Minimal Topology for a Radial Basis Functions Neural Network for Pattern Classification

Adrian G. Bors and Moncef Gabbouj

Signal Processing Laboratory, Tampere University of Technology, P.O. Box 553, Tampere, Finland

In the context of pattern classification, the success of a classification scheme often depends on the geometrical properties of the pattern classes under consideration. As radial basis functions (RBF) neural networks have largely been applied in pattern classification problems, in this paper we present a brief overview of different trends in radial basis functions neural networks and their applications. The meanings of the weights and the processing units for a RBF network applied for pattern classification are given. A new learning algorithm for a RBF neural network is proposed in this paper. This algorithm gives a solution for classifying configurations of patterns in a feature space providing the minimum number of hidden units for the network implementation. The learning is based on the backpropagation algorithm. The performance of the proposed algorithm is assessed on different artificial and real applications. The algorithm is successfully applied for estimating a distribution, as well as for separating signals in a multiple access communication system and for recognizing static speech.


1. INTRODUCTION

In this paper, after a brief overview of the principal trends in radial basis functions neural networks, we propose a solution for finding the minimal number of hidden units for a radial basis functions structure and we apply this algorithm to different artificial and real tasks.

Radial basis functions (RBF) neural networks have been used extensively in many engineering and scientific applications [4, 18, 20, 23]. The aim is to model an unknown system, which has observable inputs and outputs, regarded as synthesizing an approximation of a set of multidimensional functions.

The principal task in a pattern recognition scheme is to find a suitable mapping between a given set of patterns and their corresponding classes. The first stage is the selection of features for the respective patterns; the second is the classification of the patterns in the feature space. In order to accomplish this task, it is necessary to establish some rules upon which to base the classification decisions. In a supervised learning task, a neural network is presented with a set of given input–output pairs which after a learning stage retrieves the appropriate output for the given input and generalizes for new inputs. In a pattern classification task, the neural network approximates the a posteriori conditional probability for each class $p(C_i | x)$, $i = 1, \ldots, M$, where $M$ is the number of classes and $x$ is an $N$-dimensional vector denoting a pattern. In nonparametric methods this is done based on an efficient learning of the a priori probabilities $p(x|C_i)$ drawn from the test set, where the a priori and a posteriori probabilities are related through the Bayes rule. The estimation of these probabilities is equivalent to finding the optimal boundaries between different classes. The success of a pattern classification scheme depends on the geometrical properties of the pattern classes under consideration and by the algorithm characteristics employed for the respective task. Different characteristics which must be taken in consideration for a neural network are the network architecture, the number of neurons, and the type of activation function. To obtain a good generalization, one has to memorize into the network as much knowledge about the problem as possible while limiting the number of units and connections appropriately. A two-layer feed forward neural network was found sufficient to model an arbitrary input output mapping [1] and is suitable to be used as a general pattern clas-
sifier. An important problem is to find for a selected activation function the optimal number of parameters or hidden units needed for a given task. If there are too few units the network memorizes poorly, and if there are too many, then it generalizes poorly.

Radial basis activation functions for the processing units provide the network with the capability of forming more complex nonlinear mappings in the input space, which is equivalent to what multilayer perceptrons can provide through an intermediate mapping [11]. The RBF network implements a two-layer structure, where the hidden layer functions are of squashing type. They were found suitable for approximate unknown nonlinear functions [4–9,15,16] and to implement a close replica of the optimal classifier [7,11]. The RBF network requires less computation time for the learning [8], and a more compact topology than other neural networks [11]. This method is similar to the Parzen windows for the estimation of probability density functions [3]. In the Parzen windowing method, there are as many functions as observations from the underlying distribution, each of them centered on one observation. The RBF method uses a decomposition of the density function in a mixture of a finite number of component densities, where the component densities are implemented by the hidden units. This mapping constructs a linear function space of the basis kernels, which depends on the position of the known data points according to a given metric. RBF networks have been used successfully in time-series prediction [9,12,15,16], in speech recognition [7,8,17], in adaptive control [10], in equalization [12,20–22], in signal detection [23], etc.

Broomhead and Lowe [4] studied the capability of RBF for multivariable functional interpolation. The generalization property of the network is synonymous with the interpolation between the data points. For a given set of data points, there may be an infinity of decompositions in different kernel functions, constrained to pass through the given observations. The output weights are computed in this case using the inverse of the matrix with basis functions taken in different points and the corresponding values for the desired outputs. This matrix should not be singular in order to obtain a solution. In [5], using simple properties of the basis functions, it is shown that a neural network with a single layer of hidden units of Gaussian type is a universal approximator. These results were extended in [6], where certain conditions were given in order to prove that a class of RBF networks with the same smoothing factor in each kernel node is suitable for universal approximation. In [7], Niranjan and Fulbrook have studied the problem of nonlinear pattern classification and have shown the similarity of the RBF method with potential functions (Ai-

zerman, Braverman, and Rozonoer [25]), kernel discriminant analysis, and other methods. A comparative study for the classes boundaries, obtained for different methods is provided for a XOR configuration and for a random problem, as well as for static speech classification [7].

In [9], Poggio and Girosi treated the problem of networks for approximation in a theoretical framework, based on the regularization techniques. The approximation of a continuous function depends on finding the weights for the given basis. The distance between the function and its approximation is measured by a cost function. The gradient descent algorithm can be used to find suitable values for the weights in order to obtain a good approximating representation for the underlying function. The cost function assumed in [9] is given by

$$ E(X) = \sum_i (Y_i(\mathbf{X}_i) - F_i)^2 + \lambda \| \mathbf{P} \mathbf{Y}(X) \|^2, \tag{1} $$

where $\lambda$ is the regularization parameter and is related to the degree of generalization, $\mathbf{P}$ is an operator, $Y_i$’s are the outputs of the network and $F_i$’s are the desired outputs. The operator $\mathbf{P}$ represents prior knowledge about the problem and it is usually a derivative operator. For a radially symmetric $\mathbf{P}$ the solution is given by

$$ Y(\mathbf{X}) = \sum_i c_i G(\| \mathbf{X} - \mathbf{X}_i \|), \tag{2} $$

where $G(X)$ is a Green function related to $\mathbf{P}$. The regularization networks are equivalent to the generalized splines and to RBF. The network of Poggio and Girosi has two layers where the number of hidden units is set a priori. The centers of these hidden units are a subset of the input samples. The number of inputs is equal to the number of independent variables of the problem. Before using the gradient algorithm, a suitable initialization is given for the weights. In order to decrease the high computational complexity required for finding exact solutions when the sample size is large, an approximation to the regularized solution was introduced in [9]. In [18], a network with radially Gaussian units whose centers are uniformly distributed, is shown to have the best approximation property. The error bound of the approximation is derived in terms of the number of hidden units.

