One Step Strategy for Learning RBF Network Parameters

Mladen Široki

Faculty of Mechanical Engineering and Naval Architecture, University of Zagreb, Zagreb, Croatia

In this paper a new, one step strategy for learning Radial Basis Functions network parameters is proposed. In the RBF network model developed by Poggio and Girosi three modifiable sets of parameters: positions of the centers $\mathbf{t}$, weighed norm $\|\mathbf{x}-\mathbf{t}\|_{\mathbf{w}}^2$ and output layer weights $\mathbf{c}$ have to be determined during the learning stage. The authors suggest that these parameters be set by some iterative nonlinear optimization method, such as gradient descent, conjugate gradient or simulated annealing method. The basic idea of this work is: if hidden layer radial basis functions are set to be a multivariate Gaussian function, unknown parameters can be learned from the training set much faster, in a single step, by well known statistical methods, than by iterative optimization. In this approach the positions of the centers are learned by K-means clustering method, weighed norms are calculated as a Mahalanobis distances between $\mathbf{x}$ and $\mathbf{t}$, and optimal output layer weights are found by pseudoinversion. Calculation of Mahalanobis distances involves estimation of hidden units covariance matrices Σ, that replace weighed matrices $\mathbf{W}$. Two classification examples illustrate the usefulness of the method.

Keywords: Neural Networks, Radial Basis Functions Networks, Learning, Classification

1. Introduction

The Radial Basis Functions networks (RBF, also called regularization networks or Hyper Basis Functions — Hyper BFs) are three layer feedforeward artificial neural networks shown in Fig. 1. The first layer of the network consists of the input units, whose number n is equivalent to the number of variables of the problem (dimension of the input vector $\mathbf{x}$). Second (hidden) layer is made of nonlinear units fully connected to the input layer. Each hidden unit is parametrizied by its center (vector $\mathbf{t}$). Number and positions of the hidden units should be set during the learning stage. Each hidden unit i evaluate the function $h(\|\mathbf{x}-\mathbf{t}_i\|)$, where h is a radial basis function, $\|\mathbf{x}-\mathbf{t}_i\|$ is a distance between input point given by vector $\mathbf{x}$ and the center of the hidden unit i given by vector $\mathbf{t}_i$ in n dimensional space. Output layer is linear and fully connected to the hidden layer. Weights of the output layer $\mathbf{c}$ are the unknown coefficients that should be determined during the learning stage.

Poggio and Girosi [Po89] have shown that learning an input-output mapping from an example is a problem of hypersurface reconstruction, and is related to classical approximation techniques, including regularization theory. They have also shown that regularization is equivalent to the RBF network shown on Fig. 1. when all data points are used as centers of hidden units.

RBF networks form mappings from an n dimensional input vector $\mathbf{x}$ to an output $F(\mathbf{x})$ ($(R^n \rightarrow R$ mapping) of the form

$$F(\mathbf{x}) = \sum_{i=1}^{N} c_i h(\|\mathbf{x}-\mathbf{t}_i\|) \qquad (1)$$

where c_i are the weights to be determined, and N is the total number of data points. In the case of interpolation (expansion 1) all points from the training set should be used as centers. The unknown coefficients c_i can be recovered imposing the interpolation conditions $F(\mathbf{x}_j) = y_j (j = 1,\ldots,N)$, that substituted in equation (1) yields the linear system

$$y_j = \sum_{i=1}^{N} c_i h(\|\mathbf{x}_j-\mathbf{t}_i\|) \quad j = 1,\ldots,N \qquad (2)$$

