Data dimensionality reduction with application to improving classification performance and explaining concepts of data sets

Xiuju Fu

Institute of High Performance Computing,
Science Park 2, 117528, Singapore
E-mail: fuxj@ihpc.a-star.edu.sg

Lipo Wang*

School of Electrical and Electronic Engineering,
Nanyang Technological University,
Block S1, Nanyang Avenue, 639798, Singapore
E-mail: elpwang@ntu.edu.sg
http://www.ntu.edu.sg/home/elpwang
*Corresponding author

Abstract: Data dimensionality reduction is usually carried out before patterns are input to classifiers. In order to obtain good results in data mining, selecting relevant data is desirable. In many cases, irrelevant or redundant attributes are included in data sets, which interfere with knowledge discovery from data sets. In this paper, we propose a rule-extraction method based on a novel separability-correlation measure (SCM) ranking the importance of attributes. According to the attribute ranking results, the attribute subsets that lead to the best classification results are selected and used as inputs to a classifier, such as an RBF neural network in our paper. The complexity of the classifier can thus be reduced and its classification performance improved. Our method uses the classification results with reduced attribute sets to extract rules. Computer simulations show that our method leads to smaller rule sets with higher accuracies compared with other methods.

Keywords: RBF; rule extraction; classification; neural networks; data mining; knowledge discovery.

Reference to this paper should be made as follows: Fu, X. and Wang, L. (2005) 'Data dimensionality reduction with application to improving classification performance and explaining concepts of data sets', *Int. J. Business Intelligence and Data Mining*, Vol. 1, No. 1, pp.95–117.

Biographical notes: Xiuju Fu received the BS degree and the MS degree in 1995 and 1999, respectively, both from the Beijing Institute of Technology (China). She received the PhD degree from the Nanyang Technological University (Singapore). Currently, she is a research fellow in the Institute of High Performance Computing, Singapore. She has co-authored more than ten papers in conference proceedings and journals. She received the Best Student Paper Award in the 2000 Augsburg Symposium on Data Mining and Statistics. Her current research areas include neural networks, support vector machines, genetic algorithms, data mining, classification, data dimensionality reduction, and rule extraction.

Lipo Wang holds a US patent and has published over 50 journal papers, 15 books, and 70 conference presentations in computational intelligence. He is Associate Editor for six international journals. He chairs the Emergent Technologies Technical Committee, IEEE Computational Intelligence Society. He is founding Chair of both IEEE Engineering in Medicine and Biology Chapter Singapore and IEEE Computational Intelligence Chapter Singapore. He is Past President of Asia-Pacific Neural Network Assembly, Program Chair for 2006 International Joint Conference on Neural Networks, and Program Co-Chair for 2007 IEEE Congress on Evolutionary Computation.

1 Introduction

Data dimensionality reduction (DDR) aims at removing irrelevant attributes while maintaining the information of data sets. DDR has become an important aspect of data mining since human experts and corporate managers are able to make better use of lower-dimensional data compared to higher-dimensional ones. In addition, DDR can decrease the computational burden for various automated processes. for example, when a radial basis function (RBF) neural network is used to classify data. Reduced data dimensionality leads to less complicated network structure and thus increases efficiency in processing data. Removal of irrelevant data can significantly improve the accuracy of a classifier. In addition, with a fewer number of attributes obtained by DDR techniques, concise rules with higher accuracies can be obtained in rule-extraction tasks.

DDR techniques may be classified into feature (attribute) extraction and feature selection. Feature extraction creates a number of new features through a transformation of the raw features. Linear discriminant analysis (LDA) (Bollacker and Ghosh, 1996; Kawatani and Shimizu, 1998; Liu and Wechsler, 1998) and principal component analysis (PCA) (Kambhatla and Leen, 1993) are two popular techniques for feature extraction. Although these types of transformations are designed to maintain concepts in the data, it is difficult to prevent by-products from affecting the original concepts in the data detrimentally.

In feature selection techniques, the attributes that are important for maintaining the concepts of the original data are selected from the entire attribute set. No new features or unwanted by-products are generated with feature selection. How to determine the importance level of attributes is the key to feature selection techniques. The idea of ranking feature importance for selecting features out of the original feature set derives from the fact that different features contribute to classification differently, i.e., irrelevant features degrade the performance of classification both in the classification accuracy and the complexity of representing classification results.

Mutual information based feature selection (MIFS) (Bollacker and Ghosh, 1996; Battiti, 1994) is a common method of feature selection, in which 'the information content' of each attribute (feature) is evaluated with regard to class labels and other attributes. By calculating mutual information, the importance levels of features are ranked based on their ability to maximise the evaluation formula. However, in the MIFS, the number of features to be selected needs to be pre-defined. Dash et al. (1997) use an entropy measure (SUD) to evaluate the relative importance of attributes. The entropy measure is based on the similarities of different instances without considering class

labels. Attribute subsets were then selected based on attribute importance ranking and classification accuracy with C4.5.

Kononenko (1994) proposed Relief-F to rank attribute importance, in which M nearest neighbours are searched from M different classes for a given instance. A pair of instances is formed including the given instance and one of its nearest neighbours. Together with the nearest neighbour from the same class, there are $M+1$ pairs of instances. The difference in an attribute in each pair of instances is calculated. The probabilities of these differences are used to evaluate the importance of the attribute.

Rule extraction is an important task in data mining. Without rule extraction, trained neural networks are sometimes criticised as black boxes, since they do not offer explicit explanations as to how the trained networks work. Usually, a rule consists of an IF part and a THEN part. The premise parts of rules are combinations of attributes. There are three kinds of rule decision boundaries, i.e., hyper-rectangular, hyper-plane, and hyper-ellipse. Due to its explicit form and perceptibility, hyper-rectangular decision boundaries are often employed in rule extraction, such as rules extracted from the MLPs (Bologna and Pellegrini, 1998; Ishibuchi and Nii, 1996; Lu et al., 1996) and from RBF neural networks (McGarry et al., 1999; McGarry and MacIntyre, 1999). In order to obtain symbolic rules with hyper-rectangular decision boundaries, a special *interpretable* MLP (IMLP) was constructed in Bologna and Pellegrini (1998). In an IMLP network, each hidden neuron receives a connection from only one input unit, and the activation function used for the first hidden layer neurons is the threshold function. In Lu et al. (1996), the range of each input attribute was divided into intervals. The attribute was then encoded as a binary string accordingly. Rules with hyper-rectangular decision boundaries were thus obtained. Ishibuchi and Nii (1996) extracted fuzzy IF-THEN rules. To determine the threshold function, sub-intervals, and membership functions, prior knowledge on how to divide the ranges of the attributes is desirable. Unsuitable division of attribute ranges leads to low rule accuracy. The division will then have to be adjusted, and the training procedure and the rule-extraction procedure will have to be repeated.

In this paper, we first introduce a separability-correlation measure (SCM) for determining the importance of the original attributes (Section 2) (Fu and Wang, 2003). We describe a simplified RBF classifier by allowing for large overlaps between patterns of the same class (Section 3).

We then propose a rule-extraction method based on DDR (Section 4). In an RBF classifier, the boundary of the receptive field of the kernel function is a hyper-sphere. Rules with hyper-rectangular decision boundaries are extracted based on the training result of the RBF neural network using gradient descent.

Experimental results on reducing data dimensionality, obtaining a simpler architecture of the RBF classifier, and extracting rules are shown in Section 5. Finally, we conclude the paper in Section 6.

