Computational Statistics (2001) 16:361–372

Computational Statistics

© Physica-Verlag 2001

Linguistic Rule Extraction From a Simplified RBF Neural Network

Xiuju Fu[1] and Lipo Wang[2]

[2] School of Electrical and Electronic Engineering
Nanyang Technological University
Block S2, Nanyang Avenue
Singapore 639798
Email: elpwang@ntu.edu.sg
http://www.ntu.edu.sg/home/elpwang

SUMMARY

Representing the concept of numerical data by linguistic rules is often desirable. In this paper, we present a novel rule-extraction algorithm from the radial basis function (RBF) neural network classifier for representing the hidden concept of numerical data. Gaussian function is used as the basis function of the RBF network. When training the RBF neural network, we allow for large overlaps between clusters corresponding to the same class, thereby reducing the number of hidden units while improving classification accuracy. The weights connecting the hidden units with the output units are then simplified. The interval for each input in the condition part of each rule is adjusted in order to obtain high accuracy in the extracted rules. Simulations using some bench-marking data sets demonstrate that our approach leads to more accurate and compact rules compared to other methods for extracting rules from RBF neural networks.

KEYWORDS: rule extraction, data mining, radial basis function neural network, overlaps, classifier, concise rules

1 INTRODUCTION

Extracting rules to represent the concept of data is an important aspect of data mining. The concept of data refers to the distribution of data, the relationship between patterns (samples) of data and their corresponding class labels, and the relationship between attributes (variables of samples) and class labels, etc.. The task of extracting concise, i.e., a small number of, rules from a trained neural network is usually challenging due to the complicated architecture of a neural network.

Some research work has been carried out in extracting rules through RBF neural networks. In [2][5], redundant inputs are removed before rule extraction. Huber [7] selects rules according to importance; however, the accuracy is reduced with pruning. McGarry [11][12][13] extracts rules from RBF neural networks by considering the parameters of Gaussian kernel functions and weights which connect hidden units to the output layer. However, when the number of rules is small, the accuracy is low. When the accuracy is acceptable, the number of rules becomes large.

In this paper, we propose a novel technique to extract rules from the RBF neural network. First, the number of hidden units is reduced due to our modification when training the RBF neural network. The weights connecting hidden units with output units are simplified (pruned) subsequently. Then the interval for each input in the condition part of each rule is adjusted in order to obtain a high rule accuracy. We show that our technique leads to a compact rule set with desirable accuracy.

This paper is organized as follows. In Section 2, we proposed a way to obtain an efficient RBF classifier based on a modification in which large overlaps are permitted between clusters with the same class label. Our proposed algorithm to extract rules from the trained RBF classifier is described in Section 3. Experimental results are shown in Section 4. Finally, Section 5 presents the conclusions of this paper.

2 THE RBF NEURAL NETWORK

The RBF neural network [1] attracts much attention because of its simplicity in structure and fast speed in learning.

[1]Xiuju Fu, studying for Ph.D degree in School of EEE, Nanyang Technological University, Block S2, Nanyang Avenue, Singapore 639798, Email: p146793114@ntu.edu.sg

In the RBF neural network, the activation of a hidden unit is determined by the distance between the input vector and the center vector of the hidden unit. 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. Assume there are M classes in the data set, we write the m-th output of the network as follows:

$$y_m(\mathbf{X}) = \sum_{j=1}^{K} w_{mj}\phi_j(\mathbf{X}) + w_{m0}b_m \quad , \tag{1}$$

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

$$\phi_j(\mathbf{X}) = \mathrm{e}^{\frac{||\mathbf{X}-\mathbf{C_j}||^2}{2\sigma_j^2}} \quad , \tag{2}$$

where $\mathbf{C_j}$ and σ_j are the center and the width for the j-th hidden unit, respectively, and are determined during learning.

Therefore, the density of the input data is modeled by the mixture model of Gaussian functions:

$$p(\mathbf{X}) = \sum_{j=1}^{M} P(j)\phi_j(\mathbf{X}) \tag{3}$$

where the parameters $P(j)$ are the prior probabilities for the data points included in the receptive field of the jth kernel function, and ϕ_j is the jth Gaussian kernel function of the network.

The weights connecting the hidden layer and the output layer can be determined by the linear least square (LLS) method [1][16], which is fast and free of local minima, in contrast to the multilayer perceptron neural network.