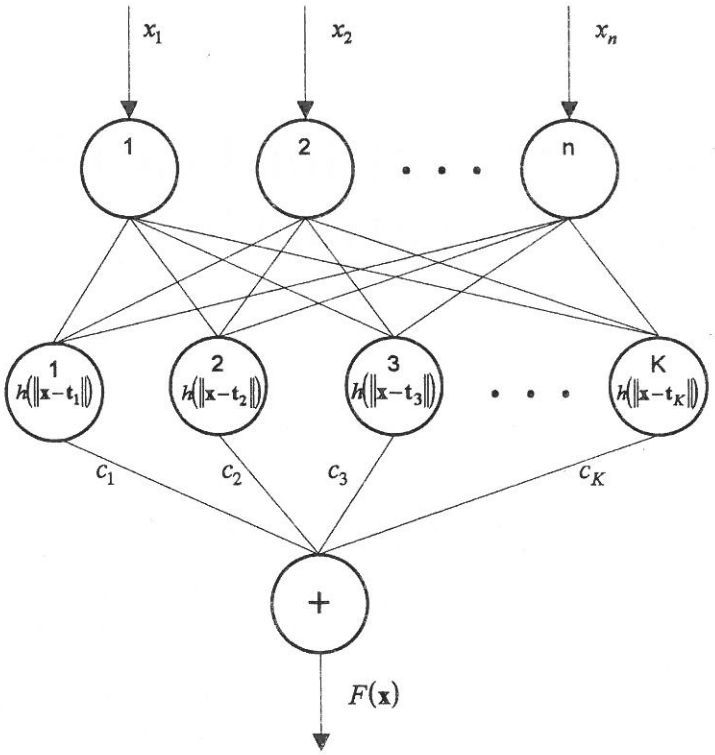

Fig. 1. Three layer feedforward RBF network

where $\mathbf{x}_j$ is the input vector and y_j is the desired output of pattern j from the training set.

Defining vectors $(\mathbf{y})_j = y_j$, $(\mathbf{c})_i = c_i$ and symmetric $(N \times N)$ matrix $(\mathbf{H})_{ij} = h(\|\mathbf{x}_j - \mathbf{t}_i\|)$, the coefficients of the equation (2) are given by

$$\mathbf{c} = \mathbf{H}^{-1}\mathbf{y} \tag{3}$$

According to the equations (1)–(3) structure of the RBF network is completely determined by the problem, and, what is even more important, weights c_i can be calculated by inversion of the matrix $\mathbf{H}$ in only one step. Disadvantage of the method is in the number of samples N which is often very large, which makes the calculation of matrix inversion very time consuming, if not even practically impossible. Also, the training data mostly contain a certain amount of noise, and in this case generalization of the network could be better accomplished if hypersurface is approximated rather than interpolated through the training points. Approximation can be accomplished by an expansion of the following form

$$F(\mathbf{x}) = \sum_{i=1}^{K} c_i h(\|\mathbf{x} - \mathbf{t}_i\|) \tag{4}$$

where $\mathbf{t}_i$ are K points (centers), whose coordinates have to be chosen, and $K < N$.

Imposing $F(\mathbf{x}_j) = y_j$ in expansion (4) leads to the following linear system

$$y_j = \sum_{i=1}^{K} c_i h(\|\mathbf{x}_j - \mathbf{t}_i\|) \quad j = 1, \ldots, N \tag{5}$$

The system is overconstrained (N equations for K unknowns) and by least-squares approach the optimal solution can be found by expansion

$$\mathbf{c} = \mathbf{H}^{+}\mathbf{y} \tag{6}$$

where $(\mathbf{H})_{ij} = h(\|\mathbf{x}_j - \mathbf{t}_i\|)$ is a rectangular $(N \times K)$ matrix and $\mathbf{H}^{+}$ is the Moore–Penrose pseudoinverse of $\mathbf{H}$ that can be computed as

$$\mathbf{H}^{+} = (\mathbf{H}^T\mathbf{H})^{-1}\mathbf{H}^T \tag{7}$$

Poggio and Girosi went further proposing an extension to RBF network with two sets of modifiable parameters in addition to the weights $\mathbf{c}$: moving centers and adjustable norm-weights [Po90]. They have suggested that moving centers is equivalent to task-dependent clustering and changing the norm weights is equivalent to task-dependent dimensionality reduction.

With this extension the output of the RBF network is calculated by expansion

$$F(\mathbf{x}) = \sum_{i=1}^{K} c_i h(\|\mathbf{x} - \mathbf{t}_i\|_{\mathbf{W}}^2) \qquad (8)$$

where $\|\mathbf{x} - \mathbf{t}_i\|_{\mathbf{W}}^2$ is weighed norm calculated by

$$\|\mathbf{x} - \mathbf{t}_i\|_{\mathbf{W}}^2 = (\mathbf{x} - \mathbf{t}_i)^T \mathbf{W}^T (\mathbf{x} - \mathbf{t}_i) \qquad (9)$$

and $\mathbf{W}$ is the square matrix that is the subject of learning.

