Reformulated Radial Basis Neural Networks Trained by Gradient Descent

Nicolaos B. Karayiannis, *Member, IEEE*

***Abstract*— This paper presents an axiomatic approach for constructing radial basis function (RBF) neural networks. This approach results in a broad variety of admissible RBF models, including those employing Gaussian RBF's. The form of the RBF's is determined by a generator function. New RBF models can be developed according to the proposed approach by selecting generator functions other than exponential ones, which lead to Gaussian RBF's. This paper also proposes a supervised learning algorithm based on gradient descent for training reformulated RBF neural networks constructed using the proposed approach. A sensitivity analysis of the proposed algorithm relates the properties of RBF's with the convergence of gradient descent learning. Experiments involving a variety of reformulated RBF networks generated by linear and exponential generator functions indicate that gradient descent learning is simple, easily implementable, and produces RBF networks that perform considerably better than conventional RBF models trained by existing algorithms.**

***Index Terms*— Clustering, function approximation, generator function, gradient descent learning, learning vector quantization, radial basis function, radial basis neural networks, reformulation.**

I. 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.

Broomhead and Lowe [4] were the first to explore the use of RBF's in the design of neural networks and to show how RBF networks model nonlinear relationships and implement interpolation. Micchelli [16] showed that RBF neural networks can produce an interpolating surface which exactly passes through all the pairs of the training set. Poggio and Girosi [22] viewed the training of RBF neural networks as an ill-posed problem, in the sense that the information in the training data is not sufficient to reconstruct uniquely the mapping in regions where data are not available. From this point of view, learning is closely related to classical approximation techniques, such as generalized splines and regularization theory. Park and Sandberg [20], [21] proved that RBF networks with one layer of RBF's are capable of universal approximation. Under certain mild conditions on the RBF's, RBF networks are capable of approximating arbitrarily well any function. Similar proofs also exist in the literature for feedforward neural models with sigmoidal nonlinearities [5].

The performance of an RBF network depends on the number and positions of the RBF's, 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: 1) strategies selecting the RBF centers randomly from the training data [4]; 2) strategies employing unsupervised procedures for selecting the RBF centers [14], [18]; and 3) strategies employing supervised procedures for selecting the RBF centers [10], [11], [22].

Most of the existing learning algorithms for RBF neural networks with Gaussian RBF's employ different schemes for updating the *output weights*, i.e., the weights that connect the RBF's with the output units, and the centers of the RBF's, i.e., the vectors in the input space that represent the *prototypes* of the input vectors included in the training set. Moody and Darken [18] proposed a hybrid learning process for training RBF networks, which is widely used in practice. They used the k-means clustering algorithm [2] and a "P nearest-neighbor" heuristic to determine the positions and widths of the RBF's, respectively. The output weights are updated using a supervised least-mean-squares learning rule. The basic problem with such a learning scheme is that the implementation of the desired input–output mapping has no effect on the formation of the prototypes. As a result, such a learning scheme fails to exploit the potential of RBF neural networks by not fully utilizing the information contained in the training set [14]. For example, in pattern classification applications the labels of the input vectors are not used in the formation of the prototypes, which only satisfy the criterion behind the unsupervised scheme used.

RBF neural networks were developed as a natural reaction to the slow convergence of gradient descent learning algorithms developed for feedforward neural networks with sigmoidal hidden units, such as the error backpropagation algorithm and its variants. Although gradient descent was always an alternative for training RBF networks, this approach has often been overlooked in favor of faster learning schemes that separate the computation of the radial basis centers and the update of the output weights. The emphasis on learning speed prevented the development of simple and

Manuscript received April 8, 1997; revised November 20, 1997 and November 19, 1998.

The author is with the Department of Electrical and Computer Engineering, University of Houston, Houston, TX 77204-4793 USA.

Publisher Item Identifier S 1045-9227(99)03893-X.

1045–9227/99$10.00 © 1999 IEEE

easily implementable supervised learning algorithms capable of exploiting the potential of RBF neural networks. As a result of the ineffectiveness of the learning schemes widely used in practice for training RBF neural networks, these neural models have been associated with erratic behavior and poor performance.

This paper presents an axiomatic approach to constructing reformulated RBF neural networks that can be trained by a fully supervised learning algorithm based on gradient descent. The flexibility of the proposed approach allows the selection of RBF's that provide many alternative solutions to the tradeoff between the convergence of gradient descent learning and the performance of trained RBF networks. This paper is organized as follows: Section II presents a nonlinear mapping that can produce unsupervised learning vector quantization algorithms and supervised RBF neural networks. Section III presents an axiomatic approach for reformulating RBF neural networks by using basis functions formed in terms of admissible generator functions. Section IV presents a fully supervised learning algorithm for RBF neural networks based on gradient descent. Section V presents a sensitivity analysis of gradient descent learning. Section VI evaluates the performance of reformulated RBF neural networks produced by linear and exponential generator functions and trained using the proposed learning algorithm. Section VII contains concluding remarks.

II. Unsupervised and Supervised Learning Models Based on a Nonlinear Mapping

The common ingredient of vector quantization and RBF neural networks is the representation of the input vectors by a finite set of prototypes. In vector quantization, the prototypes are used to form a partition of the input vectors. In RBF neural networks, the prototypes are used to form the outputs of the RBF's which feed the upper associative network that implements the desired input–output mapping. The prototypes of RBF neural networks are frequently determined by employing unsupervised clustering techniques or *learning vector quantization* (LVQ) [14]. LVQ is the name used in this paper for *unsupervised* vector quantization algorithms implemented by training a competitive neural network using gradient descent [8]. This section shows that recent developments in learning vector quantization research can be particularly useful in reformulating RBF neural networks. This is accomplished by considering a nonlinear mapping from an n_i-dimensional Euclidean space, which represents the input space, into the real axis.

Consider the finite set of input vectors $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_M\} \subset \mathbb{R}^{n_i}$, which are represented by the set of prototypes $\mathcal{V} = \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_c\} \subset \mathbb{R}^{n_i}$. Consider also the mapping $\mathbb{R}^{n_i} \to \mathbb{R}$ described by

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

where $f(x)$ and $g(x)$ are differentiable everywhere functions of $x \in (0, \infty)$ and $\|\cdot\|$ denotes the Euclidean norm. This mapping can be the basis for developing LVQ algorithms and RBF neural networks.

A. Learning Vector Quantization Models

For $w_0 = 0$ and $w_j = 1/c, 1 \le j \le c$, the mapping defined in (1) reduces to

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

For any functions $f(\cdot)$ and $g(\cdot)$ satisfying the condition $f(g(x)) = x$, (2) represents a discrepancy measure. This is obvious in the trivial case where $c = 1$, that is, when all input vectors $\mathbf{x}_k \in \mathcal{X}$ are represented by a single prototype $\mathbf{v}_1$. If $c = 1$, then $\hat{y}_k = \|\mathbf{x}_k - \mathbf{v}_1\|^2$ and the average R of $\hat{y}_k, 1 \le k \le M$, becomes

$$R = \frac{1}{M}\sum_{k=1}^{M} \|\mathbf{x}_k - \mathbf{v}_1\|^2. \quad (3)$$

According to (3), R measures the discrepancy associated with the representation of the vectors $\mathbf{x}_k \in \mathcal{X}$ by the prototype $\mathbf{v}_1$. In fact, the optimal prototype $\mathbf{v}_1$ can be obtained from $\nabla_{\mathbf{v}_1} R \equiv \partial R/\partial \mathbf{v}_1 = \mathbf{0}$ as the centroid of $\mathbf{x}_k \in \mathcal{X}$, i.e.,

$$\mathbf{v}_1 = \frac{1}{M}\sum_{k=1}^{M} \mathbf{x}_k. \quad (4)$$

For any pair of functions $f(\cdot)$ and $g(\cdot)$ that satisfy the condition $f(g(x)) = x$, consider the function R defined as the average of $\hat{y}_k, 1 \le k \le M$, that is

$$R = \frac{1}{M}\sum_{k=1}^{M} f\left(\frac{1}{c}\sum_{j=1}^{c} g(\|\mathbf{x}_k - \mathbf{v}_j\|^2)\right). \quad (5)$$

For any $c \ge 1$, R measures the discrepancy associated with the representation of $\mathbf{x}_k \in \mathcal{X}$ by the finite set of prototypes $\mathcal{V} = \{\mathbf{v}_1, \mathbf{v}_2, \cdots, \mathbf{v}_c\}$, provided that [12]:

- $f(x)$ and $g(x)$ are both monotonically decreasing functions of $x \in (0, \infty)$ and $g'(x)$ is a monotonically increasing function of $x \in (0, \infty)$; or
- $f(x)$ and $g(x)$ are both monotonically increasing functions of $x \in (0, \infty)$ and $g'(x)$ is a monotonically decreasing function of $x \in (0, \infty)$.

Functions of the form (5) that satisfy the above conditions are often referred to as *reformulation functions*, because their structure was determined by reformulating fuzzy clustering techniques such as the *fuzzy c-means* (FCM) and *entropy-constrained fuzzy clustering* (ECFC) algorithms [6], [7], [9], [13]. Minimization of reformulation functions of the form (5) using gradient descent leads to *unsupervised* LVQ algorithms that can be used to determine the prototypes [8], [9], [13].

A broad variety of LVQ and clustering algorithms can be developed using functions of the form $g(x) = (g_0(x))^{\frac{1}{1-m}}, m \ne 1$, where $g_0(\cdot)$ is an admissible *generator function* [12]. If $g_0(x) = x$, then $g(x) = x^{\frac{1}{1-m}}$, $f(x) = g^{-1}(x) = x^{1-m}$, and (5) gives

$$R_m = \frac{1}{M}\sum_{i=1}^{M}\left(\frac{1}{c}\sum_{j=1}^{c}(\|\mathbf{x}_i - \mathbf{v}_j\|^2)^{\frac{1}{1-m}}\right)^{1-m}. \quad (6)$$

The function (6) relates to the reformulation function that corresponds to the FCM algorithm [6], [13]. Minimization of (6) using gradient descent resulted in batch LVQ algorithms which are closely related to the *fuzzy learning vector quantization* (FLVQ) algorithm [8], [13]. If $g_0(x) = \exp(\sigma x)$ and $m = 1/(1-\mu)$, then $g(x) = \exp(-\sigma\delta_\mu x)$ with $\delta_\mu = (1-\mu)/\mu$, $f(x) = g^{-1}(x) = -\ln x/(\sigma\delta_\mu)$, and (5) gives

$$R_\mu = \frac{1}{M}\sum_{i=1}^{M}\ln\left(\frac{1}{c}\sum_{j=1}^{c}\exp(-\sigma\delta_\mu||\mathbf{x}_i - \mathbf{v}_j||^2)\right)^{-\frac{1}{\sigma\delta_\mu}}. \quad (7)$$

The function (7) relates to the reformulation function that corresponds to ECFC algorithms [7], [9]. Minimization of (7) using gradient descent resulted in *entropy-constrained learning vector quantization* (ECLVQ) algorithms [9].

B. Function Approximation and RBF Models

Consider the mapping (1) and let $f(\cdot)$ be a linear function of the form $f(x) = x$. Under this assumption, (1) gives

$$\hat{y}_k = w_0 + \sum_{j=1}^{c} w_j g(||\mathbf{x}_k - \mathbf{v}_j||^2), \quad 1 \le k \le M. \quad (8)$$

If the prototypes $\mathbf{v}_j, 1 \le j \le c$, and the weights $w_j, 0 \le j \le c$, are adjustable parameters, the model (8) can be used for function approximation provided that the function $g(\cdot)$ satisfies certain mathematical conditions. In such a case, the adjustable parameters of the model

$$\hat{y} = w_0 + \sum_{j=1}^{c} w_j g(||\mathbf{x} - \mathbf{v}_j||^2) \quad (9)$$

can be determined so that (9) implements a desired mapping $\mathbb{R}^{n_i} \to \mathbb{R}$ specified by the training set $(y_k, \mathbf{x}_k), 1 \le k \le M$. The adjustable parameters of the model (9) are frequently updated by minimizing some measure of the discrepancy between the actual response $\hat{y}_k$ of the model to the corresponding input $\mathbf{x}_k$ and the expected output y_k.