2 Separability-correlation measure for feature importance ranking

2.1 A class separability measure

The probability of correct classification is large when the distances between different classes are large. Therefore, the subset of features that can maximise the separability between classes is a desirable objective of feature selection.

Class separability may be measured by the intraclass distance (the distance of patterns within class) S_w and the interclass distance (the distance between patterns of different classes) S_b (Devijver and Kittler, 1982):

$$S_w = \sum_{i=1}^{C} \frac{P_i}{n_i} \sum_{k=1}^{n_i} [(\bar{\bar{X}}_{ik} - \bar{m}_i)(\bar{\bar{X}}_{ik} - \bar{m}_i)^T]^{1/2}, \tag{1}$$

and

$$S_b = \sum_{i=1}^{C} P_i [(\bar{m}_i - \bar{m})(\bar{m}_i - \bar{m})^T]^{1/2}. \tag{2}$$

Here C is the number of classes in the data set. n_i is the number of patterns in the ith class. P_i is the probability of the ith class. $\bar{\bar{X}}_{ik}$ is the normalised data vector, whose jth attribute, $X_{ik}(j)$ is normalised as:

$$\bar{\bar{X}}_{ik} = \frac{X_{ik}(j)}{\mathrm{Max}(x_j) - \mathrm{Min}(x_j)}, \tag{3}$$

where $\mathrm{Max}(x_j)$ and $\mathrm{Min}(x_j)$ are the maximum and minimum of the jth attribute in the data set, respectively. $j = 1, 2, \ldots, n$. n is the number of attributes. $X_{ik}(j)$ is the original (unnormalised) data. $\bar{m}_i$ is the mean vector of the ith class:

$$\bar{m}_i = \frac{\sum_{k=1}^{n_i} \bar{\bar{X}}_{ik}}{n_i}. \tag{4}$$

$\bar{m}$ is the mean of all patterns in the data set:

$$\bar{m}_i = \frac{\sum_{i=1}^{c} \sum_{k=1}^{n_i} \bar{\bar{X}}_{ik}}{n_i}. \tag{5}$$

N is the total number of patterns in the data set, i.e., $N = n_1 + n_2 + \cdots + n_c$.

The greater S_b is and the smaller S_w is, the better the separability of the data set. Therefore, the ratio of S_w *and* S_b can be used to measure the distinction of the classes: the smaller the ratio, the better the separability (Devijver and Kittler, 1982).

If removing attribute k_1 from the data set leads to less class separability, i.e., a greater S_w/S_b, compared to the case where attribute k_2 is removed, one may consider attribute k_1 more important for classification of the data set than attribute k_2 is, and vice versa. Hence, we may rank the importance of the attributes by calculating the intraclass-to-interclass distance ratio with each attribute omitted in turn.

However, the ratio S_w/S_b does not always work well as a class separability measure. For example, consider two classes, with one class surrounding the other, but are completely separable. Since and $\bar{m}_1$, $\bar{m}_2$ and $\bar{m}$ defined in equation (4) and equation (5) are equal, $S_b \rightarrow 0$, which indicates total *inseparability*. Here there is a need to have other importance measures.

2.2 An attribute-class correlation measure

In addition to the separability of classes in the data set, the correlation between the changes in attributes and their corresponding changes in class labels should be taken into account when ranking the importance of attributes. This correlation directly links features with class labels. To two different patterns, if their class labels are different, the variations of attributes in the two patterns are considered to be the affecting factor for the variation of class labels and should be weighted positively; if the class labels are the same, the variations in the attributes are irrelevant in deciding the classes and should be weighted negatively. The correlation measure can be a useful factor by combining together with our class separability measure.

We introduce the following correlation between the kth attribute and the class labels in the data set:

$$C_k = \sum_{i \neq j} \left| \bar{\bar{X}}_{ik} - \bar{\bar{X}}_{jk} \right| \times \text{magn}(y_i - y_i), \tag{6}$$

where $\bar{\bar{X}}_{ik}$ and $\bar{\bar{X}}_{jk}$ are the kth attributes of the ith pattern and the jth pattern, respectively. y_i and y_j are the class labels of the ith pattern and the jth pattern respectively.

For any y, magn(y) = 1 if $|y| > 0$ and magn(y) = –0.05 if $|y| = 0$. A great magnitude of C_k shows that there is a close correlation between class labels and the kth attribute, which indicates the great importance of attribute k in classifying the patterns, and vice versa.

2.3 The separability-correlation measure for attribute importance ranking

We introduce the following separability-correlation measure (SCM) to evaluate the importance levels of attributes by combining the above two measures:

$$R_k = \chi \overline{S_k} + (1 - \chi) \overline{C_k}, \tag{7}$$

where $S_k = S_{wk}/S_{bk}$. $S_k = S_k - \text{Min}(S_k)/\text{Max}(S_k) - \text{Min}(S_k)$ is the normalisation of S_k. Max(S_k) and Min(S_k) are the maximum and minimum of all S_k, respectively. $k = 1, 2, \ldots, n$. n is the number of attributes. S_{wk} and S_{bk} are intraclass and interclass distances calculated with the kth attribute omitted from each pattern, respectively. For example, the ith pattern $\bar{X}_{ik} = \{x_{i1}, x_{i2}, \ldots, x_{ik-1}, x_{ik+1}, \ldots, x_{in}\}$ becomes $\bar{X}_i = \{x_{i1}, x_{i2}, \ldots, x_{ik-1}, x_{ik+1}, \ldots, x_{in}\}$ when R_k is calculated. $\overline{C_k} = C_k - \text{Min}(C_k)/\text{Max}(C_k) - \text{Min}(C_k)$ is the normalisation of C_k. χ is the parameter to weight the two items for the final measure. Here $0 \geq \chi \geq 1$ and χ is determined empirically: the best choice of χ should lead to a subset of attributes, which results in the highest classification accuracy.

The importance levels of attributes are ranked using the values of R_k. The greater the magnitude of R_k, the more important the kth attribute. We will demonstrate the use of our SCM method in Section 4.

We use a combination of two measures, i.e., class separability and attribute-class correlation, because either of them alone does not work well, as shown in our experimental results presented later in the paper.

2.4 *Bottom-up, top-down, and exhaustive search for ranking attributes*

Either bottom-up search or top-down search can be used for ranking attribute importance. In bottom-up search, we begin from an empty set. The SCM is used for evaluating each attribute by omitting this attribute from each pattern, i.e., the attribute is considered to be more important if its corresponding SCM magnitude is larger than others. The selected attribute is included in the empty attribute subset. This operation is continued until n attributes are included. The order of attributes entering the attribute set indicates the importance order of attributes. In bottom-up search, the number of SCM calculations is n (n is the number of attributes).

In top-down searches, we start from the complete attribute set. Each attribute is removed from the attribute set temporarily for calculating its SCM. Then the least important attribute, whose corresponding value in SCM is the smallest, is eliminated from the current attribute set. The steps are iterated until only one attribute is left in the attribute set. The number of SCM calculations for class separability measure is $n + (n-1) + \cdots + 1 = n(n+1)/2$. In Section 4, the differences of the two searches are shown.

In exhaustive search, all the possible attribute subsets are examined. The number of computations needed is $C_n^1 + C_n^2 + \cdots + C_n^{n-1} + C_n^n = 2^n - 1$. In the branch and bound algorithm, some attribute sets are avoided examining, which leads to the saving in computation; however, the number of calculations still grows exponentially with n. In addition, the computational saving is achieved by requesting the feature selection evaluation function being with the monotonicity property (Devijver and Kittler, 1982). The classification accuracy of a neural network classifier is often used as the feature selection evaluation. With the interference of irrelevant attributes for the classification accuracy, the evaluation results from classifiers may not obey the montonicity property.