So far, RBF network has been presented as a method for interpolation and approximation of continuous functions. Computer experiments show that RBF network can also be successfully used for classification and Boolean learning problem. Broomhed and Lowe used RBF network to learn XOR problem [Br88], Moody and Darken used similar kind of network for classification of 10 distinct vowel sounds [Mo89], Wolpert showed that a simple form of RBF perform well on the classification task of NetTalk [Po89], etc. In the case of classification the structure of RBF network is the same as presented in Fig. 1., only the output layer has more units, and their number is equal to the number of classes present in the problem. This paper is mainly concerned with using RBF network as a classifier, and in the following section the one step learning strategy for classification problems will be proposed.

2. One Step learning Method

According to the expansion (8), RBF networks have three modifiable sets of parameters $(\mathbf{c}, \mathbf{t}, \mathbf{W})$ that have to be determined during the learning stage. When compared to other neural networks paradigms, the main advantages of RBF network are that these parameters have a clear interpretation, and also, that the original form of learning (equations 1–7) can be performed in one step only. Every center vectors $\mathbf{t}_i$ present one cluster of training data, $\mathbf{W}$ weighs the importance of input variables and $\mathbf{C}$ weighs the output from hidden layer units. Poggio and Girosi have shown that these parameters can be optimized using gradient-descent method analogous to backpropagation. They have also suggested that some other iterative optimization methods, such as conjugate gradient or simulated annealing method may be more efficient than gradient descent and should be used in practice. In their model they used the same matrix $\mathbf{W}$ for all hidden units.

In this work hidden layer radial basis functions are set to be a general multivariate normal density function (Gaussian) defined by

$$h_i(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}|\Sigma_i|^{1/2}} \times \\ \times \exp\left(-\frac{1}{2}(\mathbf{x} - \mu_i)^T \Sigma_i^{-1} (\mathbf{x} - \mu_i)\right) \qquad (10)$$

where $\mathbf{x}$ is an n component input vector, μ_i is an n component mean vector (in our case center of hidden layer unit $i(\mu_i = \mathbf{t}_i)$), and Σ_i is a covariance matrix of training set points that belongs to center i.

The quantity

$$r^2 = ((x - \mu_i)^T \Sigma_i^{-1} (\mathbf{x} - \mu_i)) \qquad (11)$$

s called the squared Mahalanobis distance [Du73] from $\mathbf{x}$ to μ_i, and is obvious that it has the same meaning as the weighed norm used in Poggio and Girosi development, defined by expansion (9).

The basic idea of this work is that if the hidden layer radial basis functions are set to be multivariate Gaussian functions defined by expansion (10), unknown hidden layer parameters (centers coordinates $\mathbf{t}_i = \mu_i$, and covariance matrices Σ_i) can be determined directly from the training set using some well known statistical pattern recognition techniques, much faster than by optimization process proposed by Poggio and Girosi.

The first problem is to find centers coordinates $\mathbf{t}_i$. In this work, similarly as in the works of Poggio and Girossi, Moody and Darken and others, the positions of the centers are determined by K-means clustering method. Some other unsupervised clustering techniques, including Kohonen networks can also be applied. The second and more critical problem is estimation of the covariance matrices Σ_i. If it is possible to assume that training data belongs to cluster i, present by its center $\mathbf{t}_i$, are normaly distributed over this