To implement a RBF network one should know the type of function used, the number of inputs, hidden units, and outputs (the structure denoted in this paper as $N$-$L$-$M$, respectively), as well as the input and the output weights. In [8], Moody and Darken gave a solution for using a smaller number of hidden units than data using a local representation for hid-
den units. The number of hidden units is chosen a priori and no special assumptions are made about the basis functions such as orthonormality or being uniformly distributed over the input space. The centers of the basis functions are found through the k-means clustering algorithm. The basis functions used are radially symmetric, where the widths can be computed using different "P nearest neighbour" methods. These methods vary the widths to achieve different response overlap between each unit and its neighbors. The output weights are found through a supervised learning algorithm using the LMS algorithm (Wiener [26]). Because of the unsupervised clustering algorithm which precedes it, the convergence of the LMS occurs quickly. In [8], the algorithm was applied for the prediction of a chaotic time series and for a phoneme classification problem. The chaotic time series which was chosen is Mackey–Glass differential equation. The topology used for this problem had two real inputs, one real output and between 100–10,000 hidden units. This approach clearly treats problems where there are less hidden units than training samples.

Sanner and Slotine [10] employed a network with Gaussian RBF to adaptively compensate the nonlinearities of an adaptive control system. The learning and control are attempted simultaneously in a control system and the stability should be guaranteed. A constructive procedure was given for selecting centers and variances of a finite number of Gaussian nodes so that the network is capable of approximating the underlying function to a certain tolerance on a subset. A nonadaptive component in the algorithm for the regions where the approximation is inaccurate, is required for a globally stable control. Derived from the sampling theory, the controller function can be estimated in a set of small hypercubes obtained through the sampling of the state space using an appropriate interpolating function. The centers as well as the variances which are common for all the basis functions are calculated on the basis of the spectral truncation radius and of the oversampling parameter, which controls the distance to the first copy of the repeating spectra. Thus, the feed-in weights are assumed fixed and only the feed-out weights are adjusted. It was shown that this system is a globally stable closed loop system with tracking errors which converge to zero.

In order to find the optimal number of hidden units for a neural network, different methods were used such as increasing or decreasing architecture or using a regularization factor in the cost function similar to that from (1). The number of hidden units depends on the training set of the learning data used, as well as the type of activation function.

For RBF in [11], a hierarchically self-organizing algorithm was used in order to find the minimal number of hidden units. Different types of basis functions are analyzed, e.g., sigmoidal, sinusoidal, and Gaussian. The generalization power of an activation function may be dependent on the local characteristics of a particular mapping. The interpolation accuracy needs to be ensured adaptively through the addition of new units. The Gaussian RBF was found suitable not only in generalizing a global mapping but also in refining local features without much altering the already learned mapping. The weights associated with one hidden unit are \( N^2 + N + M \), where \( N \) inputs and \( M \) outputs are assumed; the cross-correlation terms of the underlying covariance matrix are also taken into account. The learning algorithm recovers new computational units whenever necessary for improving network performance. The network starts with no hidden units and adds a new unit with a larger width in the beginning of the learning, which is reduced for the following units until a minimal radius is obtained. After adding a new node all the weights are adjusted using backpropagation. Each basis unit has an accommodation boundary, defined in the input space, upon which the respective unit has influence. A new hidden unit is added if a new pattern cannot be placed in the accommodation boundary of one of the existing RBF, which has the same class representation as that of the new sample. The algorithm will otherwise update the parameters of the existing basis functions until saturation is reached. The parameter saturation vector \( s_j \) for the \( j \)-th unit is given by the expression

\[
s_j(p) = \alpha \frac{\partial E}{\partial w_j} + (1 - \alpha)s_j(p - 1).
\]

where \( \alpha \in (0, 1) \) is the decaying factor, \( p \) represents the teaching pattern, and \( E \) is the cost function (in this case the squared error of the pattern \( p \) for all the outputs) and \( w_j \) 's are the weights of the \( j \)-th unit. An important factor in this method is the procedure for reducing the radius of the new added unit. If the effective radii of the RBF are reduced too rapidly then the network will generate more units than the minimum required because individual basis functions may not have time to converge to their optimal shapes. The reduction of the widths enables the network to learn the details. A few pattern classification problems as well as continuous function approximation were considered as applications for the algorithm in [11].

The orthogonal least-squares algorithm [12] is used for the selection of the positions of RBF centers. The problem of selecting the RBF centers from the data set can be regarded as the selection of a subset of significant regressors, based on the individual contributions to the desired output energy of each ortho-
nal basis vector. This algorithm, using batches of training sets for learning, was applied to the modeling of a real-world time series and as a communication channel equalizer. Orthonormal RBF neural networks were studied in [13], where each hidden unit is trained individually. Once the number of hidden units is found, the output weights must be recomputed. The decreasing architecture method for RBF was explored in [14], where groups of patterns in the input space are allowed to be in the activation field of the same unit, adjusting the input to hidden weights. The solution is a supervised clustering algorithm, where groups of patterns in the input space are allowed to be placed in the activation field of the same unit. The clusters determined through the k-means algorithm are successively merged if they correspond to patterns from the same class. To find the principal axes and the length along each axis for a basis function (the weights), the Gram–Schmidt orthogonalization procedure is used.

An on-line learning algorithm for RBF called resource allocating was introduced by Platt [15]. This network allocates a new computational unit whenever the network performs poorly on a newly presented pattern. If the network performs well, the network parameters are updated using standard LMS algorithm. The network starts with no hidden units; as patterns are presented to it, according to a novelty condition, they are compared with the patterns which have been previously stored. A new pattern is considered novel if the input is “far away” from the existing centers, and if the distance between the network output and desired output is “large.” In order to reduce the number of units, the centers are also updated using gradient descend. The algorithm was tested on a particular chaotic time series generated by Mackey–Glass delay-difference equation. This model was extended in [17] to include multiple outputs, where the basis functions are common to all output mappings. The adaptation is applied only for hidden to output weights using the Kalman filter. A lower bound similar with that in [15] was derived for the novelty criteria. This ensures that the number of hidden units is bounded. The algorithm was applied in prediction of the chaotic series [16] and in static speech classification [17]. 