Due to the computational burden of optimal search methods, one has to resort to suboptimal feature selection methods. In classification tasks, since the goal is to obtain better classification accuracy with less complicated construction of classifiers, the strategy of using the classification accuracy as evaluation for selecting features is used widely. We use suboptimal search and RBF classifiers as evaluators in this paper.

3 A modified method for constructing an efficient RBF classifier

RBF neural networks are widely used for function approximation, pattern classification, and so on. In this paper, we use an RBF classifier together with the SCM to select the best subsets of attributes. In the RBF neural network, the activation of a hidden unit is determined by the distance between the input vector and the centre vector of the hidden unit. The weights connecting the hidden layer and the output layer can be determined by a linear least square (LLS) method (Bishop, 1995), which is fast and free of local minima, in contrast to the multilayer perceptron neural network.

There are three layers in the RBF neural network, i.e., the input layer, the hidden layer with Gaussian activation functions, and the output layer. The architecture of the RBF neural network is shown in Figure 1. In this paper, we use the RBF network for classification. If there are M classes in the data set, we write the mth output of the network as follows:

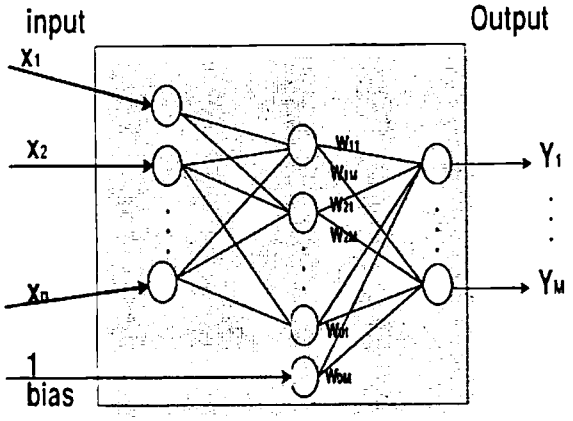
$$y_m(\mathbf{X}) = \sum_{j=1}^{K} w_{mj}\varnothing_j(\mathbf{X}) + w_{m0}b_m. \tag{8}$$

Here **X** is the n-dimensional input pattern vector, $m = 1, 2, \ldots, M$. K is the number of hidden units. M is the number of output. w_{mj} is the weight connecting the jth hidden unit to the mth output node. b_m is the bias. w_{m0} is the weight connecting the bias and the mth output node. $\varnothing_j(\mathbf{X})$ is the activation function of the jth hidden unit:

$$\varnothing_j(\mathbf{X}) = e^{\frac{-\|\mathbf{X}-C_j\|^2}{2\sigma_j^2}}. \tag{9}$$

where C_j and σ_j are the centre and the width for the jth hidden unit, respectively, which are adjusted during learning.

Figure 1 Architecture of an RBF neural network



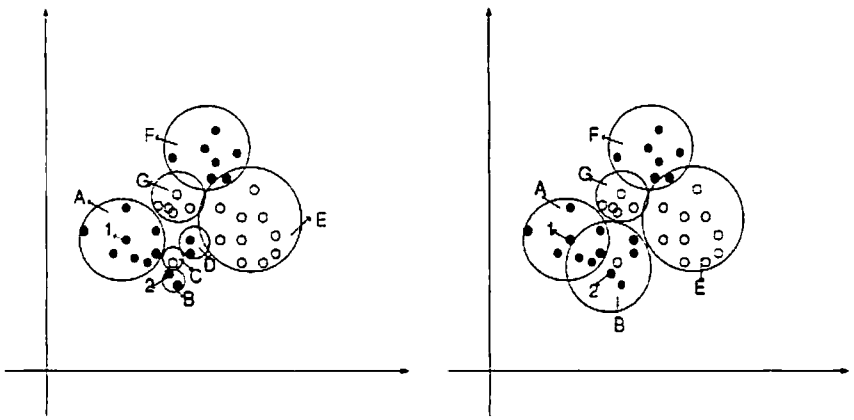
Finding the centres, widths, and the weights connecting hidden nodes to the output is the key for constructing and training the RBF classifier. Both the dimensionality and the distribution of the input patterns affect the number of the hidden units. If the dimensionality is reduced, the number of hidden units will also be decreased.

Overlapped receptive fields of different clusters can improve the performance of the RBF classifier in rejecting noise when dealing with noisy data (Maffezzoni and Gubian, 1994). In Roy et al. (1995) and Kaylani and Dasgupta (1994), overlapping Gaussian kernel functions are created to map out the territory of each cluster with a smaller number of Gaussians. In those previous methods, the clusters are formed as follows. A pattern is randomly selected from the data set V as the initial centre of a cluster. The radius of this cluster is chosen in such a way that the ratio between the number of patterns of a certain class (in-class patterns) and the total number of patterns in the cluster is not less than a pre-defined value θ. Once this cluster is formed, all patterns inside this cluster are 'removed' from the data set and do not participate in the formation of other clusters.

The value of θ is determined empirically and is related to an acceptable classification error rate. Since θ determines the radii of the clusters, it also indirectly determines the degree of overlaps between different clusters. Generally, a large θ leads to small radii of clusters; thus it leads to small overlaps between the Gaussians for different clusters and a small classification error rate for the training data set. Since a small classification error is desired, there usually exist small overlaps between the Gaussians representing the clusters.

Let us consider a simple example. Suppose $\theta = 0.8$, i.e., there must have at least 80% in-class patterns in each cluster. In Figure 2(a), suppose cluster A has been formed and its members 'removed' from the data set V. Suppose pattern 2 is subsequently selected as the initial centre of a new cluster and cluster B is thus formed. Clusters C through G are then formed similarly in sequence. We see that clusters B, C, and D are quite small, and therefore the effectiveness of the above clustering algorithm needs to be improved.

Figure 2 A comparison between (a) existing algorithms: small overlaps between clusters, and (b) the modified algorithm with reduced number of clusters: small overlaps between clusters of different classes, but large overlaps between clusters of the same class



We describe an algorithm to reduce the number of clusters as follows. We first make a copy V_c of the original data set V. When a qualified cluster (the ratio of in-class patterns should be higher than θ), e.g., cluster A in Figure 2(b) (same as in Figure 2(a)), is generated, the members in this cluster are 'removed' from the copy data set V_c, but the patterns in the original data set V remain unchanged. Subsequently, the initial centre of the next cluster is selected from the *copy data set* V_c, but the candidate members of this cluster are patterns in the *original data set* V, and thus include the patterns in the cluster A. Subsequently when pattern 2 is selected as an initial cluster centre, a much larger cluster B, which combines clusters B, C, and D in Figure 2(a), can still meet the θ-criterion and can therefore be created. By allowing for large overlaps between clusters for *the same class*, we can further reduce the number of clusters substantially. This will lead to more efficient construction of RBF networks, and is demonstrated by computer simulations in the following section.

We therefore use the following algorithm to construct an eFFcient RBF classifier, incorporating the above modification to the existing algorithms (Roy et al., 1995, 1997):

1 Initialise for training

- we divide the data set into three parts: the training data set, the validation data set, and the test data set.
- in order to derive the widths of the kernel functions, a general scale of neighbourhood δ_0 is obtained by calculating the standard deviation of the data set (Kononenko, 1994; Hruschka and Ebecken, 1999; Ishibuchi and Nii, 1996; Liu and Wechsler, 1998; Dash et al., 1997).