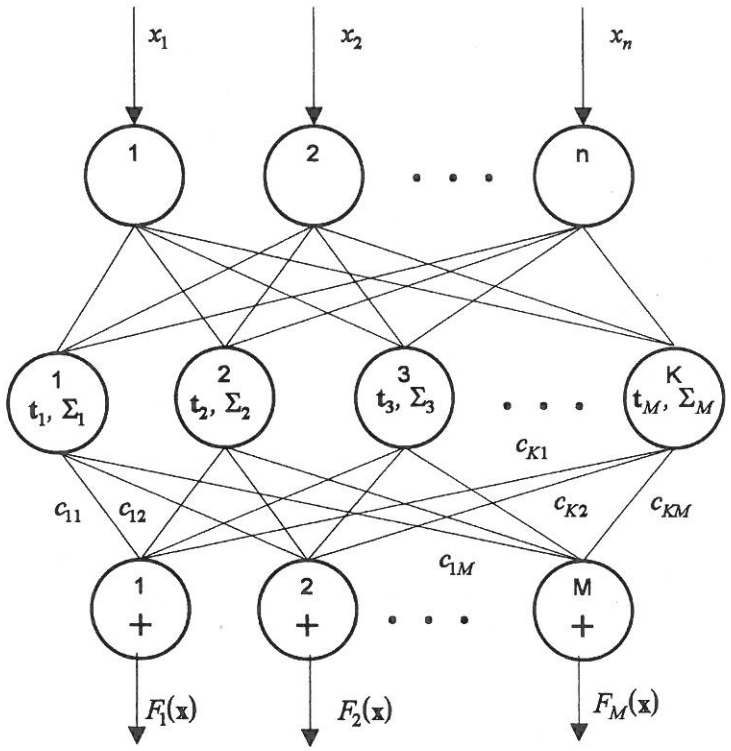

Fig. 2. RBF network for classification problem

cluster, covariance matrices Σ_i can be estimated by expansion

$$\Sigma_i = \frac{1}{N_i - 1} \sum_{j=1}^{N_i} (\mathbf{x}_{ij} - \mu_i)(\mathbf{x}_{ij} - \mu_i)^T \quad (12)$$

where Σ_i is square symmetrical covariance matrix of samples belongs to cluster i, N_i is number of samples belongs to cluster i, $\mathbf{x}_{ij}$ are training data belongs to cluster i and $\mu_i = \mathbf{t}_i$ is cluster i mean vector.

According to the expansion (11), matrix Σ_i must be inverted to obtain Mahalanobis distances. In [Du73] it is shown that if the number of samples per cluster N_i is less or equal to the number of input variables n, matrix Σ_i is guaranteed to be singular. This means that the training set should consist of at least $n + 1$ samples per cluster, and in most of the cases a good estimate of Σ_i requires several times more samples per cluster than is the dimension n of input vector.

This method is primarily developed for classification problems, and in this case experiments have shown that better results accomplished when hidden layer parameters $(\mathbf{t}, \Sigma)$ were learned for each class, and each cluster of the class separately. After the hidden layer parameters are determined, optimal output layer weights $\mathbf{C}$ can be calculated by the least square approach using equations (4)–(7). Estimation of hidden layer covariance matrices is derived from Bayesian learning, and when there is only one center set for each class (in this case the center is the mean vector of the class) output from hidden layer unit i $h_i(\mathbf{x})$ is equivalent to the Bayesian determination of conditional densities $p(\mathbf{x}|\omega_i)$ (probability that input vector $\mathbf{x}$ belongs to class ω_i). The difference between this method and the Bayes classifier is that here it is possible to set more clusters for the same class, and also the outputs from the hidden units are weighed by output layer weights $\mathbf{C}$.

The proposed RBF network for classification is shown in Fig. 2. Unknown parameters can be determined by the following algorithm.

Algorithm

1. Set the number of centers for each class.

2. Calculate the position of centers for each class separately using K-means clustering method, or some other clustering technique.

3. Group the data to the clusters for each class

separately, and estimate clusters unbiased covariance matrices by expansion (12).

4. Calculate the inversion of covariance matrices.

5. Calculate the squared Mahalanobis distance r^2 from each input vector $\mathbf{x}$ of the training set to each hidden unit center $\mathbf{t}_i$, what yields rectangular $(N \times K)$ matrix $\mathbf{R}$, where N is total number of training data and K is total number of centers (RBF hidden units).

6. Calculate matrix $\mathbf{H}$ by expansion

$$(\mathbf{H})_{ij} = h(\mathbf{R})_{ij} \tag{13}$$

where $h(r)$ is multivariate Gaussian function defined by expansion (10). In equation (10) $(2\pi)^{n/2}|\Sigma_i|$ is only a scaling factor and it can be dropped in calculation.

