Gradient Descent Learning of Radial Basis Neural Networks

Nicolaos B. Karayiannis
Department of Electrical & Computer Engineering
University of Houston
Houston, Texas 77204-4793, USA
Karayiannis@UH.EDU

Abstract

This paper presents an axiomatic approach for building RBF neural networks and also proposes a supervised learning algorithm based on gradient descent for their training. This approach results in a broad variety of admissible RBF models, including those employing Gaussian radial basis functions. The form of the radial basis functions is determined by a generator function. A sensitivity analysis explains the failure of gradient descent learning on RBF networks with Gaussian radial basis functions, which are generated by an exponential generator function. The same analysis verifies that RBF networks generated by a linear generator function are much more suitable for gradient descent learning. Experiments involving such RBF networks indicate that the proposed gradient descent algorithm guarantees fast learning and very satisfactory generalization ability.

1. Introduction

A *radial basis function* (RBF) neural network is usually trained to map a vector $\mathbf{x}_k \in \mathbb{R}^{n_i}$ into a vector $\mathbf{y}_k \in \mathbb{R}^{n_o}$, where the pairs $(\mathbf{x}_k, \mathbf{y}_k), 1 \leq k \leq M$, form the training set. If this mapping is viewed as a function in the input space $\mathbb{R}^{n_i}$, learning can be seen as a function approximation problem. According to this point of view, learning is equivalent to finding a surface in a multidimensional space that provides the best fit to the training data. Generalization is therefore synonymous with interpolation between the data points along the constrained surface generated by the fitting procedure as the optimum approximation to this mapping.

The performance of an RBF network depends on the number and positions of the radial basis functions, their shape, and the method used for learning the input-output mapping. The existing learning strategies for RBF neural networks can be classified as follows: (i) strategies selecting the radial basis function centers randomly from the training data [1], (ii) strategies employing unsupervised procedures for selecting the radial basis function centers [5], and (iii) strategies employing supervised procedures for selecting the radial basis function centers [7].

It is widely believed that the choice of radial basis functions is not critical to the performance of RBF networks. However, it is the selection of Gaussian basis functions that dictated the development of the current learning schemes for RBF neural networks. The common characteristic of the majority of the existing learning schemes is that the prototypes are updated separately according to some (usually unsupervised) clustering algorithm while the weights of the upper associative network are updated by a supervised procedure to implement the desired mapping. In conclusion, the use of Gaussian basis functions prevented the development of a simple and easily implementable learning algorithm based on gradient descent such as the error back propagation algorithm that is one of the main reasons behind the popularity of conventional feed-forward neural networks.

2. From Learning Vector Quantization to Radial Basis Function Neural Networks

Consider the finite set of feature vectors $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_M\} \subset \mathbb{R}^{n_i}$, which are represented by the set of prototypes $\mathcal{V} = \{\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_c\} \subset \mathbb{R}^{n_i}$. Clustering algorithms are frequently developed to solve a constrained minimization problem using alternating optimization [4]. This approach results in necessary conditions for the two sets of unknowns involved in the objective function minimized, that is, the prototypes representing the feature vectors and the membership func-

0-7803-4122-8/97 $10.00©1997 IEEE

tions that assign the feature vectors to the prototypes. Learning vector quantization and clustering algorithms can alternatively be developed by using gradient descent to minimize a *reformulation function*, that is, a function of the prototypes obtained by substituting the optimal set of membership functions in the original objective function [3, 4]. The development of learning vector quantization and clustering algorithms reduces to the construction of admissible reformulation functions [2]. Consider the mapping $\mathbb{R}^{n_i} \rightarrow \mathbb{R}$ described by

$$\hat{y}_k = f(S_k) = f\left(\frac{1}{c}\sum_{j=1}^{c} g\left(\|\mathbf{x}_k - \mathbf{v}_j\|^2\right)\right), \; 1 \leq k \leq M, \tag{1}$$

where $f(x)$ and $g(x)$ are differentiable everywhere functions of $x \in (0, \infty)$ and $\|\cdot\|$ denotes the Euclidean norm. Suppose $f(\cdot)$ and $g(\cdot)$ satisfy the condition $f(g(x)) = x$ and let

$$R = \frac{1}{M}\sum_{k=1}^{M} \hat{y}_k = \frac{1}{M}\sum_{k=1}^{M} f\left(\frac{1}{c}\sum_{j=1}^{c} g\left(\|\mathbf{x}_k - \mathbf{v}_j\|^2\right)\right). \tag{2}$$