2 Set stage $L = 1$ (L indicates the steps in one training procedure); $\delta(L) = \delta_0$. $\delta = \alpha \times \delta_0$, where $\delta(L)$ is the initial radius of clusters at training stage L (training stage indicates the status of the training procedure, in which initial radius of clusters are affected), and δ is the increment step for the radius. α is the changing rate of radius.

3 Generate V_c, a copy of the original training data set V.

4 Forming clusters

(a) Count the number of patterns in classes in V_c. If the number of patterns in a class is fewer than a pre-defined number (we use 4 in this paper), the patterns in the class will not be selected.

(b) Set sub-stage $L_s = 1$ (L_s indicates the shrinking degree in the radius of cluster in a sub-stage training procedure), $\delta(L_s) = \delta(L)$.

(c) Select randomly a pattern from V_c as an initial cluster centre, search in V all the patterns within $\delta(L)$-neighbourhood of the centre pattern. Thus, large overlaps are permitted among clusters of the same class as we described earlier.

(d) Check whether the ratio between in-class patterns and the total patterns in the subset is greater than a pre-defined value θ. If the ratio is less than θ, set $L_s = L_s + 1$, and $\delta(L_s) = \delta(L_s) - \delta$. Search the patterns within $\delta(L_s)$-neighbourhood of the selected pattern. Stop only if the ratio criterion is met or if $L_s \geq 1/2\alpha$. Count the number of epochs N_e, if $N_e \geq Di$ (i is the number of patterns in V_c. D is an integer. Empirically, $D = 3$.), $\delta = 0.9\theta$. Repeat 4 until the training set V_c is empty.

5 Calculate the centre and width of each cluster: the centre is the mean pattern of all patterns in the cluster and the width is the standard deviation of these patterns.

6 Obtain weights by the LLS method (Bishop, 1995).

7 Calculate E_{tr} (the classification error of the training set) and E_v (the classification error of the validation set). Stop if both E_{tr} and E_v are smaller than a pre-specified value E_0. In this paper, $E_0 = E_{Pre}$ (E_{Pre} is a pre-defined value of classification error rate. $E_{Pre} = 2\%$ in this paper). Else:

- If $E_v(L) < E_v(L-1)$, set $L = L + 1$ and $\delta(L) = \delta(L) - \delta 2$. Go to 3.
- If $E_{tr}(L) > E_{tr}(L-1)$ and $E_v(L) > E_v(L-1)$, $L = L + 1$, $\delta(L) = (L) - \delta$, go to 3.
- If $L > 1/\alpha$ or $E_v > E_0$ go to 2.

Compared to Roy et al.'s (1995, 1997) original algorithm, we have made the following changes:

1 In Step 4, large overlaps among clusters of the same class are allowed in order to reduce the number of hidden units without reducing the classification accuracy.

2 In Step 6, LLS method (Bishop, 1995) is used to obtain the weights between the hidden layer and the output layer.

3 In Roy et al. (1995), six training stages were used corresponding to $\theta = \{50\%, 60\%, \ldots, 100\%\}$, respectively. In each stage, θ is unchanged and all clusters of this stage should meet the θ-criterion. A classification result is obtained for this θ. The classification error rate from a stage is compared to that in the previous stage. Whether to continue to the next training stage is determined by a comparison in the classification results. If the classification accuracy is better than the previous stage, i.e., it is possible to obtain a higher accuracy if θ is increased. Thus, the data are trained again using a larger θ. Otherwise, the training is stopped. Although a larger θ leads to a higher accuracy in the training data set, it may lead to poorer generalisation in the testing data set. Hence, in our algorithm (Step 2). θ is automatically adjusted according to the training condition, i.e., how many patterns of the class concerned are left. With the decreasing number of patterns, θ is decreased by a certain factor, say 0.9 in our simulations.

The aim of the modification described above, i.e., allowing for large overlaps among clusters of the same class, is to decrease the number of Gaussian hidden units without reducing the classification accuracy. Overlaps between clusters of *different* classes affect the classification error rate, i.e., the larger the overlaps between clusters of *different* classes, often the larger the classification error rate. However, in our paper, large overlaps between clusters of the *same* class are allowed, which is not expected to degrade the classification accuracy. Since large overlaps between clusters of the same class can help combine small clusters into larger ones, and noise may thus be suppressed by these combinations, the accuracy of classification may even be improved.

The cost of the modification is increased training time. Assume the number of patterns in the data set is N. The number of hidden units is M_1 when allowing for large overlaps among clusters of the same class, and the number of hidden units is M_2 without allowing for large overlaps. For a certain hidden unit k, the number of patterns in this cluster is N_k. The processing time for one pattern is T in the training procedure. Assume that both algorithms (with the modification and without the modification) have the same initial conditions. The initial centre of each candidate cluster is selected randomly. Thus, the average number of trials in two algorithms for searching qualified clusters may be assumed to be the same, say M_0. With the modification, all N patterns in the data set will be checked when searching for a qualified cluster in each trial. Time required for one trial is thus NT. For M_0 trials, the total time required is $M_0 \times NT$. Without the modification, if a cluster is considered to be qualified, the patterns included in this cluster will be removed from the data set. Thus, the total time required for classification without the modification is

$$\sum_{m=1}^{M_0} T\left(N - \sum_{k=1}^{m-1} N_k\right) = \left(M_0 N - \sum_{m=1}^{M_0}\sum_{k=1}^{m} N_k\right) T < M_0 NT.$$

Thus, more time is needed when applying the modification. However, the cost is worthwhile for thless complicated classifiers with higher accuracy are desired in most cases.

Based on the attribute importance ranking, we further reduce the structural complexity and improve the performance of the RBF network as follows. According to the rank of importance level obtained by the algorithm described in Section 2, J most important attributes are used for classification with the RBF neural network for $J = 1, 2, \ldots, N-1, N$. The classification error rate is calculated for each J. Thus, N classification error rates are calculated corresponding to N subsets of attributes. For small J, classification error rate decreases as J increases until all relevant attributes are included. As J increases further, the classification error rate may remain unchanged or even increase because redundant or irrelevant attributes are included. The best subset of attributes is the one with the smallest classification error rate.

4 The rule-extraction method

The rule-extraction algorithm proposed here is based on the widths and the centres of the Gaussian kernel functions, and the weights connecting the hidden neurons to the output layer. Each hidden neuron of the RBF neural network is responsive to a subset of input patterns (instances).

The rules extracted in our paper are in an IF-THEN form. Rule i (corresponding to hidden neuron i) is written as follows:

IF input 1 is within the interval (L_{1i}, U_{1i})
AND input 2 is within the interval (L_{2i}, U_{2i})
.
.
.
AND input n is within the interval (L_{ni}, U_{ni})
THEN the class label of the input pattern is k_i.

Here U_{ji} and L_{ji} are the upper limit and the lower limit of interval j in rule i, respectively.

The objective of tuning the rule premises is to determine the boundaries of rules so that a high rule accuracy is obtained for the testing data set. Before starting the tuning process, all of the premises of the rules must be initialised. Let us assume that the number of attributes is n. The number of rules equals the number of hidden neurons in the trained RBF network. The number of the premises of rules equals to n. The upper limit $U(j, i)$ and the lower limit $L(j, i)$ of the jth premise in the ith rule are initialised according to the trained.

RBF classifier as:

$$U_0(j, i) = \mu_{ji} + \sigma_i, \tag{10}$$

$$L_0(j, i) = \mu_{ji} - \sigma_i. \tag{11}$$

where μ_{ji} is the jth item of the centre of the ith kernel function. σ_i is the width of the ith kernel function.