In this paper we use Gaussian RBF network with two full-connected layers structure which allows many outputs and we give a solution for finding the minimal number of hidden units. We give the interpretation of the network elements in Section 2 in the context of pattern classification. In Section 3, we introduce an algorithm based on backpropagation which finds the minimal topology required by a given problem. This algorithm provides a solution to the problems encountered in conventional learning techniques due to the existence of local minima, and it has good convergence properties.

In Section 4, different artificial examples are provided where we consider classification problems with mixed patterns from different classes with a great degree of overlapping between the classes as well as non-parametric estimation of distributions. In the last part of Section 4, an application of this network to equalization for a multiple access communication system is provided, where the network learns to model the inverse of a set of different channels. Another application considered in this paper is in static pattern recognition for a subset of the Peterson-Bailey vowels problem, known as a benchmark in speech recognition. Some conclusions are given in Section 5.

2. THE NETWORK STRUCTURE

A set of optimal boundaries between different classes can generally be described as nonlinear functions which should be estimated in a pattern classification task. We can always represent a continuous N-dimensional function through a linear combination of kernel functions. This decomposition is in a manner straightforward implemented in a two-layer full connected RBFN. The network can be viewed as a mapping from the features hyperspace to the classes. We consider each pattern represented through a vector and we assign for each class a suitable code.

The network can be fed with real vectors, each entry corresponding to a pattern feature. The inputs of the network model an input hyperspace where the boundaries between the different classes are built by the network. For the hidden units’ activation functions, we use a locally tuned function,

$$\sigma'[X_i] = \sum_{k=1}^{N} \left( \frac{w[j, h] - X_i[h]}{r[j, h]} \right)^2,$$  \hspace{1cm} (4)

where $X_i$ is the pattern vector and $w[j, h], r[j, h], j = 1, \ldots, L, h = 1, \ldots, N$ are the weights between the inputs and hidden units, $N$ is the number of inputs, and $L$ is the number of hidden units. This type of function defines the metrics in the input space.

The hidden units’ output functions are of exponential type:

$$\phi'[X_i] = \exp (-\sigma'[X_i]).$$  \hspace{1cm} (5)

If we substitute (4) in (5), we obtain the unnormalized Gaussian function with a diagonal covariance
matrix. This type of function is the most used as the kernel function, and includes different variants ranging from radially symmetric type RBF [8] to more general RBF where the cross-correlation terms in the covariance matrix are not neglected [11]. Using two layers of weights for the input to hidden unit connections, the functions implemented by the hidden layer are more general, and in consequence we need a smaller number of hidden units. A multidimensional Gaussian can easily be decomposed in a product of Gaussians with lower dimensionality:

$$\phi'(X(1), \ldots, h, \ldots, N) = \phi'(X(1)) \cdot \ldots \cdot \phi'(X(h)) \cdot \ldots \cdot \phi'(X(N)).$$

(6)

The geometrical representation of this function (5) is an hyperellipsoid in the features space with the constraint that the principal axis is aligned with the features’ axis. According to this representation, the weights have a well defined meaning: \(w[j, h]\) corresponds to the center and \(r[j, h]\) to the shape parameter of the \(j\)th hyperellipsoid on the \(h\) axis. This type of function is similar with the potential function [7,11], where an equipotential line is the class boundary. If one pattern is situated near the center of a basis function, according to the given metric (4), the respective hidden unit will be activated; otherwise, the activation will decrease inversely proportional with the distance from the center:

$$\phi'[X_i] \rightarrow \begin{cases} 1 & \text{if } \{X_i \in C_i | \sigma[X_i] < \varepsilon\} \\ 0 & \text{if } \{X_i \not\in C_i | \sigma[X_i] > \varepsilon\} \end{cases}$$

(7)

The activation regions around the centers are determined by the parameter \(r[j, h]\). A small value for this parameter gives a localized region and a good recording for the respective patterns. On the other hand, a large value for it gives a large activation region and a better generalization.

The property of localization, gives to the basis functions the possibility to distinguish between different patterns. For two arbitrary patterns \(X_1, X_2 \in \Re^N\), one can find two functions \(\phi', \phi''\) such that \(\phi'[X_1] \neq \phi''[X_2]\). The hidden units represent an intermediate level of mapping between the input space and the outputs. They take the first decision in the classification of the patterns, grouping them into clusters. The clustering procedure is governed by the metrics used for the basis functions (4).

The activation functions for the outputs are weighted sums of hidden units outputs,

$$\sigma_k^h[X_i] = \sum_{j=1}^{L} \lambda[k, j] \phi'[X_i].$$

where \(k\) is the output unit, \(h = 1, \ldots, M\), and \(\lambda[k, j]\) is the weight between the hidden layer and the output layer. Geometrically, the weights \(\lambda[k, j]\) corresponds to the height of the \(j\)th basis function. Thus, the two-layer neural network maps an \(N\)-dimensional space (features space) into an \(N + 1\)-dimensional space where the \((N + 1)\)st dimension is weighted by the \(\lambda[k, j]\)'s. Each hypersurface of separation given by \(\sigma_k^h\) will enclose one class.

The output functions are of sigmoidal type:

$$Y[k, X_i] = \frac{1}{1 + \exp(-\sigma_k^h[X_i])}.$$  

(9)

The purpose of the output functions (5) and (9) is to limit the output values to the interval [0, 1]. In this case, we use continuous-valued outputs, but have boolean targets in which case we can only expect the outputs \(Y[k]\) to come within some margin of the targets. The desired outputs are codified:

$$F[k = 1, 0] \rightarrow \begin{cases} 1 & \text{if } X_i \in C_k \\ 0 & \text{if } X_i \not\in C_k \end{cases}$$

(10)

When an input pattern is presented to the network after the learning stage, only one output should be activated. We can easily determine the intersection between the boundary surfaces and the plane situated at equal distance by the target outputs. From (8),

$$\sigma_k^h[X_i] = 0 \rightarrow Y[k, X_i] = 0.5.$$  

(11)

Using this condition we can determine the intersection between the field of activation for an output and the half distance hyperplane. This condition can be used in order to decide the activation field for different outputs. However, this can generate uncertainty in the regions where there was not any given training pattern.

The decision for the class membership is given by the maximum output:

$$Y_i[X] = \text{Max} \{Y[k, X_i]\}, \quad k = 1, \ldots, M.$$  

(12)

This decision will ensure the partition of the input space between the different classes, where for each given pattern will be assigned only one class.