7. According to the equations (4)–(7) impose the interpolation conditions to calculate the output layer weights $\mathbf{C}$ by expansion

$$\mathbf{C} = \mathbf{H}^{+}\mathbf{Y} \tag{14}$$

where $\mathbf{C}$ is $(K \times M)$ weight matrix, $\mathbf{H}^{+}$ is $(K \times N)$ Moore Penrose pseudoinversion of $\mathbf{H}$, and $\mathbf{Y}$ is $(N \times M)$ target output matrix and M is the number of different classes present in the problem.

If the number of training samples is not large enough, some of the covariance matrices Σ_i can be singular (as is described, this happens when N_i is less or equal to n). In this case one solution is to choose a smaller number of centers per class, and another possibility is to pool the data that belong to the same class, and calculate only one covariance matrix for each class. With the second approach all hidden layer units that represent the same class share the same covariance matrix.

The proposed method is local in terms of calculating centers and covariance matrices, but global in terms of calculating the output layer weights. The advantages of local training is that it teaches hidden units centers $\mathbf{t}_i$ for each class separately. It also weighs the influences of input variables for each class and each cluster of the class separately, as it is reasonable to assume that some variables are not equally important for every class. Global training of output layer weights ensures that weights are optimal (by least-squares approach) for the chosen parameters of hidden layer units. The number of hidden units per class has to be established during the learning process by trial and error method.

3. Examples

In this section two classification examples are presented to illustrate the usefulness of the proposed method. In both examples there is one output unit per class. For each class, if an input is a member of that class, the corresponding training output is 1, otherwise it is 0. The RBF classifier performs classification by assigning each input to the class corresponding to the linear output unit with the largest output value. The error rate is determined by computing the percentage of samples of a given class that were wrongly assigned to some other class by the RBF classifier. The overall error rate is the percentage of all input samples that were misclassified. The learning error rate is an average square error rate on training set calculated by expansion

$$E = \frac{1}{2 \times N \times M} \sum_{i=1}^{N} \sum_{j=1}^{M} (OUT_{ij} - y_{ij})^2 \tag{15}$$

where N is the total number of training data, M is the number of output units (classes), OUT_{ij} is the actual output of output unit j calculated for i-th training sample, and y_{ij} is the desired output of output unit j for the i-th training sample.

Computer programs were written in 3L Parallel C Programming language, and run on the single Intel T800, 20 MHz transputer. In all simulations the positions of the centers were determined by K-means clustering method for each class separately, and the pseudoinversion of matrix $\mathbf{H}$ is calculated by singular value decomposition [Pr89].

3.1 Circle-in-the-square example

The circle-in-the-square problem requires a system to identify which points of the square lie inside (class A) and which lie outside a circle (class B) whose area is half that of the square. This task was specified as a benchmark problem for system performance evaluation in the DARPA artificial neural network technology

(ANNT) program [Wi90]. Wilensky examined the performance of $2-n-1$ back-propagation systems on this problem. He studied systems where the number (n) of hidden units ranged from 5 to 100, and corresponding number of weights ranged from 21 to 401. Training sets ranged in size from 150 to 14000. To avoid overfitting, the training was stopped when accuracy on the training set reached 90%. This criterion level was reached most quickly (5 000 epochs) in systems with 20 to 40 hidden units. In this condition, approximately 90% of test set points, as well as training set points, were correctly classified. Carpenter at al. have investigated the same problem with Fuzzy ARTMAP classifier [Ca92]. They have shown that test set error rate is reduced from 11.4% to 2% as training set increases from 100 to 100 000 in one epoch simulations (single step learning). They have also shown that with iterative learning test error rate is reduced, and for training set of 100 items error rate is 11%, 1 000 items 5%, 10 000 items 1.7 % and finally for 100 000 items error rate is only 0.5%. In all experiments test set consist of 1 000 randomly chosen exemplars. In this work, due to the equipment limitation, experiments with only 100 and 1 000 samples training sets were performed. Same as in [Ca92], the test set was made of 1 000 samples. The RBF networks were trained and tested with a different number of centers (hidden units). In all of the simulations the number of centers was the same for each class, because all of the classes share equal a priori probability, and also the number of training data were the same for each class.

