within the lower approximation ($\underline{A}(\cdot)$) are points that definitely belong to exactly one cluster (called the hard core of a cluster) and those points whose images lie in the boundary region ($\overline{A}(\cdot) - \underline{A}(\cdot)$, that is, in the upper approximation $\overline{A}(\cdot)$ but not in the lower approximation $\underline{A}(\cdot)$) may be shared by more than one cluster (called the soft core of the cluster). Some points are permitted to lie outside the sphere and are termed outliers. By using a nonlinear transformation ϕ from data space to some high dimensional feature space, we seek the smallest enclosing rough sphere of inner radius R and outer radius T. Now the primal problem can be stated formally as:

$$\min_{R,T,\xi_i} \; R^2 + T^2 + \frac{1}{\upsilon n}\sum_{i=1}^{n}\xi_i + \frac{\delta}{\upsilon n}\sum_{i=1}^{n}\xi_i'$$

$$s.t \;\; ||\phi(\mathbf{u}_i) - \mu||^2 \leq R^2 + \xi_i + \xi_i' \quad 0 \leq \xi_i \leq T^2 - R^2 \quad \xi_i' \geq 0 \;\; \forall i \qquad (18.21)$$

Here, $\frac{1}{\upsilon n}\sum_{i=1}^{n}\xi_i$ is a penalty term for the patterns with distance from the center of the sphere in feature space being greater than R (patterns falling in the boundary region) and $\frac{\delta}{\upsilon m}\sum_{i=1}^{n}\xi'$ is a penalty term associated with the patterns whose distance from the center of the sphere in feature space is greater than T (patterns falling outside the rough sphere).

Since this is a Convex Quadratic Programming problem it is easy to write its Wolfe Dual. The Lagrangian can be written as:

$$L = R^2 + T^2 + \frac{1}{\upsilon n}\sum_{i=1}^{n}\xi_i + \frac{\delta}{\upsilon n}\sum_{i=1}^{n}\xi_i' + \sum_{i=1}^{n}\alpha_i(||\phi(\mathbf{u}_i) - \mu||^2 - R^2 - \xi_i - \xi_i')$$

$$-\sum_{i=1}^{n}\beta_i\xi_i + \sum_{i=1}^{n}\lambda_i(\xi_i - T^2 + R^2) - \sum_{i=1}^{n}\eta_i\xi_i', \tag{18.22}$$

where the Lagrange multipliers $\alpha_i, \beta_i, \lambda_i$ and η_i are non-negative $\forall i$. Using the Karush-Kuhn-Tucker (KKT) [6] conditions on Eq. (18.22), we obtain:

$$\sum_{i=1}^{n}\alpha_i = 2 \qquad \mu = \frac{1}{2}\sum_{i=1}^{n}\alpha_i\phi(\mathbf{u}_i)$$

$$\beta_i - \lambda_i = \frac{1}{\upsilon n} - \alpha_i \qquad \frac{\delta}{\upsilon n} - \alpha_i = \eta_i$$

$$\alpha_i(\|\phi(\mathbf{u}_i) - \mu\|^2 - R^2 - \xi_i - \xi_i') = 0$$

$$\lambda_i(\xi_i - T^2 + R^2) = 0$$

$$\beta_i\xi_i = 0 \qquad \eta_i\xi_i' = 0$$

From the above equations the Wolfe Dual form can be written as:

$$\min_{\alpha_i} \sum_{i,j=1}^{n}\alpha_i\alpha_j K(\mathbf{u}_i, \mathbf{u}_j) - \sum_{i=1}^{n}\alpha_i K(\mathbf{u}_i, \mathbf{u}_i)$$

$$s.t \quad 0 \le \alpha_i \le \frac{\delta}{\upsilon n} \quad for\ i = 1....n, \qquad \sum_{i=1}^{n}\alpha_i = 2. \tag{18.23}$$

Here, $K(\mathbf{u}_i, \mathbf{u}_j)$ represents the Kernel function giving the dot product $\phi(\mathbf{u}_i).\phi(\mathbf{u}_j)$ in the high dimensional feature space [2].

If $\delta > 1$, we obtain RSVC. The formulation reduces to the original SVC for $\delta = 1$. Also the values of α_i decide whether the pattern $\mathbf{u}_i$ falls in the lower approximation, the boundary region, or outside the feature space rough sphere. From KKT conditions on Eq. (18.22), it can be observed that image of points with

- $\alpha_i = 0$ lie in the lower approximation.
- $0 < \alpha_i < \frac{1}{\upsilon n}$ form the hard support vectors (support vectors marking the boundary of the lower approximation).
- $\alpha_i = \frac{1}{\upsilon n}$ lie in the boundary region (patterns that may be shared by more than one cluster).
- $\frac{1}{\upsilon n} < \alpha_i < \frac{\delta}{\upsilon n}$ form the soft support vectors (support vectors marking the boundary of the upper approximation).
- $\alpha_i = \frac{\delta}{\upsilon n}$ lie outside the rough sphere (bounded support vectors).