We give for hidden units an interpretation as subclasses which make up the output cases, where the output weights \(\lambda[k, j]\) show the contribution of this subclasses to the output classes. From (5) and (8), if \(\lambda[k, j] > 0\), then subclass \(j\) is included in class \(k\); otherwise, it is not. This partition, assumed unknown
initially, is obtained by the algorithm and depends on the selected type of RBF.

3. LEARNING ALGORITHM

In order to implement a neural network for a specific task, we have to know its structure and the corresponding weights. A learning rule allows us to find the associated parameters for a network by successive improvement from a starting point. To accomplish this task, we must have a cost function whose minimum gives the best reconstruction possible. In order to obtain a better approximation for the underlying set of functions, we use in the learning stage as a cost function the total squared error computed on all the patterns and network outputs from the training set,

$$E = \sum_{k=1}^{\mathcal{Q}} \sum_{i=1}^{N} (F_{k}(X_{i}) - Y_{k}(X_{i}))^2,$$  \hspace{1cm} (13)

where \( \mathcal{Q} \) is the total number of patterns in the training set. Using a global search for the learning algorithm makes it not easily fooled by local minima and gives a better approximation for the boundaries between the classes.

The learning process is halted when all the patterns from the learning set are correctly classified. This is the reason why the learning set should be consistent with the true distribution of data for the given task; otherwise, the decomposition in the RBF may not be consistent with the underlying function associated with the given task. However, using a global search for the minimum of the cost function in all the training batch alleviates the overfitting which can occur in the case when as many basis functions as patterns are used.

The training algorithm has two main parts: weights update and topology update. For weights as well as for the topology, we define proper initialization according procedures are used [8,12,14]. The widths of the RBF's are usually computed based on the Euclidean distances between the patterns.

For the network topology, we consider a number of inputs equal the number of features (dimensions of the input space) and a number of outputs equal the number of classes. We define initially a number of hidden units equal the number of outputs. Thus, we assume that each class has just one subclass. For the weights associated with the hidden units, we use such an initialization that their activation field is situated in the regions with data corresponding to each class.

The formulas derived for these parameters in order to cluster the data, according to the Gaussian activation function used (4), are: \( w[j, h] \) is the mean and \( r[j, h] \) is the variance of the data computed for each class \( h = 1, \ldots, M \) and dimension \( h = 1, \ldots, N \),

$$w[j, h] = \frac{\sum_{i=1}^{N} X_{i}[h]}{Q_k} \hspace{1cm} (14)$$

$$r[j, h] = 2 \cdot \frac{\sum_{i=1}^{N} (X_{i}[h] - w[j, h])^2}{Q_k} \hspace{1cm} (15)$$

for \( j = 1, \ldots, L \) \((L = M)\) and \( h = 1, \ldots, N \), and where \( Q_k \) is the number of patterns in class \( k \) in the learning set.

The output weights are initialized as

$$\lambda[k, j] = 0. \hspace{1cm} (16)$$

This initial value is a good starting point for a future polarization of output weights given by the updating algorithm (A.3) according to the membership or nonmembership of a subclass to an output class. The shape of each boundary between one class and the others is a constant hyperplane equidistant from the target outputs. Using this initialization according to (8, 11, 13) and (16) the first value for the cost function is well defined as

$$E = M \cdot Q \cdot 0.25. \hspace{1cm} (17)$$

This initialization represents a good starting point for convergence. The first value for the learning rate should be appropriate in order to decrease the cost function. The weights are updated to fit the optimal boundaries using the given number of hidden units (Section 3.2) or new neurons are added to the structure if their number is found insufficient (Section 3.3).

3.2. The Weights Update

Backpropagation (Rumelhart et al. [2]) is a currently used algorithm for weights updating in
feed-forward networks [1], and was also applied in some previous approaches for RBF weights updating [7,9,11]. The algorithm updates the weights to learn the training set of input-output pairs \( \{X_i, F_i, i = 1, \ldots, Q\} \). In order to use the backpropagation algorithm for a network, the functions corresponding to different layers should be differentiable. The derivatives of the cost function (13) with respect to the weights are used for updating. The updating expressions are

\[
w^{t+1} = w^t - \Delta w^t \tag{18}
\]

\[
\Delta w^t = \eta \frac{\partial E}{\partial w^t} + \alpha \Delta w^{t-1}. \tag{19}
\]

where \( w^t \) denotes the weights at learning iteration \( t \). First, \( \lambda(k, j)^{th} \) s, \( k = 1, \ldots, M \) and \( j = 1, \ldots, L \) are updated, then \( r(i, h)^{th} \) s and \( w(i, j)^{th} \) s, \( i = 1, \ldots, L, h = 1, \ldots, N \). The complete derivation of these formulas is given in Appendix A. The momentum \( \alpha \) should be between 0 and 1 and is used as a contribution from the previous step to each weight change in order to speed up the learning convergence [1]. The total number of adaptable parameters is in the beginning of learning \( M(2N + M) \). The updating expressions (A.4) and (A.8) constitute in fact a clustering procedure, where, in the learning stage, the weights take those values in order to determine the clustering of the patterns.

A structure with only one output is equivalent with one having two outputs, in the first case we have one boundary surface to delimit one class by the other, in the second case we have two boundaries, one for each class. In the last case, the two boundaries are complementary to each other, and from (A.3) it can be derived that for \( M = 2, \lambda(1, j) = -\lambda(2, j) \).

The learning rate \( \eta \) is identical for all weights. Its value is very important for the convergence and a suitable value for the learning rate in the beginning may not be so good later on. We have used a variable learning rate according to the variation of the cost function. If the cost function increases from one step to another, then the learning rate is decreased and the previous values for the weights are recovered:

\[
\begin{align*}
\text{if } E^t > E^{t-1} & \quad \left\{ \begin{array}{l}
\eta = \eta / 2 \\
 w^{t+1} = w^t + \Delta w^t.
\end{array} \right.
\end{align*}
\]

At the next step, the algorithm will try with a smaller learning rate. This procedure of learning rate update smoothes the learning curve and finds a suitable decrement for the weight update, avoiding the oscillations.

### 3.3. Topology Update

Classical approaches for a RBF implementation allocate as many hidden units as training patterns; or, the number of hidden units should be specified a priori, where each RBF center constitutes a clustering point. In these approaches, the number of hidden units used is usually too large. In [11], a hierarchically self-organizing algorithm was derived in order to find the optimal number of hidden units, where a new neuron is added to the network structure according to a saturation index of network performance.