In the first experiment a 100 item training set (50 per class) was used. In simulations the number of centers (hidden units) increases from 1 to 50 per class (2 to 100 hidden units). In the last case, when 100 centers were set, all of the training data points were used as centers. In this experiment only two covariance matrices, one for each class, were estimated. So, all the centers (hidden units) that belong to the same class share the same covariance matrix. Results of this experiment are presented in Tab. 1. Tab. 1. shows that increasing the number of centers decreases the learning error rate. Using all training data as centers leads to the interpolation approach (matrix $\mathbf{H}$ becomes square, and weights $\mathbf{C}$ can be recovered by inversion of $\mathbf{H}$), and learning error becomes zero. However, increasing the number of centers doesn't necessarily decrease classification error rate measured on the test set, because of the generalization problem, and the Tab. 1. shows that the best performance was achieved with 15 centers per class.

In the second experiment (Tab. 2.) same training and test sets as in the first experiment were used. The difference was that covariance matrices were estimated for each center separately, so each hidden unit had its own covariance matrix. In this experiment smaller number of centers per class was used than in the first experiment, to ensure that some of the covariance matrices were not singular, or close to singular, when numerical error occured. Comparison of the Tab. 1. and

Centers per class	1	2	3	5	7	10	15	20	30	50
Class A error rate %	25.0	25.2	18	14.6	8.0	3.2	3.4	4.4	4.8	10.6
Class B error rate %	6.4	45.0	27.4	5.2	6.0	5.6	5.0	8.6	21.2	46.2
Overall error rate %	15.7	35.1	22.7	9.9	7.0	4.4	4.2	6.5	13.0	28.4
Learning error rate %	7.7	7.8	5.4	3.2	2.3	1.8	1.5	1.1	0.5	0.0

Tab. 1. Classification results on circle-in-the square example, training set consists of 100 samples, one covariance matrix for each class estimated

Centers per class	1	2	3	4	5
Class A error rate %	25.0	24.0	13.4	10.0	6.6
Class B error rate %	6.4	15.2	14.4	9.8	7.6
Overall error rate %	15.7	19.6	13.9	9.9	7.1

Tab. 2. Classification results on circle-in-the square example, training set consists of 100 samples, covariance matrix estimated for each center separately

Centers per class	1	3	5	10	15	20	25
Class A error rate %	28.6	11.2	13.2	6.2	1.4	2.4	2.0
Class B error rate %	0.0	27.6	4.2	1.4	1.2	1.8	1.6
Overall error rate %	14.3	19.4	8.7	3.8	1.3	2.1	1.8
Learning time seconds	0.7	3.4	9.2	26.9	55.7	65.7	83.6

Tab. 3. Classification results on circle-in-the square example, training set consists of 1 000 samples, covariance matrix estimated for each center separately

Centers per class	1	2	3	5	10	15	25
No. Wrong Setosa	0	0	0	0	0	0	0
No. Wrong Versicolor	0	0	0	0	2	1	0
No. Wrong Virginica	1	0	0	1	2	3	4
No. Wrong Total	1	0	0	1	4	4	4

Tab. 4. Iris example — classification results on test set consisting of 75 samples, one covariance matrix per class estimated

2. shows that, with equal number of centers per class, better results were obtained when covariance matrices were estimated for each hidden units separately. However with more then 5 centers per class, some of the covariance matrices became singular, and the network failed to learn the problem. That leads to the conclusion, that a larger training set should be used, if covariance matrices are to be estimated for each hidden unit separately.

Finally in the last experiment (Tab. 3.) a training set of 1 000 samples (500 per class) was used. Same as in previous experiment covariance matrices were estimated for each class separately. Tab. 3. shows that the best result was achieved with 15 centers per class (in this case the structure of the RBF network was 2-30-2) when only 1.3% of test samples were misclassified, and learning took only 55.7 seconds. This results are much better then the performances of back-propagation system [Wi90], and they are also better than fuzzy ARTMAP results [Ca92] on the training set of the same size.