The function approximation model (9) can be extended to implement any mapping $\mathbb{R}^{n_i} \to \mathbb{R}^{n_o}$, $n_o \ge 1$, as

$$\hat{y}_i = w_{i0} + \sum_{j=1}^{c} w_{ij} g(||\mathbf{x} - \mathbf{v}_j||^2), \quad 1 \le i \le n_o. \quad (10)$$

The model (10) describes an RBF neural network with inputs from $\mathbb{R}^{n_i}$, c RBF units, and n_o linear output units if $g(x^2) = \phi(x)$ and $\phi(\cdot)$ is an RBF. In such a case, the response of the network to the input vector $\mathbf{x}_k$ is

$$\hat{y}_{i,k} = \sum_{j=0}^{c} w_{ij} h_{j,k}, \quad 1 \le i \le n_o \quad (11)$$

where $h_{0,k} = 1, \forall k$, and $h_{j,k}$ represents the response of the RBF located at the jth prototype to the input vector $\mathbf{x}_k$, given as

$$\begin{aligned} h_{j,k} &= \phi(||\mathbf{x}_k - \mathbf{v}_j||) \\ &= g(||\mathbf{x}_k - \mathbf{v}_j||^2), \quad 1 \le j \le c. \end{aligned} \quad (12)$$

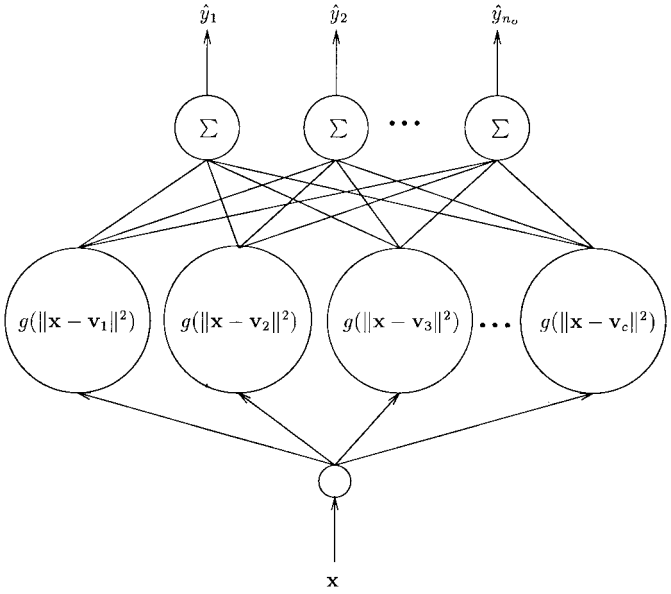


Fig. 1. An RBF neural network.

The response (11) 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 (12) of the RBF's located at the prototypes $\mathbf{v}_j, 1 \le j \le c$, as shown in Fig. 1.

III. Reformulating Radial Basis Neural Networks

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 shapes are determined by the RBF's used. The interpretation often associated with RBF neural networks imposed some silent restrictions on the selection of RBF's. For example, RBF neural networks employ very often the decreasing Gaussian RBF despite the fact that there exist both increasing and decreasing RBF's. The "neural" interpretation of the model (10) can be the basis of a systematic search for RBF's that may be used for reformulating RBF neural networks. Such a systematic search is based on mathematical restrictions imposed on RBF's by their role in the formation of receptive fields.

A. Admissible Radial Basis Functions

The interpretation of an RBF neural network as a composition of receptive fields requires that the responses of all RBF's 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 RBF becomes stronger as the input approaches its corresponding prototype. Finally, it is required that the response of any RBF becomes more sensitive to an input vector as this input vector approaches its corresponding prototype.

Let $h_{j,k} = g(||\mathbf{x}_k - \mathbf{v}_j||^2)$ be the response of the jth RBF of an RBF neural network to the input $\mathbf{x}_k$. According to the above interpretation of RBF neural networks, any admissible

RBF $\phi(x) = g(x^2)$ must satisfy the following three 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 $\nabla_{\mathbf{x}_k} h_{j,k} \equiv \partial h_{j,k}/\partial \mathbf{x}_k$ denotes the gradient of $h_{j,k}$ with respect to the corresponding input $\mathbf{x}_k$, then

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

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

These basic axiomatic requirements impose some rather mild mathematical restrictions on the search for admissible RBF's. Nevertheless, this search can be further restricted by imposing additional requirements that lead to stronger mathematical conditions. For example, it is reasonable to require that the responses of all RBF's to all inputs are bounded, i.e., $h_{j,k} < \infty, \forall j,k$. On the other hand, the third axiomatic requirement can be made stronger by requiring that

$$\left\|\nabla_{\mathbf{x}_k} h_{j,k}\right\|^2 > \left\|\nabla_{\mathbf{x}_\ell} h_{j,\ell}\right\|^2 \tag{13}$$

if $||\mathbf{x}_k - \mathbf{v}_j||^2 < ||\mathbf{x}_\ell - \mathbf{v}_j||^2$. Since $||\mathbf{x}_k - \mathbf{v}_j||^2 < ||\mathbf{x}_\ell - \mathbf{v}_j||^2$

$$\frac{\left\|\nabla_{\mathbf{x}_k} h_{j,k}\right\|^2}{||\mathbf{x}_k - \mathbf{v}_j||^2} > \frac{\left\|\nabla_{\mathbf{x}_k} h_{j,k}\right\|^2}{||\mathbf{x}_\ell - \mathbf{v}_j||^2}. \tag{14}$$

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

$$\frac{\left\|\nabla_{\mathbf{x}_k} h_{j,k}\right\|^2}{||\mathbf{x}_k - \mathbf{v}_j||^2} > \frac{\left\|\nabla_{\mathbf{x}_k} h_{j,k}\right\|^2}{||\mathbf{x}_\ell - \mathbf{v}_j||^2} > \frac{\left\|\nabla_{\mathbf{x}_\ell} h_{j,\ell}\right\|^2}{||\mathbf{x}_\ell - \mathbf{v}_j||^2} \tag{15}$$

and the third axiomatic requirement is satisfied. This implies that the condition (13) is stronger than that imposed by the third axiomatic requirement.

The selection of RBF's in accordance with the three basic axiomatic requirements can be facilitated by the following theorem.

Theorem 1: The model described by (10) represents an RBF neural network in accordance with the three basic axiomatic requirements iff $g(\cdot)$ is a continuous function on $(0,\infty)$ such that:

1) $g(x) > 0,\ \forall x \in (0,\infty)$;
2) $g(x)$ is a monotonically decreasing function of $x \in (0,\infty)$, i.e., $g'(x) < 0, \forall x \in (0,\infty)$;
3) $g'(x)$ is a monotonically increasing function of $x \in (0,\infty)$, i.e., $g''(x) > 0, \forall x \in (0,\infty)$.

Proof: The proof of this theorem is presented in Appendix A. □

The analysis that follows shows that Theorem 1 is consistent with Micchelli's interpolation theorem [16] but imposes fewer restrictions on the selection of admissible functions $g(\cdot)$. Micchelli [16] considered the solution of the interpolation problem $s(\mathbf{x}_k) = y_k, 1 \le k \le M$, by functions $s : \mathbb{R}^{n_i} \to \mathbb{R}$ of the form $s(\mathbf{x}) = \sum_{k=1}^{M} w_k g(||\mathbf{x} - \mathbf{x}_k||^2)$. This study indicated that a function $g(\cdot)$ can be used to solve this interpolation problem if the $M \times M$ matrix $\mathbf{G} = [g_{ij}]$ with entries $g_{ij} = g(||\mathbf{x}_i - \mathbf{x}_j||^2)$ is positive definite. The matrix $\mathbf{G}$ is positive definite if the function $g(\cdot)$ is *completely monotonic* on $(0,\infty)$. A function $g(\cdot)$ is called completely monotonic on $(0,\infty)$ if it is continuous on $(0,\infty)$ and its ℓth order derivatives $g^{(\ell)}(x)$ satisfy $(-1)^\ell g^{(\ell)}(x) \ge 0,\ \forall x \in (0,\infty)$, for $\ell = 0, 1, 2, \cdots$. Theorem 1 guarantees that an admissible function $g(\cdot)$ is continuous on $(0,\infty)$ and also $g(x) > 0, \forall x \in (0,\infty)$, $g'(x) < 0, \forall x \in (0,\infty)$, and $g''(x) > 0, \forall x \in (0,\infty)$. In other words, Theorem 1 requires that any admissible function $g(\cdot)$ is continuous on $(0,\infty)$ and its derivatives satisfy $(-1)^\ell g^{(\ell)}(x) > 0, \forall x \in (0,\infty)$, for $\ell = 0, 1, 2$. Clearly, Theorem 1 is less restrictive than Micchelli's interpolation theorem.

B. From Generator Functions to Radial Basis Functions

The Gaussian RBF $\phi(x) = \exp(-x^2/\sigma^2)$ corresponds to the function $g(x) = \exp(-x/\sigma^2)$. For any $\beta > 0$, there exists an $m > 1$ such that $\sigma^2 = (m-1)/\beta$. Thus, $g(x) = \exp(-x/\sigma^2)$ can be written in terms of β and $m > 1$ as

$$\begin{aligned} g(x) &= \exp(-\beta x/(m-1)) \\ &= (\exp(\beta x))^{\frac{1}{1-m}}. \end{aligned} \tag{16}$$

For any $\beta > 0$, there always exists an $m < 1$ such that $\sigma^2 = (1-m)/\beta$. Thus, $g(x) = \exp(-x/\sigma^2)$ can be also written in terms of β and $m < 1$ as

$$\begin{aligned} g(x) &= \exp(-\beta x/(1-m)) \\ &= (\exp(-\beta x))^{\frac{1}{1-m}}. \end{aligned} \tag{17}$$

The above interpretation of the Gaussian RBF $\phi(x) = \exp(-x^2/\sigma^2)$ indicates that $g(x)$ can be defined in terms of an increasing or decreasing *generator function* $g_0(x)$ as $g(x) = (g_0(x))^{\frac{1}{1-m}}, m \neq 1$. For example, the Gaussian RBF $\phi(x) = \exp(-x^2/\sigma^2)$ can alternatively be defined in terms of the increasing generator function $g_0(x) = \exp(\beta x)$, $\beta > 0$, with $m > 1$, or the decreasing generator function $g_0(x) = \exp(-\beta x)$, $\beta > 0$, with $m < 1$. The selection of generator functions $g_0(\cdot)$ that lead to admissible RBF's of the form $\phi(x) = g(x^2)$ with $g(x) = (g_0(x))^{\frac{1}{1-m}}$, $m \neq 1$, can be facilitated by the following theorem:

Theorem 2: Consider the model (10) and let $g(x)$ be defined in terms of the generator function $g_0(\cdot)$ that is continuous on $(0,\infty)$, as

$$g(x) = (g_0(x))^{\frac{1}{1-m}}, \quad m \neq 1. \tag{18}$$

If $m > 1$, then this model represents an RBF neural network in accordance with the three basic axiomatic requirements if:

1) $g_0(x) > 0,\ \forall x \in (0,\infty)$;
2) $g_0(x)$ is a monotonically increasing function of $x \in (0,\infty)$, i.e., $g_0'(x) > 0, \forall x \in (0,\infty)$;
3) $r_0(x) = \frac{m}{m-1}(g_0'(x))^2 - g_0(x)g_0''(x) > 0,\ \forall x \in (0,\infty)$.

If $m < 1$, then this model represents an RBF neural network in accordance with the three basic axiomatic requirements if:

1. $g_0(x) > 0, \forall x \in (0,\infty)$;
2. $g_0(x)$ is a monotonically decreasing function of $x \in (0,\infty)$, i.e., $g_0'(x) < 0, \forall x \in (0,\infty)$;
3. $r_0(x) = \frac{m}{m-1}(g_0'(x))^2 - g_0(x)g_0''(x) < 0,\ \forall x \in (0,\infty)$.

Proof: The proof of this theorem is presented in Appendix B. □

C. Special Cases

1) Linear Generator Functions: Consider the function $g(x) = (g_0(x))^{\frac{1}{1-m}}$, with $g_0(x) = ax + b$ and $m > 1$. Clearly, $g_0(x) = ax + b > 0$, $\forall x \in (0, \infty)$, for all $a > 0$ and $b \geq 0$. Moreover, $g_0(x) = ax + b$ is a monotonically increasing function if $g_0'(x) = a > 0$. For $g_0(x) = ax + b$, $g_0'(x) = a$, $g_0''(x) = 0$, and

$$r_0(x) = \frac{m}{m-1}a^2. \tag{19}$$

If $m > 1$, then $r_0(x) > 0, \forall x \in (0, \infty)$. Thus, $g_0(x) = ax + b$ is an admissible generator function for all $a > 0$ and $b \geq 0$. If $a = 1$ and $b = \gamma^2$, then the linear generator function becomes $g_0(x) = x + \gamma^2$. For this generator function, $g(x) = (x + \gamma^2)^{\frac{1}{1-m}}$. If $m = 3$, $g(x) = (x + \gamma^2)^{-\frac{1}{2}}$ corresponds to the inverse multiquadratic RBF, which is obtained using $\phi(x) = g(x^2)$ as

$$\phi(x) = \frac{1}{(x^2 + \gamma^2)^{\frac{1}{2}}}. \tag{20}$$

Consider also the function $g(x) = (g_0(x))^{\frac{1}{1-m}}$, with $g_0(x) = 1/(ax + b)$ and $m < 1$. For all $a > 0$ and $b \geq 0$, $g_0(x) = 1/(ax + b) > 0$, $\forall x \in (0, \infty)$. Since $g_0'(x) = -a/(ax+b)^2 < 0, \forall x \in (0, \infty)$, $g_0(x) = 1/(ax+b)$ is a monotonically decreasing function for all $a > 0$. Since $g_0''(x) = 2a^2/(ax + b)^3$

$$r_0(x) = \frac{2-m}{m-1}\frac{a^2}{(ax+b)^4}. \tag{21}$$

For $m < 1$, $r_0(x) < 0, \forall x \in (0, \infty)$, and $g_0(x) = 1/(ax + b)$ is an admissible generator function. If $a = 1$ and $b = \gamma^2$, then $g_0(x) = 1/(x+\gamma^2)$ and $g(x) = (x+\gamma^2)^{\frac{1}{m-1}}$. If $m = -1$, then $g(x) = (x + \gamma^2)^{-\frac{1}{2}}$ corresponds to the inverse multiquadratic RBF (20).