3 CONSTRUCTING AN EFFICIENT RBF CLASSIFIER

We now discuss the effect of overlapped receptive fields of Gaussian kernel functions of the RBF neural network on the number of its hidden units. Overlapped receptive fields of different clusters can improve the performance of the RBF classifier in rejecting noise when tackling with noisy data [10].

However, the overlaps between different classes will decrease the accuracy of classification when tackling with noise-free data.

It is desirable for an RBF classifier to have a small number of hidden units, and at the same time, a low classification error rate. A modification in training RBF neural network for simplifying the construction of RBF neural networks is now proposed. As known, classification error rate is mainly determined by the degree of overlap between clusters for *different classes*, and is independent of the degree of overlap between clusters for the same class. In the above clustering algorithm, if cluster radii are increased by decreasing θ (θ is the ratio of the number of in-class patterns and the total number of patterns in one cluster), the number of clusters will decrease; however, the classification error rate will also increase because of larger overlaps between clusters for *different classes*. But if we modify the clustering algorithm such that small overlaps between clusters for different classes are maintained, and large overlaps between clusters for the same class are allowed, the classification error rate should remain the same with the number of clusters decreased. That is, some clusters are enlarged in size, and at the same time, θ-criterion is satisfied in all clusters, i.e., the ratio between the number of in-class patterns and the total number of patterns in one cluster should be above a certain value.

The following steps is used to permit large overlaps between clusters with the same class label. A copy V_c of the original data set V is made first. When a qualified cluster (θ-criterion is satisfied), e.g., cluster A in Fig.1(b) (same as in Fig.1(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 is unvaried. Subsequently, the initial center 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, thus include the patterns in the cluster A. Subsequently when pattern 2 is selected as an initial cluster center, a much larger cluster B, which combines clusters B, C, and D in Fig.1(a), can still satisfy the θ-criterion and can therefore be created.

In order to obtain fewer number of clusters with high accuracy in classification at the same time, a dynamic θ is used. The initial value of θ is 100%. There are two conditions to determine whether to reduce θ or not. One condition is if the ratio between the number of remaining patterns in V_c and the number of patterns in V_c in the previous epoch (one epoch is one presentation of each pattern in the data set) is less than a certain ratio α_1. The other condition is if the epoch number for searching a suitable cluster exceeds one certain number R_{Count}. If either of the two conditions is satisfied, the θ will be reduced by a certain value (we set the value as 2.5%), since θ affects the accuracy of classification, i.e., the higher the θ, the higher the classification accuracy. The minimal θ used in this paper is 75%.

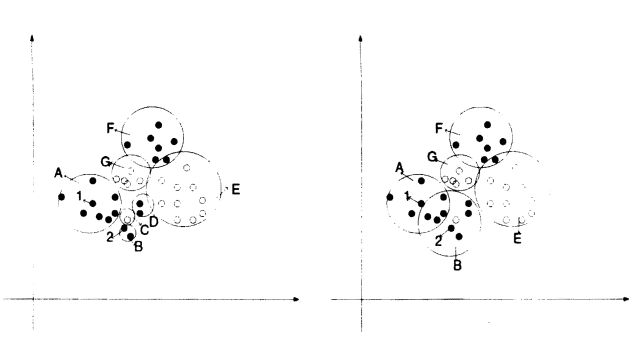

Figure 1: (a) no large overlap (b)large overlap

When training RBF neural networks, the initial centers for clusters are chosen randomly (without considering the order of classes), which gives an equal chance for each class to form clusters, in contrast to the Gaussian Masking (GM) algorithm [14]. In addition, a few patterns left will not be considered as candidates of initial cluster centers, i.e., isolated patterns (symbol K is used to represent the number of isolated patterns) in each class will be omitted in the clustering process, in order to minimize the number of clusters.

We therefore use the follow algorithm to construct an efficient RBF classifier, incorporating the above modification to the existing algorithms [14][15]:

1. Initialize for training:
   a) We divide the data set into three parts, the training data set, the validation data set, and the test data set.
   b) In order to derive the widths of the kernel functions, a general scale of neighborhood δ_0 is obtained by calculating the standard deviation of the data set.
2. Set stage $L = 1$; $\delta(L) = \delta_0$, $\delta = \alpha * \delta_0$, where $\delta(L)$ is the initial radius of clusters at training stage L 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 of all classes in V_c. If the number of patterns left for one class is larger than a predefined value K (the value corresponds to the number of isolated patterns), we continue, else go to step 7.

b) Set sub-stage $L_s = 1$, $\delta(L_s) = \delta(L)$.

c) Select randomly a pattern (from V_c) and search in V to find all the patterns in $\delta(L)$-neighborhood of the selected pattern. Thus large overlaps are permitted among clusters of the same class as we proposed above.

d) Check whether the ratio between the number of in-class patterns and the number of total patterns in the subset is greater than the pre-defined value θ, if true, a cluster is formed, then the patterns in the cluster is removed from V_c. 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 $\delta(L_s)$-neighborhood of the selected pattern. Stop only if the ratio criterion is satisfied or if $L_s \geq 5$. Repeat 4 until the training set V_c is less than $M * K$ (M is the number of classes in the data set).