Then:

- R is an admissible reformulation function of the first kind if $f(x)$ and $g(x)$ are both monotonically decreasing functions of $x \in (0, \infty)$ and $g'(x)$ is a monotonically increasing function of $x \in (0, \infty)$, which also implies that $f'(x)$ is an increasing function of $x \in (0, \infty)$.
- R is an admissible reformulation function of the second kind if $f(x)$ and $g(x)$ are both monotonically increasing functions of $x \in (0, \infty)$ and $g'(x)$ is a monotonically decreasing function of $x \in (0, \infty)$, which also implies that $f'(x)$ is an increasing function of $x \in (0, \infty)$.

Minimization of R using gradient descent leads to unsupervised learning vector quantization algorithms that can be used to determine the prototypes [2]. A broad variety of learning vector quantization and clustering algorithms were developed using functions of the form $g(x) = (g_0(x))^{\frac{1}{1-m}}, m \neq 1$, where $g_0(\cdot)$ is an admissible *generator function*, i.e., a function that satisfies certain properties and leads to admissible reformulation functions [2].

If $f(x) \neq g^{-1}(x)$, the reformulation function R can lead to supervised learning models. If $f(\cdot)$ is a linear function of the form $f(x) = a\,x + b$, (1) gives

$$\hat{y}_k = w_0 + \sum_{j=1}^{c} w_j\, g\left(\|\mathbf{x}_k - \mathbf{v}_j\|^2\right), \; 1 \leq k \leq M, \tag{3}$$

where $w_0 = b$ and $w_j = \frac{a}{c}$, $1 \leq j \leq c$. The model (3) can be used for function approximation if the prototypes $\mathbf{v}_j, 1 \leq j \leq c$, and the weights $w_j, 0 \leq j \leq c$, are adjustable parameters. The adjustable parameters of the model

$$\hat{y} = w_0 + \sum_{j=1}^{c} w_j\, g\left(\|\mathbf{x} - \mathbf{v}_j\|^2\right) \tag{4}$$

can be determined so that (4) implements a desired mapping $\mathbb{R}^{n_i} \rightarrow \mathbb{R}$ specified by the training set $(y_k, \mathbf{x}_k), 1 \leq k \leq M$. The model (4) can be extended to implement any mapping $\mathbb{R}^{n_i} \rightarrow \mathbb{R}^{n_o}$, $n_o \geq 1$, as

$$\hat{y}_i = w_{i0} + \sum_{j=1}^{c} w_{ij}\, g\left(\|\mathbf{x} - \mathbf{v}_j\|^2\right), \; 1 \leq i \leq n_o. \tag{5}$$

The model (5) describes an RBF neural network with inputs from $\mathbb{R}^{n_i}$, c radial basis function units, and n_o linear output units if $g(x^2) = \phi(x)$, and $\phi(\cdot)$ is a radial basis function. In such a case, the response of the network to the input vector $\mathbf{x}_k$ is

$$\hat{y}_{i,k} = \sum_{j=0}^{c} w_{ij}\, h_{j,k}, \; 1 \leq i \leq n_o, \tag{6}$$

where $h_{0,k} = 1, \forall k$, and $h_{j,k} = \phi\left(\|\mathbf{x}_k - \mathbf{v}_j\|\right) = g\left(\|\mathbf{x}_k - \mathbf{v}_j\|^2\right), 1 \leq j \leq c$. The response (6) of the RBF neural network to the input $\mathbf{x}_k$ is actually the output of the upper associative network. When the RBF network is presented with $\mathbf{x}_k$, the input of the upper associative network is formed by the responses $h_{j,k}$ of the radial basis functions located at the prototypes $\mathbf{v}_j, 1 \leq j \leq c$.