18.4.0.1 Cluster Assignment

Once the dual problem is solved to find the α_i values, the clusters can be obtained using the following strategy. Let us define:

$$R = G(\mathbf{u}_i) \ : \ 0 < \alpha_i < \frac{1}{\upsilon n}$$

$$T = G(\mathbf{u}_i) \ : \ \frac{1}{\upsilon n} < \alpha_i < \frac{\delta}{\upsilon n}, \tag{18.24}$$

where $G(\mathbf{u}_i)$ is defined in Eq. (18.19).

From the above equations we can define the contours that enclose the lower approximation of clusters in data space as: $\{\mathbf{u} : G(\mathbf{u}) = R\}$ and the contours that enclose the upper approximation of clusters in data space as: $\{\mathbf{u} : G(\mathbf{u}) = T\}$. Now the soft clusters in data space are found using a strategy similar to the one used in SVC. The *algorithm to find clusters* can now be given as follows.

- As in SVC, find the adjacency matrix M by considering all pairs of points $\mathbf{u}_i$ and $\mathbf{u}_j$ whose images in feature space either belong to the lower approximation of the rough sphere or are hard support vectors and then looking at the image of the path that connects them as

$$M[i,j] = \begin{cases} 1 \text{ if } G(y) \leq R & \forall y \in [\mathbf{u}_i, \mathbf{u}_j] \\ 0 \text{ otherwise} \end{cases}$$

- Find connected components for the graph represented by M. Each connected component found gives the lower approximation of a cluster $\mathbf{x_i}$.
- Find the boundary regions as:
  $\mathbf{u}_i \in \underline{A}(\mathbf{x_i})$ and pattern $\mathbf{u}_k \notin \underline{A}(\mathbf{x_j})$ for any cluster j,
  if $G(y) \leq T \ \forall y \in [\mathbf{u}_i, \mathbf{u}_k]$ then $\mathbf{u}_k \in (\overline{A}(\mathbf{x_i}) - \underline{\mathbf{A}}(\mathbf{x_i}))$

18.4.0.2 Role of υ and δ

From Eq. (18.23) it can be seen that the number of bounded support vectors is $n_{bsv} < 2\frac{\upsilon n}{\delta}$. For $\delta = 1$, $\quad n_{bsv} < 2\upsilon n = \upsilon' n \quad$ where $\upsilon' = 2\upsilon$. This corresponds to all the patterns $\mathbf{u}_i$ with $||\phi(\mathbf{u}_i) - \mu||^2 > R^2$. Since $\delta > 1$ for RSVC, we can say that $\frac{\upsilon'}{\delta}$ is the upper bound on the fraction of points permitted to lie outside T, and υ' is the upper bound on the fraction of points permitted to lie outside R. Hence, υ and δ together give us control over the width of boundary region and the number of bounded support vectors. Therefore, we can choose the values of υ and δ based on the percentage of the data we want to put in the soft core of the clusters, and what percentage we want to treat as outliers.

18.4.1 Experimental Results with RSVC

Experiments were done with a synthetically generated data set and a real world data set viz; Wine recognition data set [1], availabel at UCI Machine Learning repository from School of Information and Computer Sciences, University of California, Irvine.

18.4.1.1 Synthetic Data Set

The synthetic data is generated by sampling from four normal distributions with different mean values and uniform variance values. Thus the data set obtained get distributed in four clusters with some possible overlap between clusters. This synthetic data set and the clustering results obtained with $\upsilon = 0.25$ and $\delta = 1.25$ for RSVC are shown in Figure 18.12. From the clustering results, it may be observed that RSVC identified four clusters with some of the data points - shown by astericks - falls in the boundary region between the clusters as expected and some others does seem to belong to the upper approximation of any one cluster.