5. Calculate the center and width of each cluster: the center 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 [1].

7. Calculate E_{tr} (the classification error of the training set) and E_v (the classification error of the validation set). Stop if both of E_{tr} and E_v are less than a pre-specified value E_0. In this paper, E_0 =0.01. Else:

   a) If $E_v(L) < E_v(L-1)$, set $L = L + 1$ and $\delta(L) = \delta(L) - \delta/2$. Go to 3.

   b) If $E_{tr}(L) > E_{tr}(L-1)$ and $E_v(L) > E_v(L-1)$, $L = L + 1$, $\delta(L) = \delta(L) - \delta$,go to 3.

   c) If $L > 10$ or $E_v > E_0$ go to 2.

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, i.e., the number of hidden units will be reduced. The experimental results will be shown in Section 4.

4 RULE EXTRACTION

After training the RBF neural network, the concept of data is memorized in the RBF classifier. We now try to explain the concept of data through the RBF classifier. Since each hidden unit of the RBF neural network is responsive to a subset of patterns (instances), the weights connecting the hidden unit with output units can reflect for which output the hidden unit

mainly serves. Our rule extraction algorithm is directly based on the widths, centers of Gaussian kernel functions, and weights connecting hidden units and the output layer.

We first determine the corresponding output unit which each hidden unit mainly serves for by simplifying the connections between hidden units and output units. Consider the weight matrix (assume there are m hidden units, and n output units)

$$W = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1n} \\ x_{21} & x_{22} & \cdots & x_{2n} \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ x_{m1} & x_{32} & \cdots & x_{mn} \end{pmatrix} ,$$

where x_{ij} is the element of the weight matrix, and it refers to the weight connecting the ith hidden unit and the jth output unit.

The matrix will be converted into

$$W_1 = \begin{pmatrix} 0 & \cdots & x_{1k_1} & \cdots & 0 \\ 0 & \cdots & x_{2k_2} & \cdots & 0 \\ \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots & \cdots\cdots \\ 0 & \cdots & \cdots & x_{mk_m} & 0 \end{pmatrix} ,$$

where x_{jk_j} is the maximum value of each row j ($j = 1, ...m$) of matrix W. Thus, W_1 reflects the corresponding output which each hidden unit mainly serves for.

The suitable intervals of attributes make up the premise part of an extracted rule, i.e., the decision boundary of our rules is hyper-rectangular. Let us assume that the number of attributes is N. The upper limit $Upper(j,i)$ and the lower limit $Lower(j,i)$ of the jth attribute in the ith rule are initialized as:

$$Upper(j,i) = \mu_{ji} + \rho_{j,i} \quad , \tag{4}$$

$$Lower(j,i) = \mu_{ji} - \rho_{j,i} \quad , \tag{5}$$

$$\rho_{j,i} = \eta_j * x_{ik_i} * \sigma_i \quad . \tag{6}$$

Here μ_{ji} is the jth item of the center of the ith kernel function. Initially, $\eta_j = 1$. σ_i is the width of the ith kernel function. x_{ik_i} is the corresponding element in W_1. $\rho_{i,j}$ will be adjusted by adjusting η_j as follows:

$$\eta_j = \eta_j + Sign * 0.025. \tag{7}$$

Sign has two value +1 and −1, which is determined by the trend of change in the rule accuracy when adjusting. The initial *Sign* is +1. If the rule accuracy becomes lower with the adjusted η, *Sign* is changed to −1. Otherwise, *Sign* remains the same. The stop-criterion for the iteration is a predefined rule error rate. When adjusting the intervals to obtain high accuracy, the validation set is used for determining the direction of adjusting, which can help the extraction rules not to fit the training data set too well, and obtain good results in the testing data set.