The upper limit $U(j, i)$ and the lower limit $L(j, i)$ of the jth premise in the ith rule are tuned as follows:

$$\Delta U_t(j,i) = \eta f_t\left(\frac{\partial E}{\partial U(j,i)}\right), \tag{12}$$

$$\Delta U_{t+1}(j,i) = U_t(j,i) + \Delta U_t(j,i), \tag{13}$$

$$\Delta L_t(j,i) = \eta f_t\left(\frac{\partial E}{\partial L(j,i)}\right), \tag{14}$$

$$L_{t+1}(j,i) = L_t(j,i) + \Delta L_t(j,i), \tag{15}$$

where η is the tuning rate. Initially $\eta = 1/N_I$, where N_I is the number of iteration steps for adjusting a premise. N_I is set to be 20 in our experiments. E is the rule error rate. If the sign of $f_{t-1}(\partial E/(\partial U(j, i)))$ is different with the sign of $f_t(\partial E/(\partial U(j, i)))$, $\eta = 1.1\eta$.

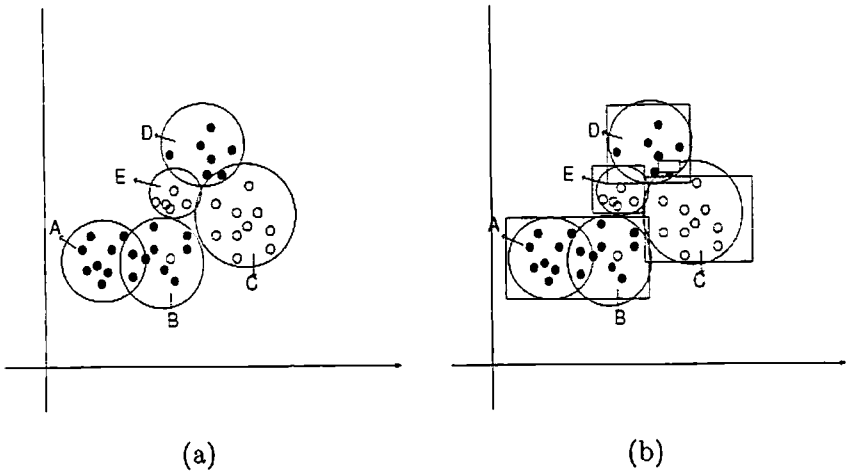
$$f_t\left(\frac{\partial E}{\partial U(j,i)}\right) = \begin{cases} 1, & \left(\frac{\partial E}{\partial U(j,i)}\right)|_t < 0 \\ -1, & \left(\frac{\partial E}{\partial U(j,i)}\right)|_t > 0 \\ f_{t-1}\left(\frac{\partial E}{\partial U(j,i)}\right), & \text{if } \left(\frac{\partial E}{\partial U(j,i)}\right)|_t = 0 \\ -f_{t-1}\left(\frac{\partial E}{\partial U(j,i)}\right), & \text{if } \left(\frac{\partial E}{\partial U(j,i)}\right)|_t = 0 \\ & \text{for } \frac{1}{3}N_I \text{ consecutive iterations.} \end{cases} \tag{16}$$

$$f_t\left(\frac{\partial E}{\partial L(j,i)}\right) = \begin{cases} 1, & \left(\frac{\partial E}{\partial L(j,i)}\right)|_t < 0 \\ -1, & \left(\frac{\partial E}{\partial L(j,i)}\right)|_t > 0 \\ f_{t-1}\left(\frac{\partial E}{\partial L(j,i)}\right), & \text{if } \left(\frac{\partial E}{\partial L(j,i)}\right)|_t = 0 \\ -f_{t-1}\left(\frac{\partial E}{\partial L(j,i)}\right), & \text{if } \left(\frac{\partial E}{\partial L(j,i)}\right)|_t = 0 \text{ for } \frac{1}{3}N_I \text{ consecutive iterations.} \end{cases} \tag{17}$$

Two rule tuning stages are used in our method. In the first tuning stage, the premises of m rules (m is the number of hidden neurons of the trained RBF network) are adjusted using gradient descent theory for minimising the rule error rate. Some rules do not make contribution to the improvement of the rule accuracy, which is due to what is stated in the following. The input data space is separated into several subspaces through training the RBF neural network. Each subspace is represented by a hidden neuron of the RBF neural network and is a hyper-ellipse. The decision boundary of our rules is hyper-rectangular.

We use gradient descent for searching the premise parts of rules. Since overlaps (Figure 3(a)) exist between clusters of the same class, some hidden neurons may be overlapped completely when a hyper-rectangular rule is formed using gradient descent (see Figure 3(b)). Thus, the rules overlapped completely are redundant for representing data and should be removed from the rule set. It is expected that this action will not reduce the rule accuracy. The number of rules will be less than the number of hidden neurons.

Figure 3 (a) Clusters in an RBF network and (b) hyper-rectangular rule decision boundaries corresponding to the clusters



(a) (b)

5 Experimental results

The Iris, Monk3, and Breast cancer data sets from the UCI Repository of Machine Learning Databases (Murphy and Aha, 1994) are used in this paper to test our algorithms for ranking attribute importance and constructing a simplified RBF network. Each data set is divided into three parts, i.e., training, validation, and test sets. We set $\alpha = 0.1$ and $\theta = 100\%$ in our experiments. Each experiment is repeated five times with different initial conditions and the average results are recorded.

Table 1 shows that when large overlaps among clusters of the same class are permitted, the number of hidden units is decreased while nearly the same classification error rate is maintained.

Table 1 Reduction in the number of hidden units in the RBF network when large overlaps are allowed between clusters for the same class. All attributes in each data set are used as inputs

Data set		*Iris*	*Monk3*	*Breast*
Classification error rate	small overlap	0.0373	0.0591	0.0292
	large overlap	0.0467	0.0688	0.0146
Number of hidden units	small overlap	5.2	33.2	34
	large overlap	4	19.6	11

Attribute importance rankings using the SCM with different χs (equation (7)) are shown in Table 2, which shows that χ affects the order of attribute importance ranking. 5 χs are used, i.e., $\chi = 0.0$, 0.4, 0.5, 0.7, and 1.0. In order to determine which order is better, a different subset of attributes are input to the RBF classifier for each order, so as to find the best subset for that order. If there are n original attributes in the data set, there are n candidate subsets of attributes as discussed in Subsection 2.4. The classification results are used to evaluate the attribute subsets. We select the subset of attributes corresponding to the lowest classification error rate for each data set and each ranking order. According to the experimental results (details for each data set are given below), when $\chi = 0.4$, the importance ranking results for the four data sets lead to the lowest or nearly the lowest validation error rates with the smallest attribute subsets.

Table 2 Attribute importance ranking using the SCM with different χ

χ	*Iris*	*Monk3*	*Breast*
0.0	4,3,1,2	5,4,2,1,6,3	7,2,4,3,8,9,5,6,1
0.4	4,3,1,2	5,2,4,1,6,3	2,7,3,4,9,5,8,6,1
0.5	4,1,3,2	5,2,4,1,6,3	2,7,3,4,9,5,1,8,6
0.7	1,4,2,3	5,2,4,1,6,3	2,7,1,3,4,9,5,8,6
1.0	1,2,4,3	5,2,3,6,4,1	1,2,7,3,4,9,5,8,6

5.1 Iris data set

There are four attributes in the Iris data set. 150 patterns are divided into three sets: i.e., 90 patterns for training, 30 for validation, and 30 for testing.