The inverse multiquadratic RBF $\phi(x) = 1/(x^2 + \gamma^2)^{\frac{1}{2}}$ is bounded at $x = 0$ because of the nonzero constant γ. If $\gamma = 0$, 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 $g_0(x) = 1/x$ with $m < 1$ are not bounded at $x = 0$. In such a case, $h_{j,k} = g(||\mathbf{x}_k - \mathbf{v}_j||^2)$ approaches infinity as the prototype $\mathbf{v}_j$ approaches the input vector $\mathbf{x}_k$. Nevertheless, both generator functions $g_0(x) = x + \gamma^2$ and $g_0(x) = 1/(x + \gamma^2)$ produce bounded RBF's for $\gamma \neq 0$.

2) Exponential Generator Functions: Consider the function $g(x) = (g_0(x))^{\frac{1}{1-m}}$, with $g_0(x) = \exp(\beta x)$, $\beta > 0$, and $m > 1$. For any β, $g_0(x) = \exp(\beta x) > 0$, $\forall x \in (0, \infty)$. For all $\beta > 0$, $g_0(x) = \exp(\beta x)$ is a monotonically increasing function. For $g_0(x) = \exp(\beta x)$, $g_0'(x) = \beta \exp(\beta x)$ and $g_0''(x) = \beta^2 \exp(\beta x)$. In this case

$$r_0(x) = \frac{1}{m-1}(\beta \exp(\beta x))^2. \tag{22}$$

If $m > 1$, then $r_0(x) > 0, \forall x \in (0, \infty)$. Thus, $g_0(x) = \exp(\beta x)$ is an admissible generator function for all $\beta > 0$. If $g_0(x) = \exp(\beta x)$, with $\beta > 0$ and $m > 1$, then $g(x) = (\exp(\beta x))^{\frac{1}{1-m}}$ corresponds to the Gaussian RBF $\phi(x) = g(x^2) = \exp(-x^2/\sigma^2)$, with $\sigma^2 = (m-1)/\beta$.

Consider also the function $g(x) = (g_0(x))^{\frac{1}{1-m}}$, with $g_0(x) = \exp(-\beta x)$, $\beta > 0$, and $m < 1$. For any β, $g_0(x) = \exp(-\beta x) > 0$, $\forall x \in (0, \infty)$. For all $\beta > 0$, $g_0'(x) = -\beta \exp(-\beta x) < 0$, $\forall x \in (0, \infty)$, and $g_0(x) = \exp(-\beta x)$ is a monotonically decreasing function. Since $g_0''(x) = \beta^2 \exp(-\beta x)$

$$r_0(x) = \frac{1}{m-1}(\beta \exp(-\beta x))^2. \tag{23}$$

If $m < 1$, then $r_0(x) < 0, \forall x \in (0, \infty)$, and $g_0(x) = \exp(-\beta x)$ is an admissible generator function for all $\beta > 0$. If $g_0(x) = \exp(-\beta x)$, with $\beta > 0$ and $m < 1$, then $g(x) = (\exp(-\beta x))^{\frac{1}{1-m}}$ corresponds to the Gaussian RBF $\phi(x) = g(x^2) = \exp(-x^2/\sigma^2)$, with $\sigma^2 = (1-m)/\beta$.

It can easily be verified that both increasing generator functions $g_0(x) = x + \gamma^2$ and $g_0(x) = \exp(\beta x)$ lead to functions $g(x) = (g_0(x))^{\frac{1}{1-m}}$, $m > 1$, which are continuous on $(0, \infty)$ and their derivatives satisfy the conditions $(-1)^\ell g^{(\ell)}(x) \geq 0$, $\forall x \in (0, \infty)$, for $\ell = 0, 1, 2, \cdots$. The same conditions are also satisfied by the functions $g(x) = (g_0(x))^{\frac{1}{1-m}}$, $m < 1$, obtained from the decreasing generator functions $g_0(x) = 1/(x+\gamma^2)$ and $g_0(x) = \exp(-\beta x)$. In other words, the functions $g(\cdot)$ considered above as special cases of Theorems 1 and 2 are completely monotonic on $(0, \infty)$ and, thus, they also satisfy Micchelli's interpolation theorem [16].

IV. A Learning Algorithm Based on Gradient Descent

An RBF neural network can be trained by minimizing the error

$$E = \frac{1}{2}\sum_{k=1}^{M}\sum_{i=1}^{n_o}(y_{i,k} - \hat{y}_{i,k})^2 \tag{24}$$

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

$$\begin{aligned}\hat{y}_{i,k} &= \mathbf{w}_i^T \mathbf{h}_k \\ &= \sum_{j=0}^{c} w_{ij} h_{j,k}\end{aligned} \tag{25}$$

where $h_{0,k} = 1$, and $h_{j,k} = g(||\mathbf{x}_k - \mathbf{v}_j||^2)$, $1 \leq j \leq c$, $\mathbf{h}_k = [h_{0,k}\ h_{1,k}\ \cdots\ h_{c,k}]^T$, and $\mathbf{w}_i = [w_{i,0}\ w_{i,1}\ \cdots\ w_{i,c}]^T$.

The update equation for the weight vectors of the upper associative network is obtained in Appendix C using gradient descent as

$$\begin{aligned}\Delta \mathbf{w}_p &= -\eta \nabla_{\mathbf{w}_p} E \\ &= \eta \sum_{k=1}^{M} \varepsilon_{p,k}^o \mathbf{h}_k\end{aligned} \tag{26}$$

where η is the learning rate and $\varepsilon_{p,k}^o$ is the *output error*, given as

$$\varepsilon_{p,k}^o = y_{p,k} - \hat{y}_{p,k}. \tag{27}$$

The update equation for the prototypes is obtained in Appendix C using gradient descent as

$$\begin{aligned}\Delta\mathbf{v}_q &= -\eta\nabla_{\mathbf{v}_q}E \\ &= \eta\sum_{k=1}^{M}\varepsilon_{q,k}^{h}(\mathbf{x}_k-\mathbf{v}_q)\end{aligned} \tag{28}$$

where η is the learning rate and $\varepsilon_{p,k}^{h}$ is the *hidden error*, given as

$$\varepsilon_{q,k}^{h}=\alpha_{q,k}\sum_{i=1}^{n_o}\varepsilon_{i,k}^{o}w_{iq} \tag{29}$$

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

An RBF neural network can be trained according to the algorithm derived above in a sequence of *adaptation cycles*, where an adaptation cycle involves the update of all adjustable parameters of the network. An adaptation cycle begins by replacing the current estimate of each weight vector $\mathbf{w}_p, 1\le p\le n_o$, by its updated version

$$\mathbf{w}_p+\eta\sum_{k=1}^{M}\varepsilon_{p,k}^{o}\mathbf{h}_k. \tag{30}$$

Given the responses $\mathbf{h}_k$ of the RBF's, these weight vectors are updated according to the output errors $\varepsilon_{p,k}^{o}, 1\le p\le n_o$. Following the update of these weight vectors, the current estimate of each prototype $\mathbf{v}_q$, $1\le q\le c$, is replaced by

$$\begin{aligned}&\mathbf{v}_q+\eta\sum_{k=1}^{M}\varepsilon_{q,k}^{h}(\mathbf{x}_k-\mathbf{v}_q) \\ &\quad=\left(1-\eta\sum_{k=1}^{M}\varepsilon_{q,k}^{h}\right)\mathbf{v}_q+\eta\sum_{k=1}^{M}\varepsilon_{q,k}^{h}\mathbf{x}_k.\end{aligned} \tag{31}$$

The update of $\mathbf{v}_q$ depends on the hidden errors $\varepsilon_{q,k}^{h}, 1\le k\le M$. The hidden error $\varepsilon_{q,k}^{h}$ is influenced by the output errors $\varepsilon_{i,k}^{o}, 1\le i\le n_o$, and the weights $w_{iq}, 1\le i\le n_o$, through the term $\sum_{i=1}^{n_o}\varepsilon_{i,k}^{o}w_{iq}$. Thus, the RBF network is trained according to this scheme by propagating back the output error.

The selection of a specific function $g(\cdot)$ influences the gradient descent learning algorithm presented above through $\alpha_{q,k}=-2g'(\|\mathbf{x}_k-\mathbf{v}_q\|^2)$, which is involved in the calculation of the corresponding hidden error $\varepsilon_{q,k}^{h}$. If $g(x)=(g_0(x))^{\frac{1}{1-m}}$, then

$$\begin{aligned}g'(x)&=\frac{1}{1-m}(g_0(x))^{\frac{m}{1-m}}g_0'(x) \\ &=\frac{1}{1-m}(g(x))^{m}g_0'(x).\end{aligned} \tag{32}$$

Since $h_{q,k}=g(\|\mathbf{x}_k-\mathbf{v}_q\|^2)$, $\alpha_{q,k}=-2g'(\|\mathbf{x}_k-\mathbf{v}_q\|^2)$ is given by

$$\alpha_{q,k}=\frac{2}{m-1}(h_{q,k})^{m}g_0'(\|\mathbf{x}_k-\mathbf{v}_q\|^2) \tag{33}$$

and the hidden error (29) becomes

$$\varepsilon_{q,k}^{h}=\frac{2}{m-1}(h_{q,k})^{m}g_0'(\|\mathbf{x}_k-\mathbf{v}_q\|^2)\sum_{i=1}^{n_o}\varepsilon_{i,k}^{o}w_{iq}. \tag{34}$$

If $g_0(x)=x+\gamma^2$, then $g_0'(x)=1$ and (33) gives

$$\begin{aligned}\alpha_{q,k}&=\frac{2}{m-1}(h_{q,k})^{m} \\ &=\frac{2}{m-1}(\|\mathbf{x}_k-\mathbf{v}_q\|^2+\gamma^2)^{\frac{m}{1-m}}.\end{aligned} \tag{35}$$

If $g_0(x)=\exp(\beta x)$, then $g_0'(x)=\beta\exp(\beta x)=\beta g_0(x)$. In this case, $g_0'(\|\mathbf{x}_k-\mathbf{v}_q\|^2)=\beta(h_{q,k})^{1-m}$ and (33) gives

$$\begin{aligned}\alpha_{q,k}&=\frac{2\beta}{m-1}h_{q,k} \\ &=\frac{2}{\sigma^2}\exp\left(-\frac{\|\mathbf{x}_k-\mathbf{v}_q\|^2}{\sigma^2}\right)\end{aligned} \tag{36}$$

with $\sigma^2=(m-1)/\beta$.

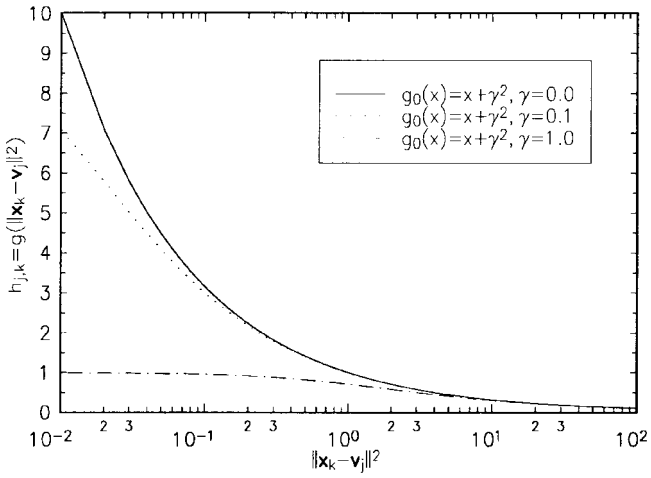
This algorithm can be summarized as follows:

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:
   - $h_{j,k}=(g_0(\|\mathbf{x}_k-\mathbf{v}_j\|^2))^{\frac{1}{1-m}},\ \forall j,k.$
   - $\mathbf{h}_k=[h_{0,k}\ \ h_{1,k}\ \ \cdots\ \ h_{c,k}]^T,\ \forall k.$
   - $\hat{y}_{i,k}=\mathbf{w}_i^T\mathbf{h}_k,\ \forall i,k.$
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:
   - $\varepsilon_{i,k}^{o}=y_{i,k}-\hat{y}_{i,k},\ \forall i,k.$
   - $\mathbf{w}_i\leftarrow\mathbf{w}_i+\eta\sum_{k=1}^{M}\varepsilon_{i,k}^{o}\mathbf{h}_k,\ \forall i.$
   - $\varepsilon_{j,k}^{h}=\frac{2}{m-1}g_0'(\|\mathbf{x}_k-\mathbf{v}_j\|^2)(h_{j,k})^{m}\ \sum_{i=1}^{n_o}\varepsilon_{i,k}^{o}w_{ij},\ \forall j,k.$
   - $\mathbf{v}_j\leftarrow\mathbf{v}_j+\eta\sum_{k=1}^{M}\varepsilon_{j,k}^{h}(\mathbf{x}_k-\mathbf{v}_j),\ \forall j.$
6) Compute the current response:
   - $h_{j,k}=(g_0(\|\mathbf{x}_k-\mathbf{v}_j\|^2))^{\frac{1}{1-m}},\ \forall j,k.$
   - $\mathbf{h}_k=[h_{0,k}\ \ h_{1,k}\ \ \cdots\ \ h_{c,k}]^T,\ \forall k.$
   - $\hat{y}_{i,k}=\mathbf{w}_i^T\mathbf{h}_k,\ \forall i,k.$
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 Step 4).