In our approach we use an increasing architecture procedure, where the conditions of adding a new hidden unit are very simple. If too few hidden units are used at one moment in the learning time, the patterns which cannot be correctly classified with the given topology will tend to cause fluctuations in the cost function. Consequently, the learning rate will decrease (20) and \( E \) will not vary from one iteration to another,

\[
| E^t - E^{t-1} | < \varepsilon, \tag{21}
\]

where \( E^t \) is the cost function value obtained at iteration \( t \). However, if in this case \( E \) has a too large value to be acceptable and not all the patterns from the training set are correctly classified, a new neuron is added to the structure \( L \leftarrow L + 1 \). This neuron will be generated with the same number of inputs and outputs as the previous ones, so the number of adaptable parameters of the network will be increased by \( 2N + M \) (Fig. 1). The input space of this neuron should be defined in one region with incorrectly classified patterns from the same class. The same initialization as in the beginning of the learning algorithm (14)–(16) is used for the connections associated with this new neuron, except \( Q_a \) is now the number of incorrectly classified patterns from one of the classes. After a number of iterations, if the cost function is saturated and the new neuron has contributed to the correct

---

**FIG. 1.** The network structure and topology update procedure.
classification of some patterns in the training set, then all the patterns are checked and a new neuron is added if some patterns still remain unclassified. Using this procedure, all the clusters of patterns belonging to the same class are found \cite{22}, each of them make up the activation region of a new hidden unit.

The threshold \( c \) (21) is important for the topology updating procedure. If too large a threshold is used, then too many neurons will be generated, if it is too small, then the learning will be too slow.

In the case of spread patterns from one class through the patterns of the other classes this procedure does not necessarily yield a decrease in the cost function. In this case, the new neuron added to the structure does not contribute to the classification of new patterns (the patterns do not form compact groups). After condition (21) is fulfilled, the weights for the newly generated neuron are changed according to unclassified pattern \( i_{L+1} \),

\[
w[L + 1, h] = X_{i_{L+1}}[h]
\]

\[
(r[L + 1, h])^2 = \frac{1}{n} \min \left( \sum_{i=1}^{N} (X_{i_{L+1}}[h] - X_i[h])^2, \right)
\]

for \( i \neq i_{L+1} \),

for \( h = 1, \ldots, N \). The parameter \( n \) represents the overlapping factor between the activation field of the newly hidden unit with the existing units. In our simulations \( n = 4 \) was assumed, representing that the width of the new unit is equal with the half-distance between the unit center and the closest pattern. These formulae are used to center the new neuron on the pattern \( i_{L+1} \). However, in our case the patterns for which we set up this initialization are selected after some previous learning stage,

\[
\lambda[k, L + 1] = -m \sum_{j=1}^{L} \lambda[k, j] \phi'[X_{i_{L+1}}]
\]

for \( h = 1, \ldots, M \), where \( i_{L+1} \) is one unclassified pattern, and with \( 1 \leq m < e^{-7} \) \cite{20,21}. It can be shown (Appendix B) that after this initialization the pattern \( i_{L+1} \) will be correctly classified and this new neuron will not modify the class membership of the other patterns, contributing thus to the decreasing of the cost function. Geometrically, the boundary surface is transformed from convex to concave in the region around pattern \( i_{L+1} \). After each initialization, all the weights are updated using backpropagation (18)-(19) in order to obtain the optimal boundary with the given topology at one moment. As many neurons as necessary are added to correctly classify all patterns in the training set.

Finally, if at the end of the learning stage, there are too many hidden units, the units which have no patterns in their input field or have the same response for all patterns without a clear decision, are dismissed.

4. SIMULATION RESULTS

We present some examples where a RBF neural network with adaptive topology is employed for supervised pattern classification tasks. In the first part, the patterns are generated artificially, in the second part the network is applied to signal separation in a telecommunication problem and in the third part the network is used for speech recognition.

4.1. Simulated Patterns

Different configurations of patterns are used to test the network and comparisons with similar methods are provided. For the sake of an easy representation, the classification task is employed in a two-dimensional space.

**Example 1.** In the first example (Fig. 2), a configuration of four classes is presented. After learning, a 2-6-4 topology was found suitable for this example. As can be seen, all four classes have a common overlapping region in the center and constitute a difficult classification task. The boundaries given by \( \sigma^2 \) (8) which delimit two of the classes, obtained after learning, are presented in Figs. 3 and 4. In the first case can be identified three two-dimensional Gaussian functions and in the second two, each with two distinct activation regions. From the symmetry of this problem, the boundaries of the other two classes are simi-

![FIG. 2. The pattern configuration in Example 1, where the classes are marked differently with (O), (X), (+), and (+).](image)
lar with these. The learning curve is shown in Fig. 5. The smoothing of this curve is evident due to the adaptive learning rate (20) used. Oscillations occur only in the points where a new neuron is added to the structure; see Fig. 5.

**EXAMPLE 2.** In another example a nonlinear classification problem is presented. Forty patterns, 20 in each class, are considered. The classes have a high degree of mixture. The pattern configuration is shown in Fig. 6 as well as the decision region for one of the classes obtained after the learning with a RBF neural network. The contours of the Gaussian functions used for the classification can easily be identified in Fig. 6. Figure 7 presents for comparison the decision region for the same class obtained assuming as many basis functions as patterns, where for the basis functions was allowed to have different widths. The widths in the second case are placed at the half from the Euclidean distance to the closest pattern from the other class, assuming a certain degree of overlapping between the basis functions. The output weights are computed through the LMS. Note in these two plots that the decision regions of the two classes are similar even though the number of hidden units employed is quite different (10 in the first case and 40 in the second case). Comparison results between these two methods and a linear classifier are provided in the Table 1. For the linear classifier, a LMS algorithm with an appropriate stopping rule was used for the training. Using a piecewise linear approximation, in a network implementation, the result of the linear classifier can be improved; however, it is very difficult to obtain 100% of correctly classified patterns with an acceptable number of hidden units.
EXAMPLE 3. In the third example, two nonparametric estimation problems are evaluated. Here, we intend to infer the true probability distribution from a given set of two-dimensional samples. In the first example, we have to estimate three classes. The distributions are given by the relations:

- class A: \( N(3, 0; 1, 0.5) \cup N(-3, 0; 1, 0.5) \)
- class B: \( N(0, 3; 0.5, 1) \cup N(0, -3; 0.5, 1) \)
- class C: \( N(0, 0; 0.5, 0.5) \)

A total of 125 patterns were generated 50 from each of the first two classes and 25 from the third class. The algorithm presented in Section 3 was tested in comparison with the Moody and Darken algorithm [8] with a fixed topology. A number of five hidden units was found suitable for this algorithm which uses the \( k \)-means algorithm as a preprocessing stage. The results are shown in Table 2.