In Tables 3–6, classification error rates are shown for all attribute subsets corresponding to different attribute importance ranking results based on different χs. $\chi = 0.4$ is selected for that it leads to the smallest attribute subset {3, 4} with the nearly lowest classification error rate.

Table 3 Classification error rates for the Iris data set with different subsets of attributes according to the importance ranking shown in Table 2 when $\chi = 0.0$ and $\chi = 0.4$. The attribute subset with the lowest validation error is highlighted in italics

Attributes used	*Error rate*		
	Training set	*Validation set*	*Test set*
4	0.1222	0.0667	0.1333
4,3	*0.0333*	*0.0000*	*0.0333*
4,3,1	0.0556	0.0333	0.1000
4,3,1,2	0.0889	0.1000	0.1000

Table 4 Classification error rates for the Iris data set with different subsets of attributes according to the importance ranking shown in Table 2 when $\chi = 0.5$. The attribute subset with the lowest validation error is highlighted in italics

Attributes used	*Error rate*		
	Training set	*Validation set*	*Test set*
1	0.3333	0.2333	0.4333
1,4	0.0778	0.0000	0.1000
1,4,2	*0.0556*	*0*	*0.0333*
1,4,2,3	0.0556	0	0.0333

Table 5 Classification error rates for the Iris data set with different subsets of attributes according to the importance ranking shown in Table 2 when $\chi = 0.7$. The attribute subset with the lowest validation error is highlighted in italics

Attributes used	*Error Rate*		
	Training set	*Validation set*	*Test set*
4	0.3333	0.3667	0.4667
4,1	0.1111	0.2000	0.1000
4,1,3	0.3333	0.2333	0.4333
4,1,3,2	*0.0778*	*0.1000*	*0.0333*

Table 6 Classification error rates for the Iris data set with different subsets of attributes according to the importance ranking shown in Table 2 when $\chi = 1.0$. The attribute subset with the lowest validation error is highlighted in italics

Attributes used	*Error Rate*		
	Training set	*Validation set*	*Test set*
1	0.4556	0.5667	0.3667
1,2	0.1889	0.3333	0.2333
1,2,4	0.0778	0.0667	0.1667
1,2,4,3	*0.0444*	*0.0667*	*0.0333*

5.2 Monk3 data set

There are six attributes in the Monk3 data set. The Monk3 data have a training set with 122 patterns and a test set with 421 patterns. We divide the test set into 200 patterns for validation and 221 patterns for testing.

In Tables 7–9, classification error rates are shown for all attribute subsets corresponding to different attribute importance ranking results based on different χ's. $\chi = 0.4$ is selected for that it leads to the smallest attribute subset {2, 4, 5} with the lowest classification error rates.

Table 7 Classification error rates for the Monk3 data set with different subsets of attributes according to the importance ranking shown in Table 2 when $\chi = 0.0$. The attribute subset with the lowest validation error is highlighted in italics

Attributes used	*Error Rate*		
	Training set	*Validation set*	*Test set*
5	0.2705	0.2328	0.2100
5,4	0.2541	0.3060	0.2450
5,4,2	*0.0902*	*0.0991*	*0.0650*
5,4,2,1	0.1967	0.2371	0.2050
5,4,2,1,6	0.1148	0.0948	0.1000
5,4,2,1,6,3	0.1885	0.2112	0.2600

Table 8 Classification error rates for the Monk3 data set with different subsets of attributes according to the importance ranking shown in Table 2 when $\chi = 0.4$, $\chi = 0.5$ and $\chi = 0.7$. The attribute subset with the lowest validation error is highlighted in italics

Attributes used	*Error Rate*		
	Training set	*Validation set*	*Test set*
5	0.1880	0.3000	0.2870
5,2	0.1780	0.2830	0.2690
5,2,4	*0.0242*	*0.0585*	*0.067*
5,2,4,1	0.0899	0.3360	0.1830
5,2,4,1,6	0.0498	0.1897	0.1320
5,2,4,1,6,3	0.0328	0.2030	0.1240

Table 9 Classification error rates for the Monk3 data set with different subsets of attributes according to the importance ranking shown in Table 2 when $\chi = 1.0$. The attribute subset with the lowest validation error is highlighted in italics

Attributes used	*Error rate*		
	Training set	*Validation set*	*Test set*
5	0.2705	0.2328	0.2100
5,2	0.2213	0.1853	0.2050
5,2,3	0.1967	0.1638	0.1400
5,2,3,6	*0.1066*	*0.0690*	*0.0700*
5,2,3,6,4	0.2131	0.1767	0.1700
5,2,3,6,4,1	0.1230	0.1552	0.1600

5.3 Breast cancer data set

There are nine attributes and 699 patterns in the Breast cancer data set. Sixteen patterns with missing attribute are removed. Of the 683 patterns left, 444 were benign, and the rest were malign. In 683 patterns, 274 patterns for training, 204 for validation, 205 for testing.

For 5 χ's, there are five different attribute importance ranking results. In Tables 10–14, classification error rates for all attribute subsets corresponding to different attribute importance ranking results based on different χs are shown. $\chi = 0.4$ is selected for it leads to the smallest attribute subset {2, 3, 7} with the lowest classification error rates.

Table 10 Classification error rates for the Breast cancer data set with different subsets of attributes according to the importance ranking shown in Table 2 when $\chi = 0.0$. The attribute subset with the lowest validation error is highlighted in italics

Attributes used	*Error rate*		
	Training set	*Validation set*	*Test set*
7	0.1100	0.0803	0.1022
7,2	0.0954	0.0803	0.0949
7,2,4	0.0391	0.0511	0.0219
7,2,4,3	*0.0318*	*0.0438*	*0.0073*
7,2,4,3,8	0.0367	0.0365	0.0219
7,2,4,3,8,9	0.0244	0.0365	0.0146
7,2,4,3,8,9,5	0.0318	0.0438	0.0146
7,2,4,3,8,9,5,6	0.0342	0.0365	0.0146
7,2,4,3,8,9,5,6,1	0.0342	0.0511	0.0219

Table 11 Classification error rates for the Breast cancer data set with different subsets of attributes according to the importance ranking shown in Table 2 when $\chi = 0.4$. The attribute subset with the lowest validation error is highlighted in italics

Attributes used	*Error rate*		
	Training set	*Validation set*	*Test set*
2	0.1100	0.0803	0.1022
2,7	0.0709	0.0657	0.0876
2,7,3	*0.0269*	*0.0365*	*0.0073*
2,7,3,4	0.0391	0.0438	0.0365
2,7,3,4,9	0.0269	0.0365	0.0219
2,7,3,4,9,5	0.0342	0.0365	0.0146
2,7,3,4,9,5,8	0.0293	0.0438	0.0073
2,7,3,4,9,5,8,6	0.0269	0.0438	0.0146
2,7,3,4,9,5,8,6,1	0.0342	0.0365	0.0146

Table 12 Classification error rates for the Breast cancer data set with different subsets of attributes according to the importance ranking shown in Table 2 when $\chi = 0.5$. The attribute subset with the lowest validation error is highlighted in italics

	Error Rate		
Attributes used	*Training set*	*Validation set*	*Test set*
2	0.1443	0.1314	0.1387
2,7	0.0513	0.0511	0.0365
2,7,3	*0.0269*	*0.0292*	*0.0073*
2,7,3,4	0.0318	0.0511	0.0146
2,7,3,4,9	0.0293	0.0438	0.0219
2,7,3,4,9,5	0.0293	0.0365	0.0292
2,7,3,4,9,5,1	0.0244	0.0438	0.0073
2,7,3,4,9,5,1,8	0.0318	0.0365	0.0146
2,7,3,4,9,5,1,8,6	0.0318	0.0365	0.0146