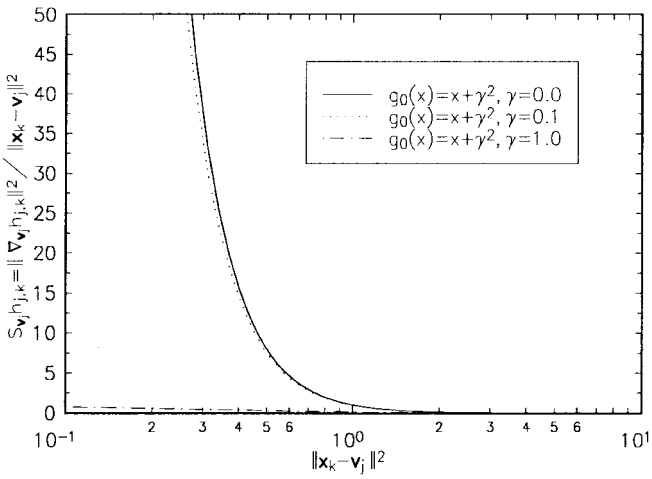
V. Sensitivity Analysis of Gradient Descent Learning

The gradient descent learning algorithm developed in Section IV attempts to train a reformulated RBF neural network to implement a desired input–output mapping by producing incremental changes of its adjustable parameters, i.e., the output weights and the prototypes. If the responses of the RBF's are not substantially affected by incremental changes of the prototypes, then the learning process reduces to incremental changes of the output weights and eventually the algorithm trains a single-layered neural network. Given the limitations of single-layered neural networks [15], such updates alone are unlikely to implement nontrivial input–output mappings.

The sensitivity of the RBF model to incremental changes of the prototypes is partly determined by the responses of the RBF's which feed the upper associative network. The analysis that follows investigates the effect of the free parameters

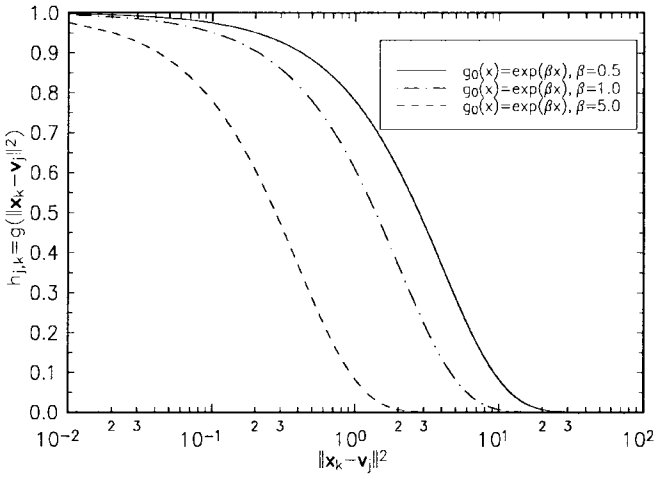


(a)

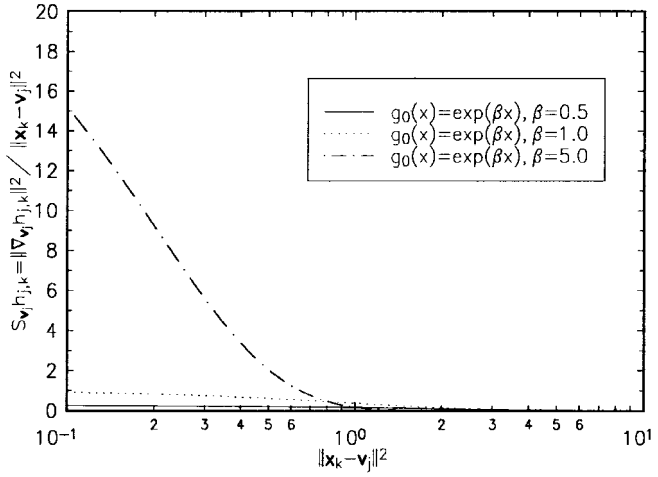


(b)

Fig. 2. The (a) response $h_{j,k} = g(\|\mathbf{x}_k - \mathbf{v}_j\|^2)$ and (b) sensitivity $S_{\mathbf{v}_j} h_{j,k} = \|\nabla_{\mathbf{v}_j} h_{j,k}\|^2 / \|\mathbf{x}_k - \mathbf{v}_j\|^2$ of the jth RBF for $g(x) = (g_0(x))^{\frac{1}{1-m}}$, with $g_0(x) = x + \gamma^2$, $m = 3$, and $\gamma = 0$, $\gamma = 0.1$, $\gamma = 1$.



(a)



(b)

Fig. 3. The (a) response $h_{j,k} = g(\|\mathbf{x}_k - \mathbf{v}_j\|^2)$ and (b) sensitivity $S_{\mathbf{v}_j} h_{j,k} = \|\nabla_{\mathbf{v}_j} h_{j,k}\|^2 / \|\mathbf{x}_k - \mathbf{v}_j\|^2$ of the jth RBF as a function of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ for $g(x) = (g_0(x))^{\frac{1}{1-m}}$, with $g_0(x) = \exp(\beta x)$, $m = 3$, and $\beta = 0.5$, $\beta = 1$, $\beta = 5$.

involved in the definition of linear and exponential generator functions on the responses of the RBF's used for constructing reformulated RBF neural networks.

If $g_0(x) = x + \gamma^2$ and $m > 1$, the response of the jth RBF to $\mathbf{x}_k$ is

$$h_{j,k} = \left(\frac{1}{\|\mathbf{x}_k - \mathbf{v}_j\|^2 + \gamma^2} \right)^{\frac{1}{m-1}}. \tag{37}$$

Fig. 2(a) plots the response $h_{j,k}$ defined in (37) as a function of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ for $m = 3$ and $\gamma = 0$, $\gamma = 0.1$, $\gamma = 1$. Clearly, there is no upper bound for the values of $h_{j,k}$ produced by $\gamma = 0$ and $m > 1$. If the generator function is $g_0(x) = x + \gamma^2$ with $\gamma \neq 0$, then the upper bound of $h_{j,k}$ can be made arbitrarily large by selecting a value of γ close to zero. If $g_0(x) = x + \gamma^2$ and $m > 1$, then the value of $h_{j,k}$ decreases and approaches zero as $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ increases and approaches infinity. As γ approaches zero, even small changes of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ can produce very large changes in $h_{j,k}$ especially for values of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ in the interval $(0, 1)$. The sensitivity of $h_{j,k}$ to changes of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ reduces considerably as the value of γ increases and approaches one.

If $g_0(x) = \exp(\beta x)$ and $m > 1$, the response of the jth RBF to $\mathbf{x}_k$ is

$$h_{j,k} = \exp\left(-\frac{\|\mathbf{x}_k - \mathbf{v}_j\|^2}{\sigma^2} \right) \tag{38}$$

where $\sigma^2 = (m - 1)/\beta$. Fig. 3(a) plots the response $h_{j,k}$ defined in (38) as a function of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ for $m = 3$ and $\beta = 0.5$, $\beta = 1$, $\beta = 5$. For $g_0(x) = \exp(\beta x)$, $h_{j,k}$ decreases exponentially from the value of one and approaches zero as $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ increases from zero to infinity. Since the value of $h_{j,k}$ is bounded by one regardless of the distance between $\mathbf{x}_k$ and $\mathbf{v}_j$, a change of the prototype $\mathbf{v}_j$ can never produce a change in the value of $h_{j,k}$ that exceeds one. The width $\sigma^2 = (m - 1)/\beta$ of the Gaussian response (38) is inversely proportional to β, which implies that increasing the value of β results in a faster decay of $h_{j,k}$ from one to zero. Thus,

increasing the value of β decreases the width of the receptive fields located at the RBF centers.

The ability of reformulated RBF neural networks to implement a desired input–output mapping depends to a large extent on the sensitivity of the responses of the RBF's to incremental changes of their corresponding prototypes. The sensitivity of the response $h_{j,k}$ of the jth RBF to changes of its corresponding prototype $\mathbf{v}_j$ can be measured by

$$S_{\mathbf{v}_j} h_{j,k} = \frac{\left\|\nabla_{\mathbf{v}_j} h_{j,k}\right\|^2}{\|\mathbf{x}_k - \mathbf{v}_j\|^2} = \alpha_{j,k}^2. \tag{39}$$

Since $\alpha_{j,k} = -2g'(\|\mathbf{x}_k - \mathbf{v}_j\|^2)$ and $g(x) = (g_0(x))^{\frac{1}{1-m}}$, the form of the sensitivity measure at (39) indicates that the selection of the generator function $g_0(\cdot)$ has a significant effect on the performance of the learning algorithm based on gradient descent. This effect is investigated in the analysis that follows by studying the behavior of the sensitivity measure (39) with respect to the free parameters involved in the definition of linear and exponential generator functions.

If $g(x) = (g_0(x))^{\frac{1}{1-m}}$ with $g_0(x) = x + \gamma^2$ and $m > 1$, then

$$\alpha_{j,k} = \frac{2}{m-1}\left(\frac{1}{\|\mathbf{x}_k - \mathbf{v}_j\|^2 + \gamma^2}\right)^{\frac{m}{m-1}}. \tag{40}$$

For this generator function, (39) gives

$$S_{\mathbf{v}_j} h_{j,k} = \left(\frac{2}{m-1}\right)^2 \left(\frac{1}{\|\mathbf{x}_k - \mathbf{v}_j\|^2 + \gamma^2}\right)^{\frac{2m}{m-1}}. \tag{41}$$

Fig. 2(b) plots the sensitivity measure (41) corresponding to $g_0(x) = x + \gamma^2$ as a function of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ for $m = 3$ and $\gamma = 0$, $\gamma = 0.1$, $\gamma = 1$. Regardless of the value of γ, the sensitivity $S_{\mathbf{v}_j} h_{j,k}$ is a monotonically decreasing function of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ and approaches zero as $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ approaches infinity. This implies that the sensitivity of the jth RBF increases as the Euclidean 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 during learning. The strength of attraction depends on the behavior of $S_{\mathbf{v}_j} h_{j,k}$ as $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ decreases and approaches zero. It is clear from Fig. 2(b) that the value of γ has a significant effect on the sensitivity $S_{\mathbf{v}_j} h_{j,k}$ especially for $\|\mathbf{x}_k - \mathbf{v}_j\|^2 < 1$. If $\gamma = 0$, $S_{\mathbf{v}_j} h_{j,k}$ is not bounded at $\|\mathbf{x}_k - \mathbf{v}_j\|^2 = 0$. If $\gamma \neq 0$, the bound of $S_{\mathbf{v}_j} h_{j,k}$ can be made arbitrarily large by selecting a value of γ close to zero. As γ increases above zero and approaches one, the bound of $S_{\mathbf{v}_j} h_{j,k}$ at $\|\mathbf{x}_k - \mathbf{v}_j\|^2 = 0$ decreases and approaches $(2/(m-1))^2$. This implies that the sensitivity of the RBF's to changes of their respective prototypes decreases as the value of γ increases. For a fixed value of γ, the sensitivity $S_{\mathbf{v}_j} h_{j,k}$ decreases as the value of m increases. Given the behavior of $h_{j,k}$ for different values of γ and m, the convergence of the gradient descent learning algorithm is expected to slow down as the value of γ increases from zero to one. Increasing the value of m for fixed value of γ is also expected to have a similar effect on the convergence of the gradient descent learning algorithm.

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

$$\alpha_{j,k} = \frac{2}{\sigma^2} \exp\left(-\frac{\|\mathbf{x}_k - \mathbf{v}_j\|^2}{\sigma^2}\right). \tag{42}$$

For this generator function, (39) gives

$$S_{\mathbf{v}_j} h_{j,k} = \left(\frac{2}{\sigma^2}\right)^2 \exp\left(-2\frac{\|\mathbf{x}_k - \mathbf{v}_j\|^2}{\sigma^2}\right). \tag{43}$$

Fig. 3(b) plots the sensitivity measure (43) corresponding to $g_0(x) = \exp(\beta x)$ as a function of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ for $m = 3$ and $\beta = 0.5$, $\beta = 1$, $\beta = 5$. Note that because of the properties of the exponential generator function the sensitivity $S_{\mathbf{v}_j} h_{j,k}$ can be expressed as a function of $\sigma^2 = (m-1)/\beta$. Thus, for a fixed value of $m > 1$, the behavior of $S_{\mathbf{v}_j} h_{j,k}$ depends exclusively on the value of β. Once again, the sensitivity $S_{\mathbf{v}_j} h_{j,k}$ decreases monotonically as $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ increases and approaches infinity. Moreover, $S_{\mathbf{v}_j} h_{j,k}$ is bounded at $\|\mathbf{x}_k - \mathbf{v}_j\|^2 = 0$ for finite values of β. If the value of β is sufficiently smaller that one, the sensitivity $S_{\mathbf{v}_j} h_{j,k}$ is not significantly affected by changes of the Euclidean distance $\|\mathbf{x}_k - \mathbf{v}_j\|^2$. This implies that the input vectors have a rather insignificant and uniform effect on various RBF's regardless of their distance from their respective prototypes. In addition, the Gaussian RBF's corresponding to such values of β are very wide as indicated by Fig. 3(a). This can make gradient descent learning impossible. As β increases, the bound of $S_{\mathbf{v}_j} h_{j,k}$ at $\|\mathbf{x}_k - \mathbf{v}_j\|^2 = 0$ increases. This leads to a stronger attraction between the prototype $\mathbf{v}_j$ and its closest input vectors. However, the corresponding Gaussian RBF's become increasingly sharp as the value of β increases. Thus, gradient descent learning may require an excessive number of RBF's to guarantee that all input vectors are included in the receptive fields located at the prototypes. This discussion indicates that reformulated RBF neural networks generated by an exponential generator function $g_0(x) = \exp(\beta x)$ can be trained by gradient descent only for a data-dependent range of values of β or, equivalently, for a data-dependent range of values of σ.