An RBF neural network is often interpreted as a composition of localized receptive fields. The locations of these receptive fields are determined by the prototypes while their shape is determined by the radial basis functions used. This interpretation of RBF neural networks requires that the responses of all radial basis functions to all inputs are always positive. If the prototypes are interpreted as the centers of receptive fields, it is required that the response of any radial basis function becomes stronger as the input approaches its corresponding prototype. Finally, it is required that the response of any radial basis function becomes more sensitive to changes in an input vector as the input vector approaches its corresponding prototype.

Let $h_{j,k} = g\left(\|\mathbf{x}_k - \mathbf{v}_j\|^2\right)$ for all $\mathbf{x}_k \in \mathcal{X}$ and $\mathbf{v}_j \in \mathcal{V}$. According to the above interpretation of RBF neural networks, any admissible radial basis function $\phi(x) = g(x^2)$ must satisfy the following three basic axiomatic requirements:

1. $h_{j,k} > 0$ for all $\mathbf{x}_k \in \mathcal{X}$ and $\mathbf{v}_j \in \mathcal{V}$.
2. $h_{j,k} > h_{j,\ell}$ for all $\mathbf{x}_k, \mathbf{x}_\ell \in \mathcal{X}$ and $\mathbf{v}_j \in \mathcal{V}$ such that $\|\mathbf{x}_k - \mathbf{v}_j\|^2 < \|\mathbf{x}_\ell - \mathbf{v}_j\|^2$.
3. If $\nabla_{\mathbf{x}_k} h_{j,k} = \partial h_{j,k}/\partial \mathbf{x}_k$ denotes the gradient of $h_{j,k}$ with respect to $\mathbf{x}_k$, then

$$\frac{\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2}{\|\mathbf{x}_k - \mathbf{v}_j\|^2} > \frac{\|\nabla_{\mathbf{x}_\ell} h_{j,\ell}\|^2}{\|\mathbf{x}_\ell - \mathbf{v}_j\|^2},$$

for all $\mathbf{x}_k, \mathbf{x}_\ell \in \mathcal{X}$ and $\mathbf{v}_j \in \mathcal{V}$ such that $\|\mathbf{x}_k - \mathbf{v}_j\|^2 < \|\mathbf{x}_\ell - \mathbf{v}_j\|^2$.

The selection of radial basis functions in accordance with the three basic axiomatic requirements can be facilitated by the following Theorem:

Theorem 1: The model described by (5) represents an RBF neural network in accordance with the three basic axiomatic requirements iff: (i) $g(x) > 0, \forall x \in (0,\infty)$, (ii) $g(x)$ is a monotonically decreasing function of $x \in (0,\infty)$, and (iii) $g'(x)$ is a monotonically increasing function of $x \in (0,\infty)$.

Since $\phi(x) = g(x^2)$, the Gaussian radial basis function $\phi(x) = \exp(-x^2/\sigma^2)$ corresponds to $g(x) = \exp(-x/\sigma^2)$. For any $\beta > 0$, there exists an $m > 1$ such that $\sigma^2 = (m-1)/\beta$. Thus, $g(x)$ can be written in terms of β and $m > 1$ as $g(x) = (\exp(\beta x))^{\frac{1}{1-m}}$. For any $\beta > 0$, there also exists an $m < 1$ such that $\sigma^2 = (1-m)/\beta$. Thus, $g(x)$ can be also written in terms of β and $m < 1$ as $g(x) = (\exp(-\beta x))^{\frac{1}{1-m}}$. This is an indication that $g(x)$ can be defined in terms of an increasing or decreasing generator function $g_0(x)$ as $g(x) = (g_0(x))^{\frac{1}{1-m}}, m \neq 1$. The selection of generator functions $g_0(\cdot)$ that lead to admissible radial basis functions of this form can be facilitated by the following Theorem:

Theorem 2: Consider the model (5) and let $g(x)$ be defined in terms of the generator function $g_0(x) > 0, \forall x \in (0,\infty)$, as $g(x) = (g_0(x))^{\frac{1}{1-m}}, m \neq 1$. Then, this model represents an RBF neural network in accordance with the three basic axiomatic requirements if: (i) $m > 1$, $g_0(x)$ is a monotonically increasing function of $x \in (0,\infty)$, and

$$\frac{m}{m-1}(g_0'(x))^2 > g_0(x)\, g_0''(x), \forall x \in (0,\infty), \quad (7)$$

(ii) $m < 1$, $g_0(x)$ is a monotonically decreasing function of $x \in (0,\infty)$, and

$$\frac{m}{m-1}(g_0'(x))^2 < g_0(x)\, g_0''(x), \forall x \in (0,\infty). \quad (8)$$

Consider the increasing generator function $g_0(x) = x > 0, \forall x \in (0,\infty)$. According to Theorem 2, the corresponding function $g(x) = (g_0(x))^{\frac{1}{1-m}}$ satisfies the three basic axiomatic requirements for all $m > 1$. If $g_0(x) = x$, then $g(x) = x^{\frac{1}{1-m}}$ corresponds to $\phi(x) = g(x^2) = x^{\frac{2}{1-m}}, m > 1$. For $m = 3 > 1$, $g(x) = x^{-\frac{1}{2}}$ and $\phi(x) = x^{-1}$. Consider also the decreasing generator function $g_0(x) = x^{-1} > 0, \forall x \in (0,\infty)$. According to Theorem 2, the corresponding function $g(x) = (g_0(x))^{\frac{1}{1-m}}$ satisfies the three basic axiomatic requirements for all $m < 1$. If $g_0(x) = x^{-1}$, then $g(x) = (x^{-1})^{\frac{1}{1-m}} = x^{\frac{1}{m-1}}$ corresponds to $\phi(x) = g(x^2) = x^{\frac{2}{m-1}}, m < 1$. For $m = -1 < 1$, $g(x) = x^{-\frac{1}{2}}$ and $\phi(x) = x^{-1}$. The function $\phi(\cdot)$ corresponding to the increasing generator function $g_0(x) = x$ with $m = 3$ and the decreasing generator function $g_0(x) = x^{-1}$ with $m = -1$ can be interpreted as the limit of the inverse multiquadratic radial basis function $\phi(x) = (x^2+c^2)^{-\frac{1}{2}}$ at $c = 0$.

The inverse multiquadratic function $\phi(x) = (x^2 + c^2)^{-\frac{1}{2}}$ is bounded at $x = 0$ because of the nonzero constant c. If $g(x) = (g_0(x))^{\frac{1}{1-m}}$, then the functions $\phi(x) = g(x^2)$ that correspond to the increasing generator function $g_0(x) = x$ with $m > 1$ and the decreasing generator function $g_0(x) = x^{-1}$ with $m < 1$ are not bounded at $x = 0$. If necessary, this problem can be fixed by replacing the generator functions $g_0(x) = x$ and $g_0(x) = x^{-1}$ by $g_0(x) = x + c^2$ and $g_0(x) = (x + c^2)^{-1}$, respectively, with $c \neq 0$.

3. Learning the Input-Output Mapping Using Gradient Descent

An RBF neural network can be trained by minimizing the error

$$E = \frac{1}{2}\sum_{k=1}^{M}\sum_{i=1}^{n_o}(y_{i,k} - \hat{y}_{i,k})^2, \quad (9)$$

where $\hat{y}_{i,k}$ is the actual response of the ith output unit to $\mathbf{x}_k$ and $y_{i,k}$ is the desired response. If the output units are linear, then $\hat{y}_{i,k} = \sum_{j=0}^{c} w_{ij}\, h_{j,k}$, where $h_{0,k} = 1$, and $h_{j,k} = g(\|\mathbf{x}_k - \mathbf{v}_j\|^2), 1 \leq j \leq c$.