In the second estimation example, we have two classes generated by:

- class 1: \( N(0, 0; 0.3, 0.1) \cup N(1, 1; 0.3, 0.1) \)
- class 2: \( N(1, 0; 0.1, 0.3) \cup N(0, 1; 0.1, 0.3) \)

These distributions displayed in Fig. 8 represent a XOR problem where each input value is corrupted by noise. The result of the approximated bimodal, two-dimensional distribution for one of the classes is shown in Fig. 9. The other one is similar. For training we have used 200 samples.

From these results, one can see the good estimation properties of the RBPN in estimating different distributions. For testing purposes, we have used other data sets than those used for training. For the Moody and Darken algorithm, the number of iterations represent only the LMS algorithm, without considering the unsupervised preprocessing stage. The adaptive topology RBPN as presented in Section 3 has better results on the test set than the Moody and Darken algorithm because of the global estimation procedure, but requires longer training time.

4.2. Equalization for a Multiple Access Communication System

Equalization is used to compensate for the time dispersion introduced by the communication channel [19]. The communication channel distorts the symbols causing intersymbol interference in the channel including the effects of the transmitter filter, the modulator, the transmitter medium and the demodulator.

We consider a multipoint to point communication system where symbols from different users arrive simultaneously. In this system, the symbols are distorted by multipath propagation, which causes intersymbol interference, and by interfering signals from other sources, whose levels are also function of propagation conditions. The equalization in this case combines the diversity channels and reduce the interference from the same source symbols or different sources’ symbols, as much as possible to maximize the probability of correct decisions. The equalization can be viewed as an inverse filtering function where the equalizer parameters converge to those values which

<table>
<thead>
<tr>
<th>Method</th>
<th>Topology</th>
<th>Number of parameters</th>
<th>Cost function</th>
<th>Correctly classified patterns (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Adaptive topology RBF</td>
<td>2-10-1</td>
<td>50</td>
<td>0.126</td>
<td>100</td>
</tr>
<tr>
<td>Fixed RBF</td>
<td>2-40-1</td>
<td>160</td>
<td>0.032</td>
<td>100</td>
</tr>
<tr>
<td>Linear</td>
<td>2-1-1</td>
<td>3</td>
<td>8.526</td>
<td>67.5</td>
</tr>
</tbody>
</table>
TABLE 2

Comparison Results for Example 3

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Method</th>
<th>Topology</th>
<th>Number of parameters</th>
<th>Number of iterations</th>
<th>Test error (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Adaptive topology RBF</td>
<td>2-3-3</td>
<td>21</td>
<td>14,041</td>
<td>5</td>
</tr>
<tr>
<td></td>
<td>Moody and Darken</td>
<td>2-5-3</td>
<td>30</td>
<td>21,81</td>
<td>6.4</td>
</tr>
<tr>
<td>2</td>
<td>Adaptive topology RBF</td>
<td>2-4-1</td>
<td>20</td>
<td>80,148</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td>Moody and Darken</td>
<td>2-4-1</td>
<td>16</td>
<td>1,277</td>
<td>7.1</td>
</tr>
</tbody>
</table>

implement the inverse of the different access channels [24].

It was shown that RBF have good approximation capability of decision boundaries between symbols when intersymbol and cochannel interferences exist [20–22]. In [21], a two-stage algorithm is proposed to model the RBF approximation for the symbol detection, it was shown that this algorithm has better capability of symbol recovering than a linear equalizer. In the first stage the RBF centers are determined and in the second the output weights are trained using the LMS algorithm. However, the required complexity of such a structure in terms of the number of hidden units and adaptive parameters is too high. In [22], a solution is proposed for finding a smaller number of hidden units for a given channel structure when a dispersive channel is assumed.

In this application, we consider transmitters which send signals with the same carrier frequency through different channels Fig. 10. We consider multipath propagation on each channel and also cochannel interference. The received signal is given by

\[ X(t) = \sum_{k=1}^{K} \sum_{l=1}^{L} h_{kl} Y_{kl}(t - kT - \tau_{kl}) + n(t). \] (25)

where \( K \) is the number of users, \( L \) is the number of possible paths, \( T \) is the duration of signal interval, \( h_{kl} \) is the channel fading coefficient, \( \tau_{kl} \) is the delay that the \( k \)th signal experiences when propagating through the \( l \)th path, and \( n(t) \) is the additive noise. We want to decide the symbol and also the sender. In the training stage, we assume that the reference signals \( Y_1(t), Y_2(t), \ldots, Y_{KL}(t) \) from all the transmitters are available (Fig. 10).

We view the problem of channel equalization as a pattern classification problem where for each pair of symbol level and user is assigned a class. For \( m \) quantization levels, the number of classes will be \( m^K \). In our simulations, we have assumed binary phase shift keying modulated signals so the symbols are bipolar +1 and −1 which are sent through different channels. We consider that all the possible combinations of the symbols from all the transmitters are equally likely. In the general case, because of the intersymbol and

![FIG. 8. The data points representing a XOR corrupted by noise where each of the two bidimensional distributions is bimodal. The learning examples from each distribution are marked distinctly with (O) and (x).](image)

![FIG. 9. The estimated 2D bimodal distribution for one of the classes.](image)
TABLE 3
Comparison Results for the Equalization System in a Multiple Access Communication System

<table>
<thead>
<tr>
<th>( \alpha )</th>
<th>Network type</th>
<th>( E ) for the training set</th>
<th>Structure ( N-L-M )</th>
<th>Number of parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>Adaptive topology RBF</td>
<td>0.333</td>
<td>2-4-4</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>Fixed two-net RBF</td>
<td>0.653</td>
<td>2 × 2-32-1</td>
<td>196</td>
</tr>
<tr>
<td>0.4</td>
<td>Adaptive topology RBF</td>
<td>4.356</td>
<td>2-4-4</td>
<td>32</td>
</tr>
<tr>
<td></td>
<td>Fixed two-net RBF</td>
<td>0.654</td>
<td>2 × 2-32-1</td>
<td>196</td>
</tr>
<tr>
<td>0.6</td>
<td>Adaptive topology RBF</td>
<td>9.519</td>
<td>2-5-4</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>Fixed two-net RBF</td>
<td>0.654</td>
<td>2 × 2-32-1</td>
<td>196</td>
</tr>
<tr>
<td>0.8</td>
<td>Adaptive topology RBF</td>
<td>6.106</td>
<td>2-5-4</td>
<td>40</td>
</tr>
<tr>
<td></td>
<td>Fixed two-net RBF</td>
<td>0.650</td>
<td>2 × 2-32-1</td>
<td>196</td>
</tr>
</tbody>
</table>

Cochannel interferences, the decision boundaries between the symbols will be nonlinear and the received signals will cluster around the noise-free symbols for all the possible situations [21,22].