For both generator functions considered, $S_{\mathbf{v}_j} h_{j,k} = \|\nabla_{\mathbf{v}_j} h_{j,k}\|^2 / \|\mathbf{x}_k - \mathbf{v}_j\|^2$ is a monotonically decreasing function of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$, which implies that

$$S_{\mathbf{v}_j} h_{j,k} = \frac{\left\|\nabla_{\mathbf{v}_j} h_{j,k}\right\|^2}{\|\mathbf{x}_k - \mathbf{v}_j\|^2} > \frac{\left\|\nabla_{\mathbf{v}_\ell} h_{\ell,k}\right\|^2}{\|\mathbf{x}_k - \mathbf{v}_\ell\|^2} = S_{\mathbf{v}_\ell} h_{\ell,k} \tag{44}$$

if $\|\mathbf{x}_k - \mathbf{v}_j\|^2 < \|\mathbf{x}_k - \mathbf{v}_\ell\|^2$. However, there is no guarantee that the generator functions satisfy the stronger condition $\|\nabla_{\mathbf{v}_j} h_{j,k}\|^2 > \|\nabla_{\mathbf{v}_\ell} h_{\ell,k}\|^2$ if $\|\mathbf{x}_k - \mathbf{v}_j\|^2 < \|\mathbf{x}_k - \mathbf{v}_\ell\|^2$.

For $g_0(x) = x + \gamma^2$ and $m > 1$, $\|\nabla_{\mathbf{v}_j} h_{j,k}\|^2$ can be obtained by combining (39) and (40) as

$$\left\|\nabla_{\mathbf{v}_j} h_{j,k}\right\|^2 = \left(\frac{2}{m-1}\right)^2 \frac{\|\mathbf{x}_k - \mathbf{v}_j\|^2}{\|\mathbf{x}_k - \mathbf{v}_j\|^2 + \gamma^2} \times \left(\frac{1}{\|\mathbf{x}_k - \mathbf{v}_j\|^2 + \gamma^2}\right)^{\frac{m+1}{m-1}}. \tag{45}$$

It can be verified that $\|\nabla_{\mathbf{v}_j} h_{j,k}\|^2$ is a monotonically decreasing function of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ for all $\|\mathbf{x}_k - \mathbf{v}_j\|^2 > \gamma^2(m -$

$1)/(m+1)$. If $\gamma = 0$, then $\|\nabla_{\mathbf{v}_j} h_{j,k}\|^2$ is a monotonically decreasing function of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ for all $\|\mathbf{x}_k - \mathbf{v}_j\|^2 > 0$. This is also clear from (45), which for $\gamma = 0$ becomes

$$\left\|\nabla_{\mathbf{v}_j} h_{j,k}\right\|^2 = \left(\frac{2}{m-1}\right)^2 \left(\frac{1}{\|\mathbf{x}_k - \mathbf{v}_j\|^2}\right)^{\frac{m+1}{m-1}}. \quad (46)$$

This implies that if the generator function is $g_0(x) = x$, then

$$\left\|\nabla_{\mathbf{v}_j} h_{j,k}\right\|^2 > \left\|\nabla_{\mathbf{v}_\ell} h_{\ell,k}\right\|^2 \quad (47)$$

if $\|\mathbf{x}_k - \mathbf{v}_j\|^2 < \|\mathbf{x}_k - \mathbf{v}_\ell\|^2$. This condition is clearly stronger than the condition (44), which is satisfied by the generator function $g_0(x) = x + \gamma^2$ with $\gamma \neq 0$. The price to be paid for selecting the value $\gamma = 0$ is that the responses of the RBF's generated by $g_0(x) = x$ are not guaranteed to be bounded. This reveals the tradeoff associated with the selection of the value of γ in practical situations.

For $g_0(x) = \exp(\beta x)$ and $m > 1$, $\|\nabla_{\mathbf{v}_j} h_{j,k}\|^2$ can be obtained by combining (39) and (42) as

$$\left\|\nabla_{\mathbf{v}_j} h_{j,k}\right\|^2 = \left(\frac{2}{\sigma^2}\right)^2 \|\mathbf{x}_k - \mathbf{v}_j\|^2 \exp\left(-2\frac{\|\mathbf{x}_k - \mathbf{v}_j\|^2}{\sigma^2}\right). \quad (48)$$

It can be verified that $\|\nabla_{\mathbf{v}_j} h_{j,k}\|^2$ is a monotonically decreasing function of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ for all $\|\mathbf{x}_k - \mathbf{v}_j\|^2 > \sigma^2/2 = (m-1)/2\beta$. Since $m > 1$, there is no finite value of β that guarantees that $\|\nabla_{\mathbf{v}_j} h_{j,k}\|^2$ is a monotonically decreasing function of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ for all $\|\mathbf{x}_k - \mathbf{v}_j\|^2 > 0$. Nevertheless, the value of $\sigma^2/2$ decreases and approaches zero as β increases and approaches infinity. The tradeoff associated with the selection of the value of β in practice is revealed by Fig. 3(a), which indicates that as the value of β increases the response $h_{j,k}$ of the RBF located at $\mathbf{v}_j$ to an input vector $\mathbf{x}_k$ diminishes very quickly with $\|\mathbf{x}_k - \mathbf{v}_j\|^2$.

VI. Experimental Results

A. Two-Dimensional Vowel Data

The performance of reformulated RBF neural networks generated by the proposed approach was evaluated and compared with that of conventional RBF networks 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 [17], [19]. This data set has been extensively used to compare different pattern classification approaches because of the significant overlapping between the points corresponding to different vowels in the F1-F2 plane [17], [19]. The available 671 feature vectors were divided into a training set, containing 338 vectors, and a testing set, containing 333 vectors. The RBF networks tested in these experiments consisted of two inputs and ten linear output units, each representing a vowel. The number of RBF's varied in these experiments from 15 to 50. All RBF networks were trained using a normalized version of the input data produced by replacing each feature sample x by $\bar{x} = (x - \mu_x)/\sigma_x$, where μ_x and σ_x denote the sample mean and standard deviation of this feature over the entire data set. The networks were trained to respond with $y_{i,k} = 1$ and $y_{j,k} = 0$, $\forall j \neq i$, when presented with an input vector $\mathbf{x}_k \in \mathcal{X}$ representing the ith vowel. The assignment of input vectors to classes was based on a winner-takes-all strategy, that is, each input vector was assigned to the class represented by the output unit of the trained RBF network with the maximum response. The conventional RBF neural networks were trained using the hybrid learning procedure proposed in [18]. The prototypes representing the locations of the RBF's in the input space were determined by the k-means algorithm while the output weights were updated to minimize the error at the output using gradient descent with learning rate $\eta = 10^{-2}$. The widths of the Gaussian RBF's were determined according to the "closest neighbor" heuristic. More specifically, the width σ_j of the RBF located at the prototype $\mathbf{v}_j$ was determined as $\sigma_j = \min_{\ell \neq j}\{\|\mathbf{v}_j - \mathbf{v}_\ell\|\}$. The reformulated RBF neural networks were trained using the gradient descent algorithm described in this paper while the learning rate was selected sufficiently small so as to avoid a temporary increase of the total error especially in the beginning of the learning process.

TABLE I
Percentage of Classification Errors Produced on the Average in Ten Trials on the Training Set (E_{train}) and the Testing Set (E_{test}) Formed from the 2-D Vowel Data by Reformulated RBF Networks Containing c Radial Basis Functions of the Form $\phi(x) = g(x^2)$, with $g(x) = (g_0(x))^{\frac{1}{1-m}}$, $g_0(x) = x + \gamma^2$, $m = 3$, and Various Values of γ. The Number Shown in Parenthesis Is the Standard Deviation

	$\gamma = 0.0$ ($\eta = 10^{-5}$)		$\gamma = 0.1$ ($\eta = 10^{-4}$)		$\gamma = 1.0$ ($\eta = 10^{-4}$)	
c	E_{train} (σ_{train})	E_{test} (σ_{test})	E_{train} (σ_{train})	E_{test} (σ_{test})	E_{train} (σ_{train})	E_{test} (σ_{test})
15	31.2 (3.8)	30.5 (3.4)	29.1 (3.4)	28.4 (2.1)	24.4 (1.2)	23.8 (0.9)
20	29.0 (2.1)	28.4 (1.3)	25.7 (1.4)	26.0 (2.7)	23.9 (1.4)	21.9 (0.6)
25	25.1 (1.6)	23.8 (2.8)	21.7 (0.8)	22.3 (1.3)	22.7 (0.8)	21.8 (1.0)
30	22.5 (2.9)	25.0 (2.2)	20.2 (1.4)	21.7 (0.5)	22.9 (0.8)	20.8 (0.9)
35	23.8 (2.7)	23.2 (2.7)	19.5 (1.5)	21.2 (2.2)	22.7 (0.2)	20.2 (0.7)
40	23.3 (1.6)	22.8 (1.7)	18.1 (2.2)	21.7 (2.5)	22.3 (0.4)	20.5 (0.8)
45	23.6 (1.5)	23.5 (1.0)	17.0 (1.8)	21.5 (1.2)	22.4 (0.5)	20.4 (0.2)
50	20.2 (1.2)	22.3 (1.6)	15.3 (1.4)	21.2 (0.9)	22.0 (0.8)	20.6 (0.4)

Table I shows the percentage of classification errors produced on the average in ten trials on the training and testing sets formed from the 2-D vowel data by reformulated RBF networks generated by linear generator functions $g_0(x) = x + \gamma^2$ with $m = 3$ and $\gamma = 0$, $\gamma = 0.1$, $\gamma = 1$. It is clear from Table I that the performance of reformulated RBF networks improved as the value of γ increased from zero to one. The improvement was significant when the reformulated RBF networks contained a relatively small number of RBF's (15 to 25). This is particularly important, since the ability of RBF networks to generalize degrades as the number of RBF's increases. The price to be paid for such an improvement in performance is slower convergence of the learning algorithm. It was experimentally verified that the gradient descent learning algorithm converges slower as the value of γ increases from zero to one. The average number of adaptation cycles required in ten trials for the convergence of the gradient descent algorithm tested for $c = 30$ with $\gamma = 0$, $\gamma = 0.1$, and $\gamma = 1$ was 6 017, 9076, and 22 170, respectively. This is consistent with the sensitivity analysis presented in Section V and the behavior of the sensitivity measure $S_{\mathbf{v}_j} h_{j,k}$, which

TABLE II
PERCENTAGE OF CLASSIFICATION ERRORS PRODUCED ON THE AVERAGE IN 10 TRIALS ON THE TRAINING SET (E_{train}) AND THE TESTING SET (E_{test}) FORMED FROM THE 2-D VOWEL DATA BY REFORMULATED RBF NETWORKS CONTAINING c RADIAL BASIS FUNCTIONS OF THE FORM $\phi(x) = g(x^2)$, WITH $g(x) = (g_0(x))^{\frac{1}{1-m}}$, $g_0(x) = \exp(\beta x)$, $m = 3$, AND VARIOUS VALUES OF β. THE NUMBER SHOWN IN PARENTHESIS IS THE STANDARD DEVIATION

	$\beta = 0.5$ ($\eta = 10^{-4}$)		$\beta = 1.0$ ($\eta = 10^{-4}$)		$\beta = 5.0$ ($\eta = 10^{-4}$)	
c	E_{train} (σ_{train})	E_{test} (σ_{test})	E_{train} (σ_{train})	E_{test} (σ_{test})	E_{train} (σ_{train})	E_{test} (σ_{test})
15	29.0 (0.2)	26.8 (1.1)	24.4 (1.6)	21.5 (1.1)	25.8 (3.7)	25.2 (2.7)
20	28.9 (0.7)	27.0 (1.6)	24.1 (0.5)	21.1 (0.7)	22.0 (0.9)	21.1 (0.3)
25	28.9 (0.9)	26.3 (1.1)	24.5 (0.8)	20.6 (0.6)	21.1 (0.4)	21.2 (0.4)
30	28.9 (1.1)	26.6 (0.4)	24.1 (0.8)	20.3 (0.3)	21.2 (1.4)	20.9 (0.7)
35	28.3 (0.9)	25.8 (1.0)	24.3 (0.7)	20.3 (0.3)	20.8 (0.6)	21.0 (0.8)
40	28.0 (0.9)	25.2 (1.0)	24.0 (0.8)	20.5 (0.4)	20.3 (0.6)	20.6 (0.3)
45	27.4 (0.9)	25.4 (0.8)	24.0 (0.3)	20.0 (0.6)	20.0 (0.7)	21.1 (0.5)
50	27.8 (0.9)	25.3 (1.0)	24.5 (0.4)	20.1 (0.6)	19.9 (0.4)	21.1 (0.2)