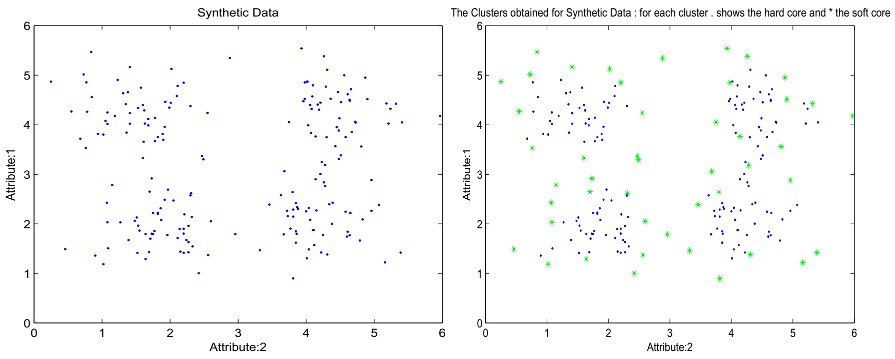


Fig. 18.12. Synthetic Data Set and RSVC result

18.4.1.2 Wine Recognition Data Set

The Wine recognition data set [1] is the results of a chemical analysis of three types (3 classes) of wines grown in the same region in Italy but derived from three different cultivators. This data is described by thirteen measurements (13 attributes) each representing the quantity of a constituent found in the wines. The data set consists of 59 instances of Classs 1 type of wine, 71 instances of Class 2 type of wine and 48 instances of Class 3 type of wine yielding a total of 178 patterns. Here we have applied principal component analysis to reduce the dimensionality of the data set to two. The data set thus obtained and the result of applying RSVC with $\upsilon = 0.29$ and $\delta = 3.5$ on this data set are shown in Figure 18.13. As expected, the RSVC algorithm created three clusters roughly identifying the existing cluster sizes. Closer examination of Figure 18.13 reveals that there are a few objects - indicated by asterisks - that belong to the boundary region between these three clusters.

From the clustering results obtained from RSVC, it may be observed that there are some data points that belong to the upper approximation of only one cluster, but not shared by more than one clusters as demanded by the rough set theory [18]. Therefore, one can regard the clusters created by RSVC more as interval sets than as rough sets.

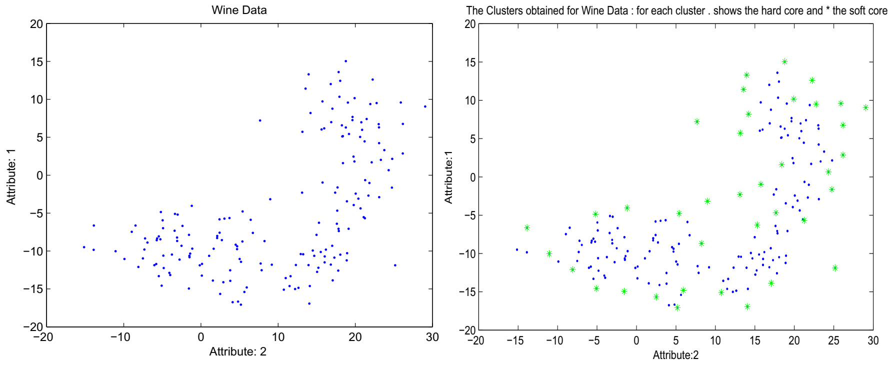


Fig. 18.13. Wine Data Set and RSVC result

18.5 Summary and Conclusions

There is a natural relationship between support vector techniques and the rough set theory. This chapter describes rough set theoretic extensions of SVM classification, regression, and clustering. The concepts of margin in SVM classification corresponds to the boundary regions in rough set theory. Use of boundary region is especially useful for soft margin SVM classifiers. Use of rough set view is shown to improve the performance of multi-classifications using both the 1-v-1 and 1-v-r approaches.

The ε-loss functions in support vector regression make it possible to create a band of lower and upper values for a function. The dual rough support vector regression described in this chapter modify the ε-loss functions using conservative and aggressive philosophies to model rough patterns in a financial time series.

Finally, the extension of support vector clustering using rough set theory allows us to create rough set representation of clusters with irregular boundaries, as opposed to traditional spherical surfaces in the conventional rough set clustering algorithms.