Compared with the technique proposed by McGarry [11][12][13], a higher accuracy with concise rules is obtained in our method. In [11][13], the input intervals in rules are expressed in the following equations:

$$X_{upper} = \mu_i + \sigma_i - S \quad , \tag{8}$$

$$X_{lower} = \mu_i - \sigma_i + S \quad , \tag{9}$$

Here S is feature "steepness", which was discovered empirically to be about 0.6 by McGarry. μ_i is n-dimensional center location of rule i, and σ_i is the width of receptive field. We note that the empirical parameter may not be suitable to all data sets. The experimental results are shown in the next section.

5 THE EXPERIMENTAL RESULTS

Two data sets, Iris and Thyroid from the UCI Repository of Machine Learning Databases, are used for testing our methods. There are 4 attributes and 3 classes in Iris data set. There are 5 attributes and 3 classes in Thyroid data set. Each data set is divided into 3 parts, i.e., training, validation, and test sets. 90 patterns of Iris data set is used for training, 30 patterns for validation and 30 patterns for testing. There are 215 patterns in Thyroid data set. 115 patterns are for training, 50 patterns for validation and 50 patterns for testing. We set $\alpha_1 = 0.8$ and we set the maximum number of epoches for searching a cluster to be R_{Count}=20. α is the changing rate of radius and we set it as 0.9 in our experiments. The number of isolated patterns K is 2. A pattern is randomly selected first, if a qualified cluster whose center is the selected pattern can not be found after adjusting η to search in the total data set for R_{Count} times, then the selected pattern will be released, a new pattern will be selected again in order to generate a new cluster.

in our experiments. The results shown in Table 1 are the average values of 5 independent experiments. The smallest number of hidden units in constructing an RBF neural network classifier is 3 for Iris data set. For Thyroid, at least 6 hidden units are needed.

Table 1: Reduction in the number of hidden units

Comparisons	Iris		Thyroid	
	Small overlap	Large overlap	Small overlap	Large overlap
Error rate in classification	0.0491	0.0387	0.048	0.0385
Number of hidden units	5.2	3.4	13.8	7.4

Table 1 shows that when large overlaps among clusters of the same class are permitted, both the number of hidden units and the classification error rate for testing data are decreased.

After the learning procedure and using our proposed rule extraction method, we obtain 3 symbolic rules for Iris data set.

The accuracy of the symbolic rules that we obtain through the proposed method is 90%-94% for Iris data set. For Thyroid data set, 6 rules are obtained, with 5 conditions in each rule, and the accuracy is 80%-85%.

For Iris data set, 3 rules are obtained, the accuracy is 94% for testing data.
rule 1:

IF the sepal length is within the interval (3.24, 5.59)

AND the sepal width is within the interval (2.27, 4.31)

AND the petal length is within the interval (0.28, 2.33)

AND the petal width is within the interval (0.00, 0.57)

THEN the class label is Setosa.

rule 2:

IF the sepal length is within the interval (4.28, 7.71)

AND the sepal width is within the interval (1.24, 3.37)

AND the petal length is within the interval (2.50, 5.49)

AND the petal width is within the interval (0.46, 1.66)

THEN the class label is Versicolor.

rule 3:

IF the sepal length is within the interval (5.26, 7.90)

AND the sepal width is within the interval (2.31, 4.11)

AND the petal length is within the interval (4.31, 6.90)

AND the petal width is within the interval (1.79, 2.50)

THEN the class label is Virginica.

Default rule:

the class label is Virginica.

We now compare our results with rule-extraction results using RBF neural networks. In [5], 5 or 6 rules were needed to represent the concept of Iris data (the accuracy was not available). Huber [7] extracted 8 rules to represent Iris data set (the accuracy is not available). In order to get a small rule base, unimportant rules were pruned according to ranking [7]. However, the accuracy of rules was reduced [7] at the same time. McGarry [11][12][13] extracted rules from RBF neural networks directly from the parameters of Gaussian kernel functions and weights. In [11], the accuracy reached 100%, but the number of rules was large (for the Iris data set, 53 rules were needed). In [12] and [13], the number of rules for the Iris data set was small, i.e., 3, but the accuracy of the extracted rules was only 40% and around 80%, respectively. The results of extracted rules for Thyroid data set using other methods were not available.

Many rule-extraction methods have been explored based on the multilayer perceptron (MLP). Desirable results have been obtained both in accuracy and numbers of rules (e.g., [3][6][8][9]). Compared with the rule extraction techniques using MLP, the accuracy of the rules extracted from the RBF neural networks is lower, however, the training of the RBF neural network can escape from local minima, which is very important for large data sets. The architecture of the RBF neural network is simpler and the training time is usually shorter in comparison with the MLP.