3.2 Iris example

The second example is the famous E. Anderson iris data example used by R. A. Fisher in his classic paper on discriminant analysis (Fisher 1936). This example was selected because of the tremendous number of results available from a wide range of classification techniques that will provide a measure of relative performance. The iris data consisted of 150 four-dimensional feature vectors in three separate classes (setosa, versicolor and virginica), 50 vectors per each class. In the first experiment (Tab. 4. and Tab. 5.) the training set is produced by 25 randomly selected patterns from each class. Remaining patterns were used for testing. The second experiment (Tab. 6.) was performed by leaving one out method (N experiments where N is the total number of data available — 150 in this problem, in each experiment $N-1$ samples are used for training and remaining one for testing) [Du73]. Like in the first example, the RBF

Centers per class	1	2	3	5	10	15	25
No. Wrong Setosa	0	0	0	0	0	0	0
No Wrong Versicolor	2	2	2	2	1	1	0
No Wrong Virginica	1	1	1	0	0	0	0
No Wrong Total	3	3	3	2	1	1	0
Learning Error rate %	6.95	5.91	4.95	4.41	1.94	0.91	0.00

Tab. 5. Iris example — classification results of training set consisting of 75 samples, one covariance matrix per class estimated

Centers per class	1	2	3	5	10	15	25
No. Wrong Setosa	0	0	0	0	1	0	0
No. Wrong Versicolor	3	2	4	4	2	2	3
No. Wrong Virginica	0	0	0	0	1	2	5
No. Wrong Total	3	2	4	4	4	4	8

Tab. 6. Iris example — classification results of all 150 patterns by leaving one out experiment, one covariance matrix per class estimated

network was trained and tested with a different number of centers, the number of centers being equal for all classes. In both experiments only one covariance matrix per class was estimated, so all the hidden units that represent the same class share the same covariance matrix. Finally in the last experiment, also performed by leaving one out method, for each cluster its own covariance matrix was estimated. In this experiment only up to 3 clusters per class were set. With more than 3 clusters per class some of the covariance matrices became singular, and it wasn't possible to calculate Mahalanobis distances from these centers. The results of this experiment are shown in Tab. 7.

Tab. 4. shows interesting results where the classifications with 5 or less centers per class performed better on the test set than on the training set (Tab. 5). An explanation could be that most of the problematic patterns were randomly selected to the training set. In another experiment the sets were exchanged: the test set was used for training and the training set was used for testing. The results obtained by exchanged set were opposite to these shown in tables 4 and 5. (better performance on training set than on test set). Tab. 4. also shows very good generalization property especially for networks with 5 and less centers per class. This example shows the same tendency as the previous example where increasing the number of centers decrease the learning error but doesn't necessarily decrease the classification error rate measured on the test set. To produce more reliable results leaving one-out experiment was performed. Results of this experiment are shown in Tab. 6. This table shows that the best classification was performed with only two centers per class resulting in only two of 150 samples being misclassified. Tab. 7. shows the result of leaving one-out experiment, when covariance matrices were estimated for each center (hidden unit) separately. This experiment is equal to the previous one when only one center per class is set, so the classification results for this case correspond to those in Tab. 6., with two and three centers per class results are better, and only one of 150 samples being misclassified. For the purpose of comparison, classification results obtained by different classification methods for the same problem are shown in Tab. 8. This table is taken from the work of P. K. Simpson [Si92].

Results presented in Tab. 8. show that classification technique proposed in this work performs better on Iris example than any other traditional, fuzzy and neural network classifiers shown in Tab. 8.

4. Conclusion and further Work

In this paper a one step strategy for learning classification problems with RBF network is proposed. The main advantage of the method is that, unlike many other iterative methods, parameters of a network can be learned very fast in one step only. Experimental results presented in this paper are very promising.

Restrictions, and possibly the areas where more progress with further investigation could be achieved are:

a) the number of hidden units has to be set by trial and error method

Centers per class	1	2	3
No. Wrong Setosa	0	0	0
No. Wrong Versicolor	3	1	1
No. Wrong Virginica	0	0	0
No. Wrong Total	3	1	1

Tab. 7. Iris example — classification results of all 150 patterns by leaving one out experiment, covariance matrices estimated for each center separately

Technique	No. Wrong
Bayes classifier[1]	2
k-nearest neighbor[1]	4
Fuzzy k-NN[2]	4
Fisher ratios[1]	3
Ho-Kashyap[1]	2
Perceptron[3]	3
Fuzzy perceptron[3]	2
Fuzzy min-max network[1]	2
RBF proposed in this work[4]	1

Tab. 8. A Comparison of the Classification performance of Various Traditional, Fuzzy and Neural Classifiers (Table is taken from [Si92])

[1] The training set comprised 75 data points (25 from each class) and the test set comprised the remaining points.