Table 13 Classification error rates for the Breast cancer data set with different subsets of attributes according to the importance ranking shown in Table 2 when $\chi = 0.7$. The attribute subset with the lowest validation error is highlighted in italics

	Error rate		
Attributes used	*Training set*	*Validation set*	*Test set*
2	0.1443	0.1314	0.1387
2,7	0.0538	0.0584	0.0438
2,7,1	0.0685	0.0876	0.0730
2,7,1,3	*0.0342*	*0.0365*	*0.0146*
2,7,1,3,4	0.0367	0.0365	0.0219
2,7,1,3,4,9	0.0269	0.0438	0.0292
2,7,1,3,4,9,5	0.0318	0.0511	0.0146
2,7,1,3,4,9,5,8	0.0293	0.0365	0.0292
2,7,1,3,4,9,5,8,6	0.0391	0.0511	0.0292

Table 14 Classification error rates for the Breast cancer data set with different subsets of attributes according to the importance ranking shown in Table 2 when $\chi = 1.0$. The attribute subset with the lowest validation error is highlighted in italics

	Error rate		
Attributes used	*Training set*	*Validation set*	*Test set*
1	0.3423	0.3869	0.3358
1,2	0.1467	0.1314	0.1460
1,2,7	0.0660	0.0730	0.0511
1,2,7,3	*0.0489*	*0.0292*	*0.0146*
1,2,7,3,4	0.0416	0.0365	0.0146
1,2,7,3,4,9	0.0367	0.0438	0.0292
1,2,7,3,4,9,5	0.0269	0.0365	0.0146
1,2,7,3,4,9,5,8	0.0318	0.0365	0.0219
1,2,7,3,4,9,5,8,6	0.0318	0.0365	0.0292

5.4 Comparisons between top-down and bottom-up search and with other methods

In this subsection, we compare results obtained from the SCM using bottom-up and top-down search for $\chi = 0.4$. We also compare with results derived from the attribute importance ranking by Relief-F (Kononenko, 1994) and SUD (Dash et al., 1997).

The classification error rates of the RBF classifier for different subsets of Iris attributes according to the importance ranking obtained with SCM using bottom-up search when $\chi = 0.4$. We obtained the same attribute ranking results, and hence the same attribute subsets from the SCM using bottom-up and top-down search (Table 15) for Iris. It is seen that as the number of attributes used increases, the test error first decreases, reaches a minimum when the first two attributes in the attribute ranking queue are used, and then increases. Hence in the Iris data set, attributes 3 and 4 are relevant attributes for classification and are then selected, which improves the classification performance and decreases the number of inputs and the number of hidden units of the RBF neural network. The classification error rate is reduced from 0.0467 to 0.0333, and the number of Gaussian hidden units is reduced from 4 to 3 (Table 16 summarises results for different data sets).

Table 15 Comparison between importance ranking results obtained by our SCM using bottom-up and top-down search when $\chi = 0.4$, the SUD and Relief-F methods

	Decreasing order of importance			
Data set	*SCM (bottom-up)*	*SCM (top-down)*	*SUD*	*Relief-F*
Iris	4,3,1,2	4,3,1,2	3,4,1,2	4,3,1,2
Monk3	5,2,4,1,6,3	5,2,4,1,6,3	5,2,4,1,6,3	2,5,4,3,6,1
Breast	2,7,3,4,9,5,1,8,6	7,2,3,4,9,5,8,6,1	1,7,3,2,5,6,4,8,9	6,2,3,7,5,1,4,8,9

Table 16 Comparison of the numbers of hidden units and classification errors before and after irrelevant attributes are removed according to the SCM ranking method

		Data set		
Comparison		*Iris*	*Monk3*	*Breast*
Input attributes	B	1,2,3,4	1,2,3,4,5,6	1,2,3,4,5,6,7,8,9
	A	4,3	5,2,4	2,7,3
Number of hidden units	B	4	19.6	11
	A	3	11.6	5
Classification error rate	B	0.0467	0.0688	0.0146
	A	0.0333	0.067	0.0073

B: before removal, A: after removal.

Further, we carry out classification by inputting different attribute sets to the RBF classifiers based on the attribute importance ranking results obtained by SUD (Dash et al., 1997) and Relief-F (Kononenko, 1994) methods (reproduced in Table 15). We compare the classification error rates on test data sets corresponding to the selected attribute subsets obtained using the SCM, SUD, and Relief-F methods (Table 17).

Table 17 Comparison between classification error rates on testing data sets with the best attribute subsets obtained by our SCM, SUD and Relief-F methods

	Classification error rates		
Data set	*SCM*	*SUD*	*Relief-F*
Iris	0.0333	0.0333	0.0333
Monk3	0.067	0.0.067	0.09
Breast	0.0073	0.0146	0.0073

Attributes 3 and 4 lead to the lowest error rates (Table 17) in both SUD and Relief-F methods. Hence, the selected attribute subset for Iris data set is {3, 4} according to SUD and Relief-F, which is the same as the result based on our SCM method.

We obtained the same attribute ranking results and hence the same attribute subsets using our SCM with both bottom-up and top-down search for Monk3 data set (Table 15). Attributes 2, 4, and 5 should be selected for Monk3 data set, which decreases the classification error rate from 0.0688 to 0.067, the number of inputs from 6 to 3, and the number of Gaussian hidden units from 19 to 11 (Table 16). Attributes 2, 4, and 5 should be selected according to both SUD and Relief-F methods, which is the same as the attribute subset obtained based on our SCM method.

For Breast cancer data set, the attribute ranking queue corresponding to our SCM with bottom-up search is {2, 7, 3, 4, 9, 5, 1, 8, 6} and is {7, 2, 3, 4, 9, 5, 8, 6, 1} with top-down search. In both bottom-up and top-down search, attributes 2, 3, and 7 are considered to be important for classification and are then selected, which decreases the classification error rate from 0.0146 to 0.0073, the number of inputs from 9 to 3, and the number of hidden units of the RBF neural network from 11 to 5 (Table 16). The attribute subset including the first 5 attributes (attributes 6, 2, 3, 7, and 5) in the Relief-F attribute ranking queue leads to the lowest classification error rates. According to the classification results based on the ranking result of the SUD method, attributes 1, 7, 3, 2, and 5 should be selected because the subset leads to the lowest error rates. The classification error rates on test data set when the selected attribute subsets are used as inputs for RBF classifiers are 0.0073, 0.0146, 0.0073 for our SCM, SUD, and Relief-F, respectively (Table 17). Hence, the attribute subset based on our SCM method is the smallest with the highest classification accuracy.

5.5 Comparisons between our rules and rules obtained by other methods

In this section, we compare rules extracted by our method and rules obtained by other methods. For Iris data set, based on the attribute subset {3, 4} selected, two rules are obtained, with two antecedents per rule. The accuracy is 100% for testing data set (Table 18). We compare our rule-extraction results for Iris with other methods in Table 19.

For Monk3 data set, based on the attribute subset {2, 4, 5} selected above, we obtain 3 rules, with 3 antecedents per rule (Table 20). The rule accuracy is 98% for testing data set. Setiono (2000) extracted two rules, with 5.83 antecedents per rule and 100% rule accuracy for Monk3 data set based on the pruned MLP. We obtain three rules with three antecedents per rule.