Using a RBF network, we define for the multichannel equalization task a compact structure where for each symbol—sender pair we assign an output. In the case of bipolar signals we have \( M = 2^K \) outputs. The number of network inputs is \( N \). For the symbol detection, we use the last \( N \) received samples, where \( N \) gives the order of the equalizer. For simulation purposes we have considered two users each with a two-path propagation direction.

\[
X(t) = \alpha(Y_1(t) + 0.5Y_4(t - 1)) + (1 - \alpha)(Y_5(t) + 0.2Y_2(t - 1)) + n(t), \quad (26)
\]

where the energies of the \( K \) transmitted signals are assumed to be equal, \( \alpha \) measures different signal-to-interference ratios, the additive noise is assumed to be white and Gaussian, and each channel has two taps delay. This two-channel system is similar with that used in [21] where the equalization was used only for the first channel.

The RBF network is applied to recover the transmitted signals. We have considered two tap delays, only the last two symbols are used for the signal prediction. We can define four classes, for all possible combinations of signals from different classes:

\[
\begin{align*}
H_1 &= \{ X | Y_1(t) = 1, \ Y_2(t) = 1 \} \\
H_2 &= \{ X | Y_1(t) = 1, \ Y_2(t) = -1 \} \\
H_3 &= \{ X | Y_1(t) = -1, \ Y_2(t) = 1 \} \\
H_4 &= \{ X | Y_1(t) = -1, \ Y_2(t) = -1 \}.
\end{align*}
\]

For training, 320 samples are used with a signal-to-noise ratio of 20 dB. From these, a \( K \)-means algorithm is used to select 64 centers which approximate the error-free states. For different cochannel interferences tested, each produces a distinct distribution of the data.

For comparison purposes we have defined a structure with a number of subnets equal to the number of users, where each network is used for detecting one user. The inputs are common for all these subnets and each has only one output. We have trained two RBF neural networks, with a fixed topology one for each user where the other user is assumed to be undesirable interference. Two networks with a structure 2-32-1, denoted compactly as 2 × 32-1, are used. These networks were trained through a two stages learning algorithm similar with that used in [21] and where the centers are found in the first stage using an adaptive \( k \)-means algorithm. Like in [21] we have used the same width for all the basis functions and the output weights are trained using the LMS algorithm. The training time for this structure is shorter comparing with our algorithm at the expense of a more complex structure. Comparative results in the structure and number of necessary parameters are given in Table 3.

A comparison between these two methods for different signal-to-noise ratios is presented in Fig. 11.
From the results in Fig. 9, one can infer that both structures have similar memorization properties, but they differ in the generalization capabilities. For \( a = 0.4 \) and 0.6, the RBF with adaptive topology perform better than the fixed topology RBF for smaller signal-to-noise ratio, because they have better generalization properties but not in the cases \( a = 0.2 \) and 0.8. However, the complexity obtained in the number of parameters and hidden units is 5-6 times less for the adaptive structure.

### 4.3. Static Speech Classification

Much work in applying neural networks to speech recognition has focused on phoneme recognition. RBF networks were used and successfully learned to form nonlinear classification regions for vowel classes [7,8,17]. A benchmark used for classification purposes is that measured by Peterson and Barney (1952) in a study in which they evaluated the formant frequencies from different American speakers subjects.

In [81], Moody and Darken used a RBF two-stage algorithm for classifying 10 distinct vowel sounds on the basis of the first and second formant frequencies. They have obtained an error rate of 26.7% for 20 units and 18% using 100 units. In [7], for the training was chosen a subset of the data by clustering each class separately, to retain the samples that were maximally spread. For the RBF, they assumed the same number of units as data they have and each class was trained independently (eight times for eight classes). In [17], an architecturally dynamic network was used for the Peterson-Barney vowels in an on-line classification task. This algorithm finds the number of hidden units, using the consistency of the present data with that of the past from which it has already learned a representation. Using 85 hidden units, they obtained a misclassification error of 24%.

To test our algorithm we use a subset of the Peterson and Barney data containing the first two formants frequencies for 9 spoken vowels from 38 male speakers, each with 2 occurrences for the same vowel. For each vowel, we define a class, and each formant frequency corresponds to a feature in the input space. For training purpose we used 90 vectors (10 for each class) representing the formants frequencies from 5 speakers, and for testing, 504 vectors from other 28 speakers. We have used the following phonemes: Y, I, H, EH, AE, AH, AO, UH, UW, ER. The results obtained compared with the nearest neighbor and the Moody and Darker algorithm are presented in Table 4. A topology of 2-9-9, where each class has only one corresponding hidden unit, was found suitable for this problem. For the Moody and Darken algorithm [8], we have used the same topology with 9 hidden units and 9 output units, and we have applied the k-means algorithm for detecting the centers of the basis functions. All the algorithms use the same training data (which are stored for nearest neighbor). The nearest neighbor method gives better results as known from some other studies [8,17]. However it requires a longer time to find the best classification for a given input pattern and it needs to store the entire set of data.
The decision regions formed by different classes are shown in Fig. 12 for the Gaussian network as well as the vectors used for training.

5. CONCLUSIONS

In the first part of this paper, we have presented a brief overview of different trends in radial basis functions neural networks and their applications. The success of a classification scheme depends on the geometrical properties of the pattern classes under consideration. Different pattern classification schemes are suitable in different situations and may not work properly in others. We give the meanings for the weights and processing units for a RBF network applied for pattern classification.

A new learning algorithm for the RBF neural network is presented in this paper. This algorithm gives a solution for classifying configurations of patterns in a feature space providing the minimum number of hidden units for the network implementation. The learning is based on backpropagation algorithm where the conditions of inserting new neurons in the structure as well as the initializations of the weights are given. However, the algorithm requires a long time of training in order to find a suitable structure.

The algorithm was tested on different artificial and real applications. It was successfully applied for the distribution estimation, as well as for signal separation in a multiple access communication system and for static speech recognition.

APPENDIX A

The weights updating relations are derived here. The backpropagation algorithm was applied for all the weights starting with the last layer weights and ending with the inputs weights. The cost function assumed is the total square error (13).