References

1. Asharaf, S., Shevade, S.K., Murty, N.M.: Rough support vector clustering. Pattern Recognition 38(10), 1779–1783 (2005)
2. Ben-Hur, A., Horn, D., Siegelmann, H.T., Vapnik, V.N.: Support Vector Clustering. Journal of Machine Learning Research 2(2), 125–137 (2001)
3. Boser, B.E., Guyon, I., Vapnik, V.N.: A training algorithm for optimal margin classifiers. In: Proc. 5th Annual Workshop Computational Learning Theory, pp. 144–152. ACM, New York (1992)
4. Burges, C.: A tutorial on support vector machines for pattern recognition. Data Mining and Knowledge Discovery 2(2), 121–167 (1998)
5. Cristianini, N.: Support vector and kernel methods for pattern recognition (2003), `http://www.support-vector.net/tutorial.html`
6. Fletcher, R.: Practical Methods of Optimization, 2nd edn. Wiley-Interscience, New York (2000)
7. Hoffmann, A.: Vc learning theory and support vector machines (2003), `http://www.cse.unsw.edu.au/~cs9444/Notes02/Achim-Week11.pdf`
8. Knerr, S., Personnaz, L., Dreyfus, G.: Single-layer learning revisited: A stepwise procedure for building and training a neural network. In: Neurocomputing: Algorithms, Architectures and Applications. Springer, Heidelberg (1990)
9. Lingras, P.: Applications of rough patterns. In: Rough Sets in Data Mining and Knowledge Discovery, vol. 2, pp. 369–384. Springer (1998)
10. Lingras, P.: Fuzzy-rough and rough-fuzzy serial combinations in neurocomputing. Neurocomputing Journal 36, 29–44 (2001)
11. Lingras, P., Butz, C.J.: Interval set classifiers using support vector machines. In: Proc. the North American Fuzzy Inform Processing Socety Conference, pp. 707–710. Computer Society Press, Washington, DC (2004)
12. Lingras, P., Butz, C.J.: Rough set based 1-v-1 and 1-v-r approaches to support vector machine multi-classification. Information Sciences 177, 3298–3782 (2007)
13. Lingras, P., Butz, C.J.: Rough support vector regression. European Journal of Operational Research 206, 445–455 (2010)
14. Lingras, P., Butz, C.J.: Conservative and aggressive rough svr modeling. Theoretical Computer Science Journal Section on Theory of Natural Computing 412(42), 5885–5901(2011)
15. Lingras, P., Davies, C.: Applications of rough genetic algorithms. Computational Intelligence: An International Journal 17(3), 435–445 (2001)
16. Mukherjee, S., Osuna, E., Girosi, F.: Nonlinear prediction of chaotic time series using support vector machines. In: Principe, N.M.J., Giles, L., Wilson, E. (eds.) IEEE Workshop on Neural Networks for Signal Process, vol. VII, pp. 511–519. IEEE Computer Society Press, Washington, DC (1997)
17. Muller, K.R., Smola, A., Ratsch, G., Schölkopf, B., Kohlmorgen, J., Vapnik, V.: Predicting time series with support vector machines. In: ICANN, pp. 999–1004 (1997)
18. Pawlak, Z.: Rough Sets. International Journal of Computer and Information Sciences 11, 341–356 (1982)
19. Pawlak, Z.: Rough real functions (1994), `http://citeseer.ist.psu.edu/105864.html`
20. Peters, J.F., Han, L., Ramanna, S.: Rough neural computing in signal analysis. Computational Intelligence 17(3), 493–513 (2001)
21. Platt, J.C.: Support vector machines (2003), `http://research.microsoft.com/users/jplatt/svm.html`

22. Platt, J.C., Cristianini, N., Shawe-Taylor, J.: Large margin dag's for multiclass classification. In: Advances in Neural Inform Processing Systems, pp. 547–553. MIT Press (2000)
23. da Rocha, A., Yager, R.: Neural nets and fuzzy logic. In: Kandel, A., Langholz, G. (eds.) Hybrid Architectures for Intelligent Systems, pp. 3–28. CRC Press, Ann Arbor (1992)
24. Smola, A., Scolkopf, B.: A tutorial on support vector regression. In: NeuroCOLT2 (1998)
25. Vapnik, V.: Estimation of Dependencies Based on Empirical Data. Nauka, Moscow (1979)
26. Vapnik, V.: The Nature of Statistical Learning Theory. Springer, NY (1995)
27. Vapnik, V.: Statistical Learning Theory. Wiley, NY (1998)
28. Vapnik, V.N., Golowich, S., Smola, A.: Support vector method for function approximation, regression estimation and signal processing. In: Advances in Neural Information Processing Systems, vol. 9, pp. 281–287 (1997)
29. Varma, C.S., Asharaf, S., Murty, M.N.: Rough Core Vector Clustering. In: Ghosh, A., De, R.K., Pal, S.K. (eds.) PReMI 2007. LNCS, vol. 4815, pp. 304–310. Springer, Heidelberg (2007)
30. Yang, H.: Margin Variations in Support Vector Regression for the Stock Market Prediction. M. Phil. dissertation, Department of Computer Science and Engineering, The Chinese University of Hong Kong (2003), `http://www.svms.org/finance/Yang2003.pdf`
31. Yang, H., Chan, L.-W., King, I.: Support Vector Machine Regression for Volatile Stock Market Prediction. In: Yin, H., Allinson, N., Freeman, R., Keane, J., Hubbard, S. (eds.) IDEAL 2002. LNCS, vol. 2412, pp. 391–396. Springer, Heidelberg (2002)