[2] The training set comprised 36 data points (12 from each class) and the test set comprised another 36 data points. The results were then scaled up to 150 total points for the comparison.

[3] The training and testing data were the same

[4] All data points were used for training and testing by leaving one out method

b) the method holds only for problems where it is possible to assume that patterns are normally distributed over the centers of the classes

c) the number of samples per center should be greater than the number of input variables of the problem

The experiments presented here are quite simple, with small number of inputs (2 in the first and 4 in the second experiments), and small number of classes (two and three classes problems). Currently, the author is trying to apply the method to more complicated problems with much bigger number of inputs and output classes, such as alphabetical character recognition, texture classification and others. In these problems insufficient data problem may arise, and some modification to the approach presented in this paper should be applied. The results obtained so far seem to be promising, and the author hopes that they will be presented in some further paper.

5. References

[Br88] D. S. BROOMHEAD AND D. LOWE, Multivariable functional interpolation and adaptive networks, Complex Systems, 2:321–355, 1988.

[Ca92] G. A. CARPENTER, S. GROSSBERG, N. MARKUZON, J. H. REYNOLDS AND D. B. ROSEN, Fuzzy ARTMAP: A Neural Network Architecture for Incremental Supervised Learning of Analog Multidimensional Maps, IEEE Transactions on Neural Networks, Vol 3, No. 5, pp 698–713, 1992.

[Du73] R. O. DUDA AND P. E. HART, Pattern Classification and Scene Analysis, Wiley, New York 1973.

[Gi89] F. GIROSI AND T. POGGIO, Networks and the Best Approximation Property, A. I. Memo 1164, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1989.

[Gi92] F. GIROSI, Some extensions of Radial Basis Functions and their applications in artificial intelligence, Computers Math. Applic. Vol 24, No. 12, pp. 61–80, 1992.

[Mo89] J. MOODY AND C. DARKEN, Fast learning in networks of locally tuned processing units, Neural Computation, 1(2): 281–294, 1989.

[Mo90] B. MOORE, Theory of Networks for Learning, SPIE Vol. 1294, pp. 22–30, 1990.

[Po89] T. POGGIO AND F. GIROSI, A Theory of Networks for Approximation and Learning, A. I. Memo 1140, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1989.

[Po90] T. POGGIO AND F. GIROSI, Extensions of a Theory of Networks for Approximation and Learning: dimensionality reduction and clustering, A. I. Memo 1167, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, 1990.

[Pr89] W. H. PRESS, B. P. FLAMNERY, S. A. TEUKOLSKY AND W. T. VETTERLING, Numerical Recipes The Art of Scientific Computing, Cambridge: Cambridge University Press, 1989.

[Si92] P. K. SIMPSON, Fuzzy Min-Max Neural Networks — Part 1: Classification, IEEE Transactions on Neural Networks, Vol 3, No. 5, pp 776–786, 1992.

[To74] J. T. TOU AND R. C. GONZALEZ, Pattern Recognition Principles, London: Addison Wesley, 1974.

[Wi90] G. WILENSKY, Analysis of neural network issues: Scaling, enhanced nodal processing, comparison with standard classification, DARPA Neural Network Program Review, Oct. 29–30, 1990.

Received: April, 1995
Accepted: October, 1995

Contact address:

Mladen Široki
Department of Control Engineering
Faculty of Mechanical Engineering
and Naval Architecture
University of Zagreb
Ivana Lučića 5, Zagreb
Croatia
phone: +385 1 6111 944 / 375
fax: + 385 1 615 69 40
telex: 22648 fsb Croatia
e-mail: mladen.siroki@fsb.hr

MLADEN ŠIROKI (1963) received the B.Sc., M.Sc. and Ph.D. degrees in Mechanical Engineering from the Faculty of Mechanical Engineering and Naval Architecture (FSB), University of Zagreb in 1987, 1992 and 1996 respectively. He is lecturer at the Department of Control Engineering, FSB, University of Zagreb. His interests include artificial intelligence, neural networks, pattern recognition and image processing.