Let \( F_i[k] = F_i[X_i] \) and \( Y_i[k] = Y_i[X_i] \). For \( \lambda(k,j)k = 1, \ldots, M \) and \( j = 1, \ldots, L \),

\[
\frac{\partial E}{\partial \lambda[k,j]} = (-2) \sum_{i=1}^{Q} (F_i[k] - Y_i[k]) \frac{\partial}{\partial \lambda[k,j]} Y_i[k]. \tag{A.1}
\]

Using (8) and (9),

\[
\frac{\partial}{\partial \lambda[k,j]} Y_i[k] = (1 - Y_i[k]) Y_i[k] \phi'[X_i] \tag{A.2}
\]

\[
\frac{\partial E}{\partial \lambda[k,j]} = \sum_{i=1}^{Q} (-2) Y_i[k] (1 - Y_i[k]) \times (F_i[k] - Y_i[k]) \lambda[k,j] \frac{\partial}{\partial \lambda[k,j]} Y_i[k]. \tag{A.3}
\]

For \( r[j,h]j = 1, \ldots, L \) and \( h = 1, \ldots, N \) similar to (A.3):

\[
\frac{\partial E}{\partial r[j,h]} = \sum_{k=1}^{M} \sum_{i=1}^{Q} (-2) Y_i[k] (1 - Y_i[k]) \times (F_i[k] - Y_i[k]) \lambda[k,j] \frac{\partial}{\partial r[j,h]} Y_i[k]. \tag{A.4}
\]

From (4) and (5),

\[
\frac{\partial}{\partial r[j,h]} Y_i[k] = \phi'[X_i] (w[j,h] - X_i[h])^2 \tag{A.5}
\]

\[
\frac{\partial E}{\partial r[j,h]} = \sum_{k=1}^{M} \sum_{i=1}^{Q} (-4) Y_i[k] (1 - Y_i[k]) \times (F_i[k] - Y_i[k]) \lambda[k,j] \phi'[X_i] \times (w[j,h] - X_i[h])^2 \tag{A.6}
\]

For \( w[j,h]j = 1, \ldots, L \) and \( h = 1, \ldots, N \),

\[
\frac{\partial}{\partial w[j,h]} \phi'[X_i] = 2 \frac{w[j,h] - X_i[h]}{(r[j,h])^2}. \tag{A.7}
\]
and similar to (A.6):

$$
\frac{\partial E}{\partial w[j, h]} = \sum_{k=1}^{K} \sum_{i=1}^{q} 4Y_i[k](1 - Y_i[k]) \times (F[k] - Y_i[k] \cdot \lambda[k, j] \phi'[X_i]) \times w[j, k] - X_i[h] (r[j, h])^2 .
$$

(A.8)

**APPENDIX B**

We consider the initialization for the new neuron weights according to (22), (23), and (24). We will prove that fixing this neuron on one undetermined pattern at the iteration $t$, it will be classified correctly at the next iteration $t + 1$:

$$
\sigma[k][X_{i, i+1}]^t = \sum_{j=1}^{L} \lambda[k, j] \phi'[X_{i, i+1}] = 0 \Rightarrow \phi'[X_{i, i+1}] = 1 .
$$

(B.1)

From (4), (5), and (22) we obtain

$$
\sigma[k][X_{i, i+1}]^t = \sum_{j=1}^{L} \lambda[k, j] \phi'[X_{i, i+1}] = 0 \Rightarrow \phi'[X_{i, i+1}] = 1 .
$$

(B.2)

Using (24), we will obtain a new value for the output unit activation function, and using (8) and (B.2) the pattern $X_{i, i+1}$ will be correctly classified:

$$
\sigma[k][X_{i, i+1}]^t = \sum_{j=1}^{L} \lambda[k, j] \phi'[X_{i, i+1}] = (1 - m) \sum_{j=1}^{L} \lambda[k, j] \phi'[X_{i, i+1}] = 0 .
$$

(B.3)

For $m > 1$ this change will modify the convexity of the separation boundary between class $k$ and the other classes in the region of the pattern $X_{i, i+1}$.

We will show that the other patterns correctly classified will not be modified by this procedure. Thus we assume at iteration $t$

$$
\sigma[k][X_i]^t - \sum_{j=1}^{L} \lambda[k, j] \phi'[X_i] = 0 .
$$

(B.4)

For the pattern $X_i$, with $i \neq i_{i+1}$ and from a different class than for $X_{i, i+1}$, then from (4), (5), (22), and (23) we obtain $\phi'[X_i] < \epsilon^{-n}$. Using (24) and (8), we obtain

$$
\sigma[k][X_i]^{i+1} = (1 - \epsilon^{-n}) \sum_{i=1}^{L} \lambda[k, i] \phi'[X_i] .
$$

(B.5)

and for $\epsilon^{-n} < 1$, we have

$$
\text{sgn}(\sigma[k][X_i]^{i+1}) = \text{sgn}(\sigma[k][X_i]^t) .
$$

(B.6)

As a consequence, the class membership (B.4) for the $i$th pattern will be unchanged, where the sign of $\sigma[k][X_i]^t$ gives the decision of the class membership (11). The bounds for $m$ are

$$
1 < m < \epsilon^{-n} .
$$

(B.7)

**REFERENCES**


ADRIAN G. BORS received the M.S. degree in electronics from the Polytechnic University of Bucharest, Bucharest, Romania, in 1992. His main interests lie in the areas of neural networks, nonlinear digital signal processing, and pattern recognition.

MONCEF GABBOUJ received the B.S. degree in electrical engineering in 1985 from Oklahoma State University, Stillwater. He received the M.S. and Ph.D. degrees in electrical engineering from Purdue University, West Lafayette, IN, in 1986 and 1989. From 1990 to 1992, he has been a senior research scientist with the Research Institute for Information Technology, Tampere, Finland. He is currently associate professor at the Signal Processing Laboratory at Tampere University of Technology, Tampere, Finland. His research interests include nonlinear signal and image processing, mathematical morphology, neural networks, and artificial intelligence. Dr. Gabbouj is the Director of the International University Program in Digital Signal Processing at the Signal Processing Laboratory at Tampere University of Technology. He is a member ofEta Kappa Nu, Phi Kappa Phi, and IEEE ASSP and CS societies. He is the Chairman-Elect of the IEEE Circuits and Systems Society Technical Committee on Digital Signal Processing. Dr. Gabbouj is a co-recipient of the Myril B. Reed Best Paper Award from the 29th Midwest Symposium on Circuits and Systems.