The update equation for the weights of the upper network can be obtained using gradient descent as

$$\Delta w_{pq} = -\eta \frac{\partial E}{\partial w_{pq}} = \eta \sum_{k=1}^{M} \varepsilon_{p,k}^{o}\, h_{q,k}, \quad (10)$$

where $\varepsilon_{p,k}^{o} = y_{p,k} - \hat{y}_{p,k}$ and η is the learning rate. The update equation for the prototypes can be obtained using gradient descent as

$$\Delta \mathbf{v}_q = -\eta \frac{\partial E}{\partial \mathbf{v}_q} = \eta \sum_{k=1}^{M} \varepsilon_{q,k}^{h}(\mathbf{x}_k - \mathbf{v}_q), \quad (11)$$

where

$$\varepsilon^h_{q,k} = \alpha_{q,k} \sum_{i=1}^{n_o} \varepsilon^o_{i,k}\, w_{iq}, \tag{12}$$

and $\alpha_{q,k} = -2\, g'(\|\mathbf{x}_k - \mathbf{v}_q\|^2)$. If $g(x) = (g_0(x))^{\frac{1}{1-m}}$, then

$$\alpha_{q,k} = \frac{2}{m-1} (h_{q,k})^m\, g_0' \left(\|\mathbf{x}_k - \mathbf{v}_q\|^2\right). \tag{13}$$

If $g_0(x) = \exp(\beta x)$, then $g_0'(\|\mathbf{x}_k - \mathbf{v}_q\|^2) = \beta\,(h_{q,k})^{1-m}$ and (13) gives $\alpha_{q,k} = 2\beta/(m-1)\,h_{q,k}$. If $g_0(x) = x$, then $g_0'(x) = 1$ and (13) gives $\alpha_{q,k} = 2/(m-1)\,(h_{q,k})^m$.

This algorithm can be summarized as follows:

1. Select m, η and ϵ; initialize $\{w_{ij}\}$ with zero values; randomly initialize the prototypes $\mathbf{v}_j, 1 \le j \le c$; set $h_{0,k} = 1, \forall k$.
2. Compute the initial response:
   - $h_{j,k} = \left(g_0\left(\|\mathbf{x}_k - \mathbf{v}_j\|^2\right)\right)^{\frac{1}{1-m}}, \forall j,k.$
   - $\hat{y}_{i,k} = \sum_{j=0}^{c} w_{ij}\, h_{j,k}, \forall i,k.$
3. Compute $E = \frac{1}{2}\sum_{k=1}^{M}\sum_{i=1}^{n_o}(y_{i,k} - \hat{y}_{i,k})^2$.
4. Set $E_{\text{old}} = E$.
5. Update the adjustable parameters:
   - $\varepsilon^o_{i,k} = y_{i,k} - \hat{y}_{i,k}$.
   - $w_{ij} \leftarrow w_{ij} + \eta \sum_{k=1}^{M} \varepsilon^o_{i,k}\, h_{j,k}$.
   - $\alpha_{j,k} = \frac{2}{m-1}\,(h_{j,k})^m\, g_0'\left(\|\mathbf{x}_k - \mathbf{v}_j\|^2\right)$.
   - $\varepsilon^h_{j,k} = \alpha_{j,k} \sum_{i=1}^{n_o} \varepsilon^o_{i,k}\, w_{ij}$.
   - $\mathbf{v}_j \leftarrow \mathbf{v}_j + \eta \sum_{k=1}^{M} \varepsilon^h_{j,k}\,(\mathbf{x}_k - \mathbf{v}_j)$.
6. Compute the current response:
   - $h_{j,k} = \left(g_0\left(\|\mathbf{x}_k - \mathbf{v}_j\|^2\right)\right)^{\frac{1}{1-m}}, \forall j,k.$
   - $\hat{y}_{i,k} = \sum_{j=0}^{c} w_{ij}\, h_{j,k}, \forall i,k.$
7. Compute $E = \frac{1}{2}\sum_{k=1}^{M}\sum_{i=1}^{n_o}(y_{i,k} - \hat{y}_{i,k})^2$.
8. if: $(E_{\text{old}} - E)/E_{\text{old}} > \epsilon$; then: go to 4.