TABLE III
PERCENTAGE OF CLASSIFICATION ERRORS PRODUCED ON THE AVERAGE IN 10 TRIALS ON THE TRAINING SET (E_{train}) AND THE TESTING SET (E_{test}) FORMED FROM THE 2-D VOWEL DATA BY CONVENTIONAL RBF NEURAL NETWORKS WITH c GAUSSIAN RADIAL BASIS FUNCTIONS AND REFORMULATED RBF NETWORKS CONTAINING c RADIAL BASIS FUNCTIONS OF THE FORM $\phi(x) = g(x^2)$, WITH $g(x) = (g_0(x))^{\frac{1}{1-m}}$ AND $m = 3$. THE NUMBER SHOWN IN PARENTHESIS IS THE STANDARD DEVIATION

	Conventional RBF NNs Gaussian basis functions		Reformulated RBF NNs $g_0(x) = x + \gamma^2$		Reformulated RBF NNs $g_0(x) = \exp(\beta x)$	
c	E_{train} (σ_{train})	E_{test} (σ_{test})	E_{train} (σ_{train})	E_{test} (σ_{test})	E_{train} (σ_{train})	E_{test} (σ_{test})
15	32.2 (0.8)	29.2 (2.2)	24.4 (1.2)	23.8 (0.9)	24.4 (1.6)	21.5 (1.1)
20	28.1 (2.2)	24.9 (3.2)	23.9 (1.4)	21.9 (0.6)	24.1 (0.5)	21.1 (0.7)
25	27.0 (1.3)	26.2 (0.7)	22.7 (0.8)	21.8 (1.0)	24.5 (0.8)	20.6 (0.6)
30	25.0 (0.8)	22.8 (1.2)	22.9 (0.8)	20.8 (0.9)	24.1 (0.8)	20.3 (0.3)
35	24.8 (0.8)	22.9 (0.4)	22.7 (0.2)	20.2 (0.7)	24.3 (0.7)	20.3 (0.3)
40	23.8 (2.2)	20.6 (1.2)	22.3 (0.4)	20.5 (0.8)	24.0 (0.8)	20.5 (0.4)
45	22.8 (0.9)	20.4 (0.3)	22.4 (0.5)	20.4 (0.2)	24.0 (0.3)	20.0 (0.6)
50	24.7 (0.2)	24.2 (0.9)	22.0 (0.8)	20.6 (0.4)	24.5 (0.4)	20.1 (0.6)

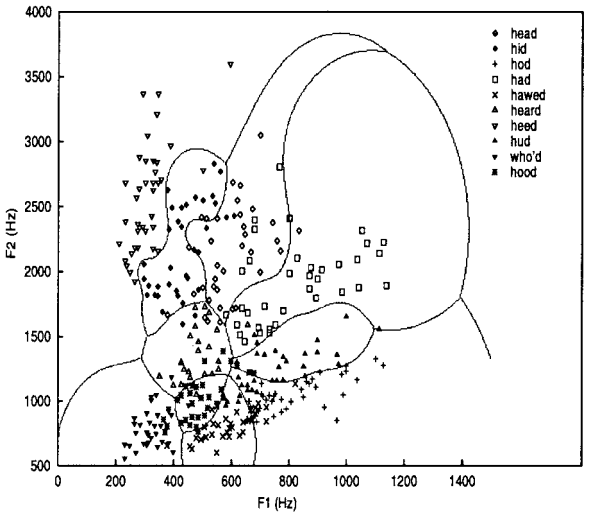
is plotted in Fig. 2(b) for $\gamma = 0$, $\gamma = 0.1$, and $\gamma = 1$. It is clear from Fig. 2(b) that the response of each RBF $h_{j,k}$ becomes increasingly sensitive to changes of $||\mathbf{x}_k - \mathbf{v}_j||^2$ as γ decreases and approaches zero.

Table II shows the percentage of classification errors produced on the average in ten trials on the training and testing sets formed from the 2-D vowel data by reformulated RBF networks generated by exponential generator functions $g_0(x) = \exp(\beta x)$ with $m = 3$ and $\beta = 0.5$, $\beta = 1$, $\beta = 5$. For $m = 3$, the values $\beta = 0.5$, $\beta = 1$ and $\beta = 5$ correspond to $\sigma^2 = 4$, $\sigma^2 = 2$ and $\sigma^2 = 0.4$, respectively. The networks were also trained with $\beta = 0.1$ ($\sigma^2 = 20$) and $\beta = 0.2$ ($\sigma^2 = 10$) but the learning algorithm did not converge and the networks classified incorrectly almost half of the input vectors from the training and testing sets. Table II indicates that even the networks trained with $\beta = 0.5$ did not achieve satisfactory performance. The performance of the trained networks on both training and testing sets improved as the value of β increased above 0.5. The best performance on the training set was achieved for $\beta = 5$. However, the best performance on the testing set was achieved for $\beta = 1$. It is remarkable that for $\beta = 1$ the percentage of classification errors on the testing set was almost constant as the number of RBF's increased above 20. The gradient descent algorithm also converged for values of β above five but the performance of the trained RBF networks on the testing set degraded. This set of experiments is consistent with the sensitivity analysis and reveals the tradeoff associated with the selection of the value of β or, equivalently, the width of the resulting Gaussian RBF's. Small values of β reduce the sensitivity of the RBF's to changes of their respective prototypes and the learning algorithm does not converge. Large values of β lead to RBF's that are more sensitive to changes of their respective prototypes. However, such values of β create sharp RBF's and that has a negative effect on the ability of the trained RBF networks to generalize. These experiments also indicate that there exists a certain range of values of β that guarantee convergence of the learning algorithm and satisfactory performance. The speed of convergence of the gradient descent algorithm increased slightly as the value of β increased from 0.5 to 5. The average number of adaptation cycles required in ten trials for the convergence of the gradient descent algorithm tested for $c = 30$ with $\beta = 0.5$, $\beta = 1$, and $\beta = 5$ was 16 595, 14 294, and 13 814, respectively.

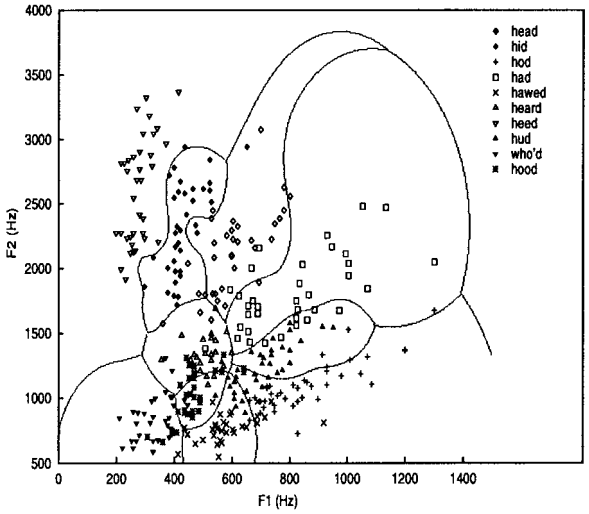
The next set of experiments compared the performance of conventional RBF neural networks with Gaussian RBF's and reformulated RBF neural networks generated by the linear generator function $g_0(x) = x + \gamma^2$ with $\gamma = 1$ and $m = 3$ and the exponential generator function $g_0(x) = \exp(\beta x)$ with $\beta = 1$ and $m = 3$. Each of these networks was tested with various numbers of RBF's in ten trials. The learning was initialized in these trials by different sets of randomly selected prototypes. In order to comparison to be fair, the same set of initial prototypes were used in each trial to initialize the training of the conventional and reformulated RBF networks with the same number of RBF's. For each of the networks tested, Table III shows the percentage of classification errors produced on the average in ten trials on the training and testing sets followed by the standard deviation (shown in parenthesis). The experiments verified that training speed is one of the major advantages of conventional RBF neural networks. The average number of adaptation cycles required in ten trials for the convergence of the learning scheme used to train conventional RBF neural networks for $c = 30$ was 2207. The experiments also revealed the erratic behavior that is often associated with RBF neural networks. There was a significant variation in the percentage of classification errors recorded on the training and testing sets as the number of RBF's increased from 15 to 50. When the number of RBF's was relatively small (from 15 to 25), the performance of the trained RBF networks on both training and testing sets was rather poor. Their performance improved as the number of RBF's increased above 30. Even in this case, however, the classification errors did not consistently decrease as the number of RBF's increased, a behavior that is reasonably expected at least for the training set. The performance of conventional RBF networks was significantly affected by the initialization of the learning process, as indicated by the fluctuations of the standard deviation values. This can be attributed to the significant impact of the initial set of prototypes on the partition of the input vectors produced by the k-means

algorithm. This partition determines the responses of the RBF's and plays a critical role in the ability of the network to implement the desired input–output mapping [14]. According to Table III, the percentage of classification errors produced on the average in ten trials by reformulated RBF neural networks on both the training and testing sets were not significantly affected by the number of RBF's. This allows the training of reformulated RBF neural networks with a small number of RBF's, which improves generalization. On the other hand, reformulated RBF neural networks were only slightly affected by the initialization process as indicated by the low values of the standard deviation of the classification errors produced by these networks in ten trials. This can be attributed to the fact that the prototypes of reformulated RBF neural networks are updated during learning as indicated by the training set. This makes reformulated RBF neural networks immune to "bad" sets of initial prototypes, which can severely affect the training and performance of conventional RBF neural networks. Compared with reformulated RBF networks generated by linear generator functions, reformulated RBF networks generated by exponential generator functions produced a higher percentage of classification errors on the training set and a slightly lower percentage of classification errors on the testing set. In fact, the reformulated RBF networks generated by linear generator functions achieved a more balanced performance on the training and testing sets.

Figs. 4 and 5 show the partition of the input space produced by reformulated RBF networks trained using gradient descent with 30 and 50 RBF's, respectively. The function $g(\cdot)$ was of the form $g(x) = (g_0(x))^{\frac{1}{m-1}}$, with $g_0(x) = x$ and $m = 3$. It is clear from Fig. 4(a) that the networks with 30 RBF's attempted to find the best possible compromise in regions of the input space with extensive overlapping between the classes. Nevertheless, the partitions produced by the trained network for the training set offer a reasonable compromise for the testing set as indicated by Fig. 4(b). Fig. 5(a) shows that the RBF network trained with 50 RBF's produced anomalous surfaces in its attempt to classify correctly input vectors from the training set that could be considered outliers with respect to their classes. Moreover, Fig. 5(b) indicates that the partition of the input space produced by the network for the training set is not appropriate for the testing set. This is a clear indication that increasing the number of RBF's above a certain threshold compromises the ability of trained RBF neural networks to generalize.



(a)

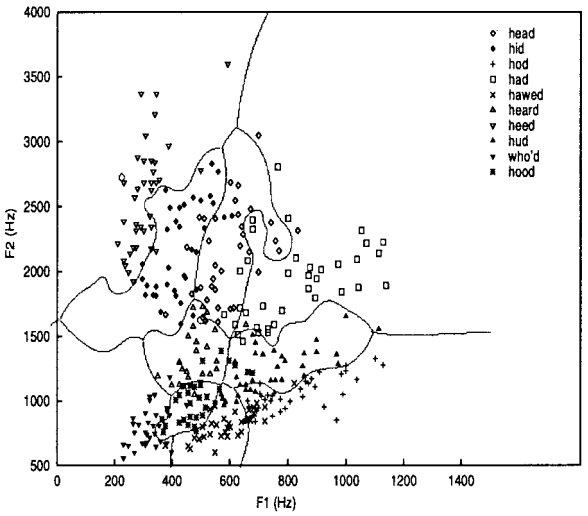


(b)

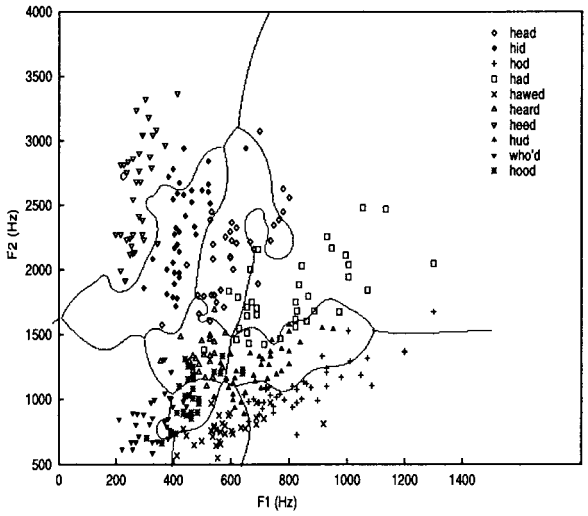
Fig. 4. Classification of the 2-D vowel data produced by a reformulated RBF neural network trained using gradient descent: (a) the training data set, and (b) the testing data set. Generator function: $g_0(x) = x$ with $m = 3$; number of RBF's: $c = 30$; learning rate: $\eta = 10^{-5}$.