For Iris data set, Fu [4] reported 94% in rule accuracy by C4.5, and the number of rules was 5. Though there was higher rule accuracy reported for Iris by the decision tree algorithm (C4.5) [3] (97.33% in testing data set, 3 rules), it had been shown that neural networks generated rules of better performance than the decision tree algorithm in noisy conditions [4].

6 CONCLUSIONS

A useful modification in constructing and training the RBF network by allowing for large overlaps among clusters of the same class is proposed, which reduces the number of hidden units while maintaining the classification performance. The reduced number of hidden units can facilitate us in searching concise rules. Rule extraction is carried out from the simplified RBF classifier in order to explain and represent the concept of data. In order to identify

clearly which hidden unit serves for which class, the weights between the hidden layer and the output layer are simplified first. Then the interval for each input as the premise of each rule is determined by iterations. The validation data set is used for making sure that the result is fit for training data set and testing data set at the same time. Our rule extraction technique is simple to implement, which is shown through our experimental results, and concise rules with high accuracy are obtained based on the RBF classifiers.

References

[1] Christopher M. Bishop, *Neural network for pattern recognition*, Oxford University Press, New York, 1995.

[2] T. Brotherton, G. Chadderdon, and P. Grabill, "Automated rule extraction for engine vibration analysis", *Proc. 1999 IEEE Aerospace Conference*, vol. 3, pp. 29-38, 1999.

[3] G. Bologna and C. Pellegrini, "Constraining the MLP power of expression to facilitate symbolic rule extraction", *Proc. IEEE World Congress on Computational Intelligence*, vol. 1, pp. 146-151, 1998.

[4] LiMin Fu, "Rule Generation from Neural Networks", *IEEE Transaction on systems, man, and cybernetic*, Vol. 24, No. 8, August, 1994.

[5] S. K. Halgamuge, W. Poechmueller, A. Pfeffermann, P. Schweikert, and M. Glesner, "A new method for generating fuzzy classification systems using RBF neurons with extended RCE learning Neural Networks", *Proc. IEEE World Congress on Computational Intelligence*, vol. 3, pp. 1589-1594, 1994.

[6] E. R. Hruschka and N. F. F. Ebecken, "Rule extraction from neural networks: modified RX algorithm", *Proc. International Joint Conference on Neural Networks*, Vol. 4, pp. 2504-2508, 1999.

[7] K.-P. Huber and M. R. Berthold, "Building precise classifiers with automatic rule extraction", *Proc. IEEE International Conference on Neural Networks*, vol. 3, pp. 1263-1268, 1995.

[8] H. Ishibuchi, M. Nii, and T. Murata, "Linguistic rule extraction from neural networks and genetic-algorithm-based rule selection", *Proc. International Conference on Neural Networks*, vol. 4, pp. 2390-2395, 1997.

[9] H. Ishibuchi and T. Murata, "Multi-objective genetic local search for minimizing the number of fuzzy rules for pattern classification problems", *Fuzzy Systems Proceedings, 1998. IEEE World Congress on Computational Intelligence*, vol. 2, pp. 1100-1105, 1998.

[10] P. Maffezzoni and P. Gubian, "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, 1994.

[11] K. J. McGarry, S. Wermter, and J. MacIntyre, "Knowledge extraction from radial basis function networks and multilayer perceptrons", *Proc. International Joint Conference on Neural Networks*, vol. 4, pp. 2494-2497, 1999.

[12] K. J. McGarry, J. Tait, S. Wermter, and J. MacIntyre, "Rule-extraction from radial basis function networks", *Proc. Ninth International Conference on Artificial Neural Networks*, vol. 2, pp. 613-618, 1999.

[13] K. J. McGarry and J. MacIntyre, "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, 1999.

[14] A. Roy, S. Govil, and R. Miranda, "An algorithm to generate radial basis function (RBF)-like nets for classification problems", *Neural networks*, vol. 8, no. 2, pp. 179-201, 1995.

[15] Asim Roy, Sandeep Govil, and Raymond Miranda, "A neural-network learning theory and a polynomial time RBF algorithm", *IEEE Transactions on neural network*, vol. 8, NO. 6, pp. 1301-1313, November 1997.

[16] N. Sundararajan , P. Saratchandran, and Y. W. Lu , *Radial basis function neural networks with sequential learning: MRAN and its applications*, Singapore , River Edge, N.J.: World Scientific, 1999.