4. Sensitivity Analysis of Gradient Descent Learning

The gradient-descent-based learning algorithm developed above attempts to train the network to implement a desired input-output mapping by producing incremental changes of the weights of the upper network and the prototypes. If the responses of the radial basis functions are not substantially affected by incremental changes of the prototypes, then the learning process reduces to incremental changes of the weights of the upper associative network which alone are unlikely to implement non-trivial input-output mappings. The ability of the network to implement a desired input-output mapping depends very strongly on the sensitivity of the responses of the radial basis functions to incremental changes of their corresponding prototypes. The sensitivity of the response $h_{j,k}$ of the jth radial basis function to changes of its corresponding prototype $\mathbf{v}_j$ can be measured by

$$S_{\mathbf{v}_j} h_{j,k} = \frac{\|\nabla_{\mathbf{v}_j} h_{j,k}\|^2}{\|\mathbf{x}_k - \mathbf{v}_j\|^2}. \tag{14}$$

If $g(x) = (g_0(x))^{\frac{1}{1-m}}$ with $g_0(x) = \exp(\beta x)$, then

$$S_{\mathbf{v}_j} h_{j,k} = \left(\frac{2}{\sigma^2}\right)^2 \exp\left(-2\frac{\|\mathbf{x}_k - \mathbf{v}_j\|^2}{\sigma^2}\right). \tag{15}$$

If $g(x) = (g_0(x))^{\frac{1}{1-m}}$ with $g_0(x) = x$, then

$$S_{\mathbf{v}_j} h_{j,k} = \left(\frac{2}{m-1}\right)^2 \left(\frac{1}{\|\mathbf{x}_k - \mathbf{v}_j\|^2}\right)^{\frac{2m}{m-1}}. \tag{16}$$

The behavior of the sensitivity measure (14) depends on the selection of the generator function. For $g_0(x) = x$, $S_{\mathbf{v}_j} h_{j,k}$ is significantly affected by even insignificant perturbations of $\mathbf{v}_j$ especially if $\|\mathbf{x}_k - \mathbf{v}_j\|^2 \le 1$. Since $m > 1$, $S_{\mathbf{v}_j} h_{j,k}$ increases as the distance between $\mathbf{v}_j$ and $\mathbf{x}_k$ decreases. As a result, $\mathbf{v}_j$ is attracted more strongly by its closest input vectors. This is also true for $g_0(x) = \exp(\beta x)$, but the behavior of $S_{\mathbf{v}_j} h_{j,k}$ is different in this case. For values of σ close to 0, the value of $S_{\mathbf{v}_j} h_{j,k}$ is significant for values of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ very close to 0 but diminishes quickly as $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ increases. In such a case, only the prototypes that are sufficiently close to feature vectors are updated while the rest remain practically unchanged. For sufficiently large values of σ, the value of $S_{\mathbf{v}_j} h_{j,k}$ is practically not affected by the value of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$. In such a case, all prototypes are updated by almost an equal amount regardless of their distance from the feature vectors. In conclusion, the behavior of $S_{\mathbf{v}_j} h_{j,k}$ for $g_0(x) = \exp(\beta x)$ implies that propagating back the output error using gradient descent has practically no effect on the prototypes. In fact, the effectiveness of gradient-descent learning is hampered in this case by the relative insensitivity of the response of the hidden units to perturbations of the prototypes.

Table 1. Number of misclassified feature vectors from the training (E_{train}) and the testing (E_{test}) sets formed from the vowel data.

	$m = 3$ ($\eta = 10^{-5}$)		$m = 4$ ($\eta = 10^{-4}$)		$m = 5$ ($\eta = 10^{-4}$)	
c	E_{train}	E_{test}	E_{train}	E_{test}	E_{train}	E_{test}
20	107	101	103	101	106	89
30	75	79	73	74	72	74
40	77	75	76	72	80	69
50	64	82	61	79	63	75

5. Experimental Results

5.1 Two-dimensional Vowel Data

The performance of RBF neural networks was evaluated using a set of 2-D vowel data formed by computing the first two formants F1 and F2 from samples of 10 vowels spoken by 67 speakers [6]. The available 671 feature vectors were divided into a training set, containing 338 vectors, and a testing set, containing 333 vectors.