B. IRIS Data

The reformulated RBF neural networks constructed and trained according to the approach proposed in this paper were also tested using Anderson's IRIS data set [1], which has extensively been used for evaluating the performance of clustering and classification algorithms. This data set contains 150 feature vectors of dimension four which belong to three physical classes representing different IRIS 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. All RBF neural networks were trained using a normalized version of the IRIS data obtained according to the same scheme used for normalizing the 2-D vowel data. The RBF networks tested in these experiments consisted of four inputs and three linear output units, each corresponding to a physical class. The number of RBF's varied in these experiments from three to five. The networks were trained to respond with $y_{i,k} = 1$ and $y_{j,k} = 0$, $\forall j \neq i$, when presented with an input vector $\mathbf{x}_k \in \mathcal{X}$ from the ith physical class. The assignment of input vectors to classes was based on a winner-takes-all strategy. The conventional RBF neural networks networks were trained using the same learning schemes employed in the experiments performed on the 2-D vowel data. The reformulated RBF networks were trained using the proposed gradient descent algorithm with learning rate $\eta = 10^{-2}$. This value of the



(a)



(b)

Fig. 5. Classification of the 2-D vowel data produced by a reformulated RBF neural network trained using gradient descent: (a) the training data set, and (b) the testing data set. Generator function: $g_0(x) = x$ with $m = 3$; number of RBF's: $c = 50$; learning rate: $\eta = 10^{-5}$.

TABLE IV
PERCENTAGE OF CLASSIFICATION ERRORS PRODUCED ON THE AVERAGE IN FIVE TRIALS ON THE TRAINING SET (E_{train}) AND THE TESTING SET (E_{test}) FORMED FROM THE IRIS DATA BY REFORMULATED RBF NETWORKS CONTAINING c RADIAL BASIS FUNCTIONS OF THE FORM $\phi(x) = g(x^2)$, WITH $g(x) = (g_0(x))^{\frac{1}{1-m}}$, $g_0(x) = x + \gamma^2$, $m = 3$, AND VARIOUS VALUES OF γ. THE NUMBER SHOWN IN PARENTHESIS IS THE STANDARD DEVIATION

	$\gamma = 0.0$ ($\eta = 10^{-2}$)		$\gamma = 0.1$ ($\eta = 10^{-2}$)		$\gamma = 1.0$ ($\eta = 10^{-2}$)	
c	E_{train} (σ_{train})	E_{test} (σ_{test})	E_{train} (σ_{train})	E_{test} (σ_{test})	E_{train} (σ_{train})	E_{test} (σ_{test})
3	3.2 (2.0)	5.6 (1.3)	3.2 (2.0)	5.6 (1.3)	2.9 (1.3)	5.1 (1.0)
4	2.4 (2.0)	3.7 (0.5)	2.4 (2.0)	3.7 (0.5)	2.4 (2.0)	4.8 (1.1)
5	0.8 (1.1)	5.3 (1.5)	0.8 (1.1)	5.6 (1.3)	0.5 (1.1)	5.1 (1.6)

learning rate guaranteed a monotonic reduction of the total error during the learning process.

Table IV summarizes the percentage of classification errors produced on the average in five trials on the training and testing sets formed from the IRIS data by reformulated RBF networks generated by linear generator functions $g_0(x) = x + \gamma^2$, with $m = 3$ and $\gamma = 0$, $\gamma = 0.1$, $\gamma = 1$. The learning process was initialized in each trial by a different set of randomly generated prototypes. The percentage of classification errors on the training set decreased consistently as the number of RBF's increased from three to five. On the other hand, the percentage of classification errors on the testing set decreased as the number of RBF's increased from three (the number of physical classes) to four but increased again when the number of RBF's increased from four to five. This is an indication that the generalization ability of the networks degraded as the number of RBF's increased despite the fact that the trained networks performed better on the training set. According to Table IV, the performance of the reformulated RBF networks was not significantly affected by the value of γ. Nevertheless, there was a slight reduction of the percentage of classification errors on the training set as the value of γ increased from zero to one. In addition, the reformulated RBF networks with $\gamma = 1$ produced almost the same number of classification errors on the testing set as the number of RBF's increased from three to five.

TABLE V
PERCENTAGE OF CLASSIFICATION ERRORS PRODUCED ON THE AVERAGE IN FIVE TRIALS ON THE TRAINING SET (E_{train}) AND THE TESTING SET (E_{test}) FORMED FROM THE IRIS DATA BY REFORMULATED RBF NETWORKS CONTAINING c RADIAL BASIS FUNCTIONS OF THE FORM $\phi(x) = g(x^2)$, WITH $g(x) = (g_0(x))^{\frac{1}{1-m}}$, $g_0(x) = \exp(\beta x)$, $m = 3$, AND VARIOUS VALUES OF β. THE NUMBER SHOWN IN PARENTHESIS IS THE STANDARD DEVIATION

	$\beta = 0.2$ ($\eta = 10^{-2}$)		$\beta = 0.5$ ($\eta = 10^{-2}$)		$\beta = 1.0$ ($\eta = 10^{-2}$)	
c	E_{train} (σ_{train})	E_{test} (σ_{test})	E_{train} (σ_{train})	E_{test} (σ_{test})	E_{train} (σ_{train})	E_{test} (σ_{test})
3	5.6 (1.0)	5.3 (1.2)	1.6 (0.5)	6.1 (0.6)	4.8 (1.3)	8.5 (1.1)
4	4.5 (0.6)	5.1 (1.3)	1.6 (1.3)	5.3 (1.2)	2.7 (2.1)	6.7 (1.2)
5	5.6 (1.3)	5.1 (0.5)	1.3 (0.8)	5.3 (0.0)	1.1 (0.5)	6.4 (1.1)

Table V shows the percentage of classification errors produced on the average in five trials on the training and testing sets formed from the IRIS data by reformulated RBF networks generated by exponential generator functions $g_0(x) = \exp(\beta x)$ with $m = 3$ and $\beta = 0.2$, $\beta = 0.5$, $\beta = 1$. In these experiments the gradient descent algorithm did not always converge for values of β below 0.2 or above one. Even when the algorithm converged, the performance of the trained networks was sensitively depended on the specific value of β used. The reformulated RBF networks trained with $\beta = 0.5$ resulted in the smallest percentage of classification errors on the training set. For $\beta = 0.5$ and $\beta = 1$, the classification errors on the training set decreased as the number of RBF's increased from three to five. However, the reformulated RBF networks trained with $\beta = 0.2$ did not exhibit the same behavior on the training set. The reformulated RBF networks trained with $\beta = 0.2$ and $\beta = 0.5$ achieved a satisfactory performance on the testing set. However, the performance of the trained RBF networks degraded as the value of β increased to $\beta = 1$. Table V indicates that the value of β has a significant effect on the ability of the trained RBF networks to generalize. Overall, the performance of the reformulated RBF networks tested in these experiments was inferior to that of reformulated RBF networks generated a linear generator functions.

TABLE VI
PERCENTAGE OF CLASSIFICATION ERRORS PRODUCED ON THE AVERAGE IN FIVE TRIALS ON THE TRAINING SET (E_{train}) AND THE TESTING SET (E_{test}) FORMED FROM THE IRIS DATA BY CONVENTIONAL RBF NEURAL NETWORKS WITH c GAUSSIAN RADIAL BASIS FUNCTIONS AND REFORMULATED RBF NETWORKS CONTAINING c RADIAL BASIS FUNCTIONS OF THE FORM $\phi(x) = g(x^2)$, WITH $g(x) = (g_0(x))^{\frac{1}{1-m}}$ AND $m = 3$. THE NUMBER SHOWN IN PARENTHESIS IS THE STANDARD DEVIATION

	Conventional RBF NNs Gaussian basis functions		Reformulated RBF NNs $g_0(x) = x + \gamma^2$		Reformulated RBF NNs $g_0(x) = \exp(\beta x)$	
c	E_{train} (σ_{train})	E_{test} (σ_{test})	E_{train} (σ_{train})	E_{test} (σ_{test})	E_{train} (σ_{train})	E_{test} (σ_{test})
3	16.8 (2.0)	16.8 (2.9)	2.9 (1.5)	5.1 (0.5)	4.8 (1.5)	8.5 (1.2)
4	15.5 (0.7)	15.0 (2.9)	2.4 (2.2)	4.8 (1.2)	2.7 (2.3)	6.7 (1.3)
5	14.1 (5.7)	13.9 (5.6)	0.5 (1.2)	5.1 (1.7)	1.0 (0.6)	6.4 (1.1)

Table VI summarizes the performance of conventional RBF neural networks and reformulated RBF neural networks generated by $g_0(x) = x + \gamma^2$ with $\gamma = 1$ and $m = 3$ and $g_0(x) = \exp(\beta x)$ with $\beta = 1$ and $m = 3$ on the training and testing sets formed from the IRIS data. The performance of the networks was measured by the percentage of classification errors recorded on the average in five trials and the corresponding standard deviation (shown in parenthesis). Overall, the performance of conventional RBF networks on both training and testing sets was unacceptable. Note that the percentage of classification errors produced by supervised classifiers on the IRIS data set is typically in the range 0%–7%. In fact, the conventional RBF networks trained to perform classification by a supervised learning scheme cannot even compete with unsupervised clustering algorithms, which typically produce clustering errors in the range 15%–20% when tested on the IRIS data set. The poor performance of conventional RBF networks in these experiments can be attributed to the well-documented failure of the k-means algorithm to produce acceptable partitions of the IRIS data [8]. This set of experiments is a clear indication that conventional RBF neural networks can implement nontrivial input–output mappings only if the unsupervised algorithm produces prototypes that constitute a satisfactory representation of the input vectors, which is not guaranteed for training sets encountered in practical applications.

VII. CONCLUSION

The success of a neural-network model depends rather strongly on its association with an attractive learning algorithm. For example, the popularity of feedforward neural networks with sigmoidal hidden units was to a large extent due to the error back propagation algorithm. On the other hand, the potential of RBF neural models for classification and function approximation was downgraded by the lack of effective and reliable learning algorithms for such models. The experiments presented in this paper verified that the erratic behavior and poor performance often associated with conventional RBF networks can only be attributed to the learning scheme employed for their training and not to RBF neural models.

According to the axiomatic approach proposed in this paper for reformulating RBF neural networks, the development of admissible RBF models reduces to the selection of admissible generator functions that determine the form and properties of the RBF's. The reformulated RBF networks generated by linear and exponential generator functions can be trained by gradient descent and perform considerably better than conventional RBF networks. Moreover, training reformulated RBF neural networks by gradient descent is not necessarily slow. Consider, for example, reformulated RBF networks generated by linear generator functions of the form $g_0(x) = x+\gamma^2$. Values of γ close to zero can speed up the convergence of gradient descent learning and still produce trained RBF networks performing better than conventional RBF networks. For values of γ approaching one, the convergence of gradient descent learning slows down but the performance of the trained RBF networks improves. The convergence of gradient descent learning is also affected by the value of the parameter m relative to one. Values of m close to one tend to produce sharp RBF's. As the value of m increases, the RBF's become wider and more responsive to neighboring input vectors. However, increasing the value of m reduces the sensitivity of the RBF's to changes of their respective prototypes and this slows down the convergence of gradient descent learning. Nevertheless, the selection of the parameter m is not crucial in practice. Reformulated RBF networks can be trained in practical situations by fixing the parameter m to a value between two and four.

One of the problems currently under investigation is the improvement of the convergence of gradient descent learning algorithms used to train reformulated RBF neural networks. This can be accomplished by searching for alternative forms of the linear generator function [11] and by employing alternative schemes for initializing the supervised training process. The most promising initialization scheme seems to be the formation of the initial set of prototypes by learning vector quantization algorithms generated by the same generator functions used to construct reformulated RBF neural networks [12]. In fact, the generator function can be seen as the concept that establishes a direct relationship between reformulated RBF models and fuzzy LVQ algorithms. This relationship makes reformulated RBF models potential targets of the search for architectures inherently capable of merging neural modeling with fuzzy-theoretic concepts, a problem that attracted considerable attention recently [23]. In this context, a problem worth investigating is the ability of reformulated RBF neural networks to detect the presence of uncertainty in the training set and quantify the existing uncertainty by approximating any membership profile arbitrarily well from sample data.

APPENDIX A

Proof of Theorem 1: Since $h_{j,k} = g(\|\mathbf{x}_k - \mathbf{v}_j\|^2)$,

$$\begin{aligned}\nabla_{\mathbf{x}_k} h_{j,k} &= \nabla_{\mathbf{x}_k} g(\|\mathbf{x}_k - \mathbf{v}_j\|^2) \\ &= g'(\|\mathbf{x}_k - \mathbf{v}_j\|^2)\nabla_{\mathbf{x}_k}(\|\mathbf{x}_k - \mathbf{v}_j\|^2) \\ &= -\alpha_{j,k}(\mathbf{x}_k - \mathbf{v}_j) \qquad \text{(A1)}\end{aligned}$$

where $\alpha_{j,k} = -2g'(\|\mathbf{x}_k - \mathbf{v}_j\|^2)$. From (A1)

$$\left\|\nabla_{\mathbf{x}_k} h_{j,k}\right\|^2 = \alpha_{j,k}^2 \|\mathbf{x}_k - \mathbf{v}_j\|^2. \qquad \text{(A2)}$$

1) According to the first axiomatic requirement, $g(\|\mathbf{x}_k - \mathbf{v}_j\|^2) > 0$ for all $\mathbf{x}_k \in \mathcal{X}$ and $\mathbf{v}_j \in \mathcal{V}$. This condition is satisfied iff $g(x) > 0, \forall x \in (0, \infty)$.