Table 18 Rule accuracy and numbers of rules for Iris data set

Results	*Iris*
Rule accuracy	
training accuracy	0.0222
validation accuracy	0.0333
testing accuracy	0.0
The number of premises/rule	2
The number of rules	2

Table 19 A comparison of results for Iris data set obtained with different methods

Methodology	*Rule accuracy* (%)	*Type of decision boundary*
Modified RX algorithm based on MLP (Hruschka and Ebecken, 1999)	97.33	hyper-plane
Inputs are transformed into discrete ones artificially based on IMLP (Bologna and Pellegrini, 1998)	97.33	hyper-rectangular
Based on RBF (McGarry and MacIntyre, 1999)	80	hyper-rectangular
Based on RBF (McGarry et al., 1999)	100	hyper-rectangular
Our algorithm	100	hyper-rectangular

Table 20 Rule accuracy and numbers of rules for Monk3 data set.

Results	*Monk3*
Rule accuracy	
Training accuracy	0.0656
Validation accuracy	0.0345
Testing accuracy	0.02
The number of premises/rule	3
The number of rules	3

For the Breast cancer data set, based on the attribute subset {2, 3, 7} selected above, we obtain four rules, with three antecedents per rule (Table 21). The rule accuracy is 97.8% for testing data set. Setiono (2000) extracted 2.9 rules and obtained 94.04% accuracy for the Breast cancer data set based on the pruned MLP.

Table 21 Rule accuracy and numbers of rules for Breast cancer data set

Results	*Breast*
Rule accuracy	
Training accuracy	0.0391
Validation accuracy	0.0291
Testing accuracy	0.0219
The number of premises/rule	3
The number of rules	4

6 Conclusions

In this paper, rule extraction is carried out in order to express the concepts of data sets. A novel SCM is used to rank the importance of attributes first. According to the ranking results, different attribute subsets are used as inputs to RBF classifiers. The attribute subsets with the lowest classification error rates and the least numbers of attributes are selected. Rules are extracted based on feature subsets selected. Compared to other methods, more concise and accurate rules are extracted for the Iris and Breast cancer data sets, while for the Monk3 data set, the rule accuracy is lower, but the antecedents per rule for Monk3 data set is smaller than other methods. In addition, rules extracted by our algorithm have hyper-rectangular decision boundaries, which is desirable due to its explicit perceptibility. Our approach eliminates the need for an error-prone transformation from continuous attributes into discrete ones as required in MLP-based methods.

References

Battiti, R. (1994) 'Using mutual information for selecting features in supervised neural net learning', *IEEE Transactions on Neural Networks*, Vol. 54, pp.537–550.

Bishop, C.M. (1995) *Neural Network for Pattern Recognition*, Oxford University Press Inc., New York.

Bollacker, K.D. and Ghosh, J. (1996) 'Mutual information feature extractors for neural classifiers', *IEEE International Conference on Neural Networks*, Vol. 3, pp.1528–1533.

Bologna, G. and Pellegrini, C. (1998) 'Constraining the MLP power of expression to facilitate symbolic rule extraction', *IEEE World Congress on Computational Intelligence*, Vol. 1, pp.146–151.

Dash, M., Liu, H. and Yao, J. (1997) 'Dimensionality reduction of unsupervised data', *Ninth IEEE International Conference on Tools with Artificial Intelligence*, pp.532–539.

Devijver, P.A. and Kittler, J. (1982) *Pattern Recognition: A Statistical Approach*, Prentice-Hall International Inc., London.

Fu, X. and Wang, L. (2003) 'Data dimensionality reduction with application to simplifying RBF network structure and improving classification performance'. *IEEE Trans. System, Man, Cybern, Part B – Cybernetics*, Vol. 33, No. 3, pp.399–409.

Hruschka, E.R. and Ebecken, N.F.F. (1999) 'Rule extraction from neural networks: modified RX algorithm', *Proceedings International Joint Conference on Neural Networks*, Vol. 4, pp.2504–2508.

Ishibuchi, H. and Nii, M. (1996) 'Generating fuzzy if-then rules from trained neural networks: linguistic analysis of neural networks', *IEEE International Conference on Neural Networks*, Vol. 2, pp.1133–1138.

Kambhatla, N. and Leen, T.K. (1993) 'Fast non-linear dimension reduction', *IEEE International Conference on Neural Network*, Vol. 3, pp.1213–1218.

Kawatani, T. and Shimizu, H. (1998) 'Handwritten kanji recognition with the LDA method', *Fourteenth International Conference on Pattern Recognition*, Vol. 2, pp.1301–1305.

Kaylani, T. and Dasgupta, S. (1994) 'A new method for initializing radial basis function classifiers systems', *IEEE International Conference on Man, and Cybernetics*, Vol. 3, pp.2584–2587.

Kononenko, I. (1994) 'Estimating attributes: analysis and extension of RELIEF', *Proceedings of European Conference on Machine Learning*, pp.171–182.

Liu, C.J. and Wechsler, H. (1998) 'Enhanced Fisher linear discriminant models for face recognition', *Fourteenth International Conference on Pattern Recognition*, Vol. 2, pp.1368–1372.

Lu, H.J., Setiono, R. and Liu, H. (1996) 'Effective data mining using neural networks', *IEEE Transactions on Knowledge and Data Engineering*, December Vol. 8, No. 6, pp.957–961.

Maffezzoni, P. and Gubian, P. (1994) 'Approximate radial basis function neural networks(RBFNN) to learn smooth relations from noisy data', *Proceedings of the 37th Midwest Symposium on Circuits and Systems*, Vol. 1, pp.553–556.

McGarry, K.J. and MacIntyre, J. (1999) 'Knowledge extraction and insertion from radial basis function networks', *IEE Colloquium on Applied Statistical Pattern Recognition (Ref. no.1999/063)*, pp.15/1–15/6.

McGarry, K.J., Wermter, S. and MacIntyre, J. (1999) 'Knowledge extraction from radial basis function networks and multilayer perceptrons', *Proceedings International Joint Conference on Neural Networks*, Vol. 4, pp.2494–2497.

Moody, J. and Darken, C.J. (1989) 'Fast learning in network of locally-tuned processing units', *Neural Computation*, Vol. 1, pp.281–294.

Murphy, P.M. and Aha, D.W. (1994) *UCI Repository of Machine Learning Databases* http://www.ics.uci.edu/ mlearn/MLRepository.html, Department of Information and Computer Science, University of California, Irvine, CA.

Nabney, I.T. (1999) 'Efficient training of RBF networks for classification', *Ninth International Conference on Artificial Neural Networks (Conf. Publ. No. 470)*, Vol. 1, pp.210–215.

Poechmueller, W., Hagamuge, S.K., Glesner, M., Schweikert, P. and Pfeffermann, A. (1994) 'RBF and CBF neural network learning procedures', *IEEE World Congress on Computational Intelligence*, Vol. 1, pp.407–412.

Roy, A., Govil, S. and Miranda, R. (1995) 'An algorithm to generate radial basis function (RBF)-like nets for classification problems', *Neural Networks*, Vol. 8, No. 2, pp.179–201.

Roy, A., Govil, S. and Miranda, R. (1997) 'A neural-network learning theory and a polynomial time RBF algorithm', *IEEE Transactions on Neural Network*, November, Vol. 8, No. 6, pp.1301–1313.

Saha, A. and Wu, C.L (1993) 'Approximation, dimension reduction, and nonconvex optimization using linear superpositions of gaussians', *IEEE Transactions On Computers*, October, Vol. 42, No. 10, pp.1222–1233.

Setiono, R. (2000) 'Extracting M-of-N rules from trained neural networks', *IEEE Transactions on Neural Networks*, March, Vol. 11, No. 2, pp.512–519.