The training and testing sets formed from the vowel data were classified by various RBF neural networks trained using the proposed gradient descent learning algorithm. In all these experiments, the function $g(\cdot)$ was of the form $g(x) = (g_0(x))^{\frac{1}{1-m}}$, with $g_0(x) = x$ and $m > 1$. The number of radial basis functions varied from $c = 20$ to $c = 50$. Table 1 summarizes the number of feature vectors from the training and testing sets classified incorrectly by each trained RBF network with $m = 3$, $m = 4$, and $m = 5$. Table 1 also shows the learning rate used for each set of experiments, which was selected in each case sufficiently small so as to avoid a temporary increase of the total error especially in the beginning of the learning process. The maximum allowable value of the learning rate increased as the value of m increased from 3 to 5, which resulted in a reduction of the number of adaptation cycles required for the completion of the learning process. The number of adaptation cycles required for training also decreased as the number of radial basis functions increased. However, Table 1 shows clearly that increasing the number of radial basis functions has a negative impact on the ability of the trained RBF networks to generalize. The number of feature vectors from the training set classified incorrectly by the trained RBF networks decreased as the number of radial basis functions increased from $c = 20$ to $c = 50$. However, the incorrect classifications

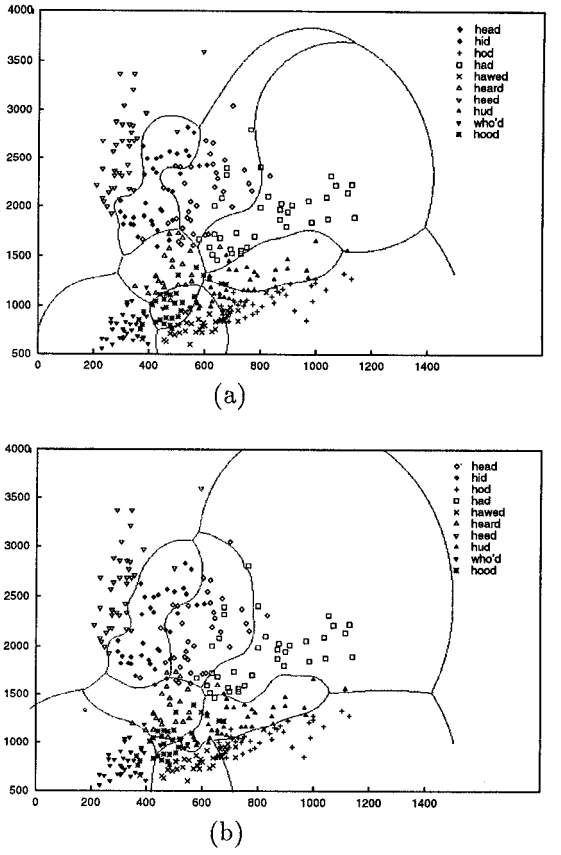

Figure 1. Classification of the 2-D vowel data produced by an RBF neural network trained using gradient descent with (a) $c = 30$ and (b) $c = 40$. $g_0(x) = x$; $m = 3$; $\eta = 10^{-5}$.

on the testing set decreased as the number of radial basis functions increased above $c = 20$ but increased again as the number of radial basis functions increased above $c = 40$. Figures 1(a) and 1(b) show the separation of the feature space produced by the RBF networks trained using gradient descent with $c = 30$ and $c = 40$, respectively. It is clear that the networks attempt to find the best possible compromise in regions of the input space with extensive overlapping between the classes. The two trained networks produce a similar separation of the feature space although some of the regions formed differ in terms of both shape and size.

5.2 IRIS Data

The RBF networks proposed in this paper were also tested using Anderson's IRIS data set. This data set contains 150 feature vectors of dimension 4 which belong to 3 physical classes representing different IRIS

Table 2. Number of misclassified feature vectors from the training (E_{train}) and the testing (E_{test}) sets formed from the IRIS data.

	$m = 2$ ($\eta = 10^{-2}$)		$m = 3$ ($\eta = 10^{-2}$)		$m = 4$ ($\eta = 10^{-2}$)	
c	E_{train}	E_{test}	E_{train}	E_{test}	E_{train}	E_{test}
2	5	10	3	8	3	8
3	2	5	1	3	1	3
4	1	5	0	3	2	2
5	3	5	0	3	2	2
6	0	4	0	4	2	2

subspecies. Each class contains 50 feature vectors. One of the three classes is well separated from the other two, which are not easily separable due to the overlapping of their convex hulls. The available 150 feature vectors were randomly divided into a training and a testing set, each containing 75 feature vectors.

The training set formed from the IRIS data was used to train various RBF neural networks by the proposed gradient descent learning algorithm. All RBF networks tested in these experiments consisted of 4 inputs and 3 linear output units, each corresponding to a physical class. The function $g(\cdot)$ was of the form $g(x) = (g_0(x))^{\frac{1}{1-m}}$, with $g_0(x) = x$ and $m > 1$. The number of radial basis functions varied in these experiments from $c = 2$ to $c = 6$, while the learning rate used to train all RBF networks was $\eta = 10^{-2}$. Table 2 summarizes the feature vectors from the training and testing sets classified incorrectly by each trained RBF network with $m = 2$, $m = 3$, and $m = 4$. The performance of the trained RBF networks on both the training and testing sets improved as the number of radial basis functions increased above $c = 2$. If $c = 2$, the number of prototypes representing the feature vectors is smaller than the number of physical classes. The performance of the trained RBF networks on both the training and testing sets improved as the value of m increased from 2 to 3. The RBF networks with $m = 3$ resulted in the smallest number of classification errors on the training set. On the other hand, the classification errors produced by the RBF networks with $m = 4$ were equally distributed among the training and testing sets.

6. Conclusions

The success of a neural network model depends rather strongly on its association with an attractive learning algorithm. For example, the popularity of conventional feed-forward neural networks was to a large extent due to the error back propagation algorithm. On the other hand, the effectiveness of RBF neural models for function approximation was hampered by the lack of a simple and easily implementable learning algorithm for such models. The limited efficiency of gradient descent learning used for training RBF networks can be attributed to the use of Gaussian radial basis functions, which are generated by an exponential generator function. The axiomatic approach presented in this paper for selecting admissible RBF models indicated that this problem can be solved by replacing the exponential by a linear generator function. The resulting RBF models involve fewer adjustable parameters while their training is fast and simple. The experimental results indicated that the trained networks are very competitive in terms of their generalization ability.

References

[1] D. S. Broomhead and D. Lowe, "Multivariable functional interpolation and adaptive networks," *Complex Systems*, vol. 2, pp. 321–355, 1988.

[2] N. B. Karayiannis, "Fuzzy and possibilistic clustering algorithms based on generalized reformulation," *Proceedings of Fifth International Conference on Fuzzy Systems*, New Orleans, LA, Sept. 8–11, 1996, pp. 1393–1399.

[3] N. B. Karayiannis, "Learning vector quantization: A review," *International Journal of Smart Engineering System Design*, in press, 1997.

[4] N. B. Karayiannis and J. C. Bezdek, "An integrated approach to fuzzy learning vector quantization and fuzzy c-means clustering," *IEEE Transactions on Fuzzy Systems*, in press, 1997.

[5] J. E. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, pp. 281–294, 1989.

[6] K. Ng and R. P. Lippmann, "Practical characteristics of neural network and conventional pattern classifiers," in *Advances in Neural Information Processing Systems 3*, R. P. Lippmann, *et al.*, (eds.), pp. 970–976, Morgan Kaufmann, San Mateo, CA, 1991.

[7] T. Poggio and F. Girosi, "Regularization algorithms for learning that are equivalent to multilayer networks," *Science*, vol. 247, pp. 978–982, 1990.