2) According to the second axiomatic requirement, $g(\|\mathbf{x}_k-\mathbf{v}_j\|^2) > g(\|\mathbf{x}_\ell-\mathbf{v}_j\|^2)$ for all $\mathbf{x}_k,\mathbf{x}_\ell \in \mathcal{X}$ and $\mathbf{v}_j \in \mathcal{V}$ such that $\|\mathbf{x}_k-\mathbf{v}_j\|^2 < \|\mathbf{x}_\ell-\mathbf{v}_j\|^2$. This condition is satisfied iff $g(x)$ is a decreasing function of $x \in (0,\infty)$, i.e., $g'(x) < 0,\ \forall x \in (0,\infty)$.
3) According to the third axiomatic requirement

$$\frac{\left\|\nabla_{\mathbf{x}_k} h_{j,k}\right\|^2}{\|\mathbf{x}_k-\mathbf{v}_j\|^2} = \alpha_{j,k}^2 > \alpha_{j,\ell}^2 = \frac{\left\|\nabla_{\mathbf{x}_\ell} h_{j,\ell}\right\|^2}{\|\mathbf{x}_\ell-\mathbf{v}_j\|^2} \tag{A3}$$

for all $\mathbf{x}_k,\mathbf{x}_\ell \in \mathcal{X}$ and $\mathbf{v}_j \in \mathcal{V}$ such that $\|\mathbf{x}_k-\mathbf{v}_j\|^2 < \|\mathbf{x}_\ell-\mathbf{v}_j\|^2$. According to the second axiomatic requirement, $\alpha_{j,k} = -2g'(\|\mathbf{x}_k-\mathbf{v}_j\|^2) > 0$ for all $\mathbf{x}_k \in \mathcal{X}$ and $\mathbf{v}_j \in \mathcal{V}$. For $\alpha_{j,k} > 0$ and $\alpha_{j,\ell} > 0$, the condition $\alpha_{j,k}^2 > \alpha_{j,\ell}^2$ is satisfied iff $\alpha_{j,k} > \alpha_{j,\ell}$. Since $\alpha_{j,k} = -2g'(\|\mathbf{x}_k-\mathbf{v}_j\|^2)$, (A3) is satisfied for all $\mathbf{x}_k,\mathbf{x}_\ell \in \mathcal{X}$ and $\mathbf{v}_j \in \mathcal{V}$ such that $\|\mathbf{x}_k-\mathbf{v}_j\|^2 < \|\mathbf{x}_\ell-\mathbf{v}_j\|^2$ iff $g'(\|\mathbf{x}_k-\mathbf{v}_j\|^2) < g'(\|\mathbf{x}_\ell-\mathbf{v}_j\|^2)$. Thus, the third axiomatic requirement is satisfied iff $g'(x)$ is a monotonically increasing function of $x \in (0,\infty)$, i.e., $g''(x) > 0, \forall x \in (0,\infty)$.

APPENDIX B

Proof of Theorem 2: If $g(x) = (g_0(x))^{\frac{1}{1-m}}$, then

$$g'(x) = \frac{1}{1-m}(g_0(x))^{\frac{m}{1-m}} g_0'(x) \tag{B1}$$

and

$$g''(x) = -\frac{1}{1-m}(g_0(x))^{\frac{m}{1-m}-1} r_0(x) \tag{B2}$$

where

$$r_0(x) = \frac{m}{m-1}(g_0'(x))^2 - g_0(x)g_0''(x). \tag{B3}$$

According to Theorem 1, $g(\cdot)$ leads to admissible RBF's if $g(x) > 0,\ \forall x \in (0,\infty)$, $g'(x) < 0,\ \forall x \in (0,\infty)$, and $g''(x) > 0,\ \forall x \in (0,\infty)$.

Consider the function $g(x) = (g_0(x))^{\frac{1}{1-m}}$, with $m > 1$.

1) The condition $g(x) > 0, x \in (0,\infty)$, is satisfied for all $m > 1$ if $g_0(x) > 0,\ \forall x \in (0,\infty)$.
2) For $m > 1$ $(1-m < 0)$, (B1) indicates that the condition $g'(x) < 0,\ \forall x \in (0,\infty)$, is satisfied if $g_0'(x) > 0,\ \forall x \in (0,\infty)$, that is, if $g_0(x)$ is a monotonically increasing function of $x \in (0,\infty)$.
3) According to (B2), the condition $g''(x) > 0,\ \forall x \in (0,\infty)$, is satisfied for $1-m < 0$ if

$$r_0(x) = \frac{m}{m-1}(g_0'(x))^2 - g_0(x)g_0''(x) > 0 \qquad \forall x \in (0,\infty). \tag{B4}$$

Consider the function $g(x) = (g_0(x))^{\frac{1}{1-m}}$, with $m < 1$.

1) The condition $g(x) > 0, x \in (0,\infty)$, is satisfied for all $m < 1$ if $g_0(x) > 0,\ \forall x \in (0,\infty)$.
2) For $m < 1$ $(1-m > 0)$, (B1) indicates that the condition $g'(x) < 0,\ \forall x \in (0,\infty)$, is satisfied if $g_0'(x) < 0, \forall x \in (0,\infty)$, that is, if $g_0(x)$ is a monotonically decreasing function of $x \in (0,\infty)$.
3) According to (B2), the condition $g''(x) > 0,\ \forall x \in (0,\infty)$, is satisfied for $1-m > 0$ if

$$r_0(x) = \frac{m}{m-1}(g_0'(x))^2 - g_0(x)g_0''(x) < 0 \qquad \forall x \in (0,\infty). \tag{B5}$$

APPENDIX C

Derivation of the Learning Algorithm: Using the definition of E in (24)

$$\nabla_{\mathbf{w}_p} E = -\sum_{k=1}^{M}\sum_{i=1}^{n_o}(y_{i,k}-\hat{y}_{i,k})\nabla_{\mathbf{w}_p}\hat{y}_{i,k} \tag{C1}$$

where

$$\begin{aligned}\nabla_{\mathbf{w}_p}\hat{y}_{i,k} &= \nabla_{\mathbf{w}_p}\left(\mathbf{w}_i^T\mathbf{h}_k\right)\\ &= \mathbf{h}_k\delta_{ip}\end{aligned} \tag{C2}$$

and δ_{ip} denotes the Kronecker delta, defined as $\delta_{ip} = 1$ if $i = p$ and $\delta_{ip} = 0$ if $i \neq p$. Substituting (C2) in (C1) gives

$$\nabla_{\mathbf{w}_p} E = -\sum_{k=1}^{M}\varepsilon_{p,k}^o\mathbf{h}_k \tag{C3}$$

where $\varepsilon_{p,k}^o = y_{p,k} - \hat{y}_{p,k}$.

The gradient of E with respect to the prototype $\mathbf{v}_q$ is

$$\nabla_{\mathbf{v}_q} E = -\sum_{k=1}^{M}\sum_{i=1}^{n_o}(y_{i,k}-\hat{y}_{i,k})\nabla_{\mathbf{v}_q}\hat{y}_{i,k}. \tag{C4}$$

Since $\nabla_{\mathbf{v}_q} h_{j,k} = 0$ for $j \neq q$,

$$\begin{aligned}\nabla_{\mathbf{v}_q}\hat{y}_{i,k} &= \nabla_{\mathbf{v}_q}\left(\sum_{j=0}^{c} w_{ij}h_{j,k}\right)\\ &= w_{iq}\nabla_{\mathbf{v}_q}h_{q,k}.\end{aligned} \tag{C5}$$

Using that $h_{q,k} = g(\|\mathbf{x}_k-\mathbf{v}_q\|^2)$,

$$\begin{aligned}\nabla_{\mathbf{v}_q}h_{q,k} &= \nabla_{\mathbf{v}_q}g(\|\mathbf{x}_k-\mathbf{v}_q\|^2)\\ &= g'(\|\mathbf{x}_k-\mathbf{v}_q\|^2)\nabla_{\mathbf{v}_q}(\|\mathbf{x}_k-\mathbf{v}_q\|^2)\\ &= \alpha_{q,k}(\mathbf{x}_k-\mathbf{v}_q)\end{aligned} \tag{C6}$$

where $\alpha_{q,k} = -2g'(\|\mathbf{x}_k-\mathbf{v}_q\|^2)$. Combining (C5) and (C6) gives

$$\nabla_{\mathbf{v}_q}\hat{y}_{i,k} = w_{iq}\alpha_{q,k}(\mathbf{x}_k-\mathbf{v}_q). \tag{C7}$$

Substituting (C7) in (C4) gives

$$\begin{aligned}\nabla_{\mathbf{v}_q} E &= -\sum_{k=1}^{M}\sum_{i=1}^{n_o}(y_{i,k}-\hat{y}_{i,k})w_{iq}\alpha_{q,k}(\mathbf{x}_k-\mathbf{v}_q)\\ &= -\sum_{k=1}^{M}\varepsilon_{q,k}^h(\mathbf{x}_k-\mathbf{v}_q)\end{aligned} \tag{C8}$$

where

$$\varepsilon_{q,k}^h = \alpha_{q,k}\sum_{i=1}^{n_o}\varepsilon_{i,k}^o w_{iq}. \tag{C9}$$

REFERENCES

[1] E. Anderson, "The IRISes of the Gaspe Peninsula," *Bull. Amer. IRIS Soc.*, vol. 59, pp. 2–5, 1939.

[2] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*. New York: Plenum, 1981.

[3] J. C. Bezdek and N. R. Pal, "Two soft relatives of learning vector quantization," *Neural Networks*, vol. 8, no. 5, pp. 729–743, 1995.

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

[5] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Math. Control, Signals, Syst.*, vol. 2, pp. 303–314, 1989.

[6] R. J. Hathaway and J. C. Bezdek, "Optimization of clustering criteria by reformulation," *IEEE Trans. Fuzzy Syst.*, vol. 3, pp. 241–246, 1995.

[7] N. B. Karayiannis, "Fuzzy partition entropies and entropy constrained fuzzy clustering algorithms," *J. Intel. Fuzzy Syst.*, vol. 5, pp. 103–111, 1997.

[8] ———, "Learning vector quantization: A review," *Int. J. Smart Eng. Syst. Design*, vol. 1, pp. 33–58, 1997.

[9] ———, "Entropy constrained learning vector quantization algorithms and their application in image compression," *SPIE Proc., vol. 3030: Applications of Artificial Neural Networks in Image Processing II*, San Jose, CA, Feb. 8–14, 1997, pp. 2–13.

[10] ———, "Gradient descent learning of radial basis neural networks," in *Proc. 1997 IEEE Int. Conf. Neural Networks*, Houston, TX, June 9–12, 1997, pp. 1815–1820.

[11] ———, "Learning algorithms for reformulated radial basis neural networks," in *Proc. 1998 IEEE Int. Joint Conf. Neural Networks*, Anchorage, AK, May 4–9, 1998, pp. 2230–2235.

[12] ———, "Soft learning vector quantization and clustering algorithms based on reformulation," in *Proc. 1998 IEEE Int. Conf. Fuzzy Syst.*, Anchorage, AK, May 4–9, 1998, pp. 1441–1446.

[13] N. B. Karayiannis and J. C. Bezdek, "An integrated approach to fuzzy learning vector quantization and fuzzy c-means clustering," *IEEE Trans. Fuzzy Syst.*, vol. 5, pp. 622–628, 1997.

[14] N. B. Karayiannis and G. W. Mi, "Growing radial basis neural networks: Merging supervised and unsupervised learning with network growth techniques," *IEEE Trans. Neural Networks*, vol. 8, pp. 1492–1506, 1997.

[15] N. B. Karayiannis and A. N. Venetsanopoulos, *Artificial Neural Networks: Learning Algorithms, Performance Evaluation, and Applications*. Boston, MA: Kluwer, 1993.

[16] C. A. Micchelli, "Interpolation of scattered data: Distance matrices and conditionally positive definite functions," *Constructive Approximation*, vol. 2, pp. 11–22, 1986.

[17] R. P. Lippmann, "Pattern classification using neural networks," *IEEE Commun. Mag.*, vol. 27, pp. 47–54, 1989.

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

[19] 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. San Mateo, CA: Morgan Kaufmann, pp. 970–976, 1991.

[20] J. Park and I. W. Sandberg, "Universal approximation using radial-basis-function networks," *Neural Comput.*, vol. 3, pp. 246–257, 1991.

[21] J. Park and I. W. Sandberg, "Approximation and radial-basis-function networks," *Neural Comput.*, vol. 5, pp. 305–316, 1993.

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

[23] G. Purushothaman and N. B. Karayiannis, "Quantum neural networks (QNN's): Inherently fuzzy feedforward neural networks," *IEEE Trans. Neural Networks*, vol. 8, pp. 679–693, 1997.

[24] E. C.-K. Tsao, J. C. Bezdek, and N. R. Pal, "Fuzzy Kohonen clustering networks," *Pattern Recognition*, vol. 27, no. 5, pp. 757–764, 1994.

Nicolaos B. Karayiannis (S'85–M'91) was born in Greece on January 1, 1960. He received the Diploma degree in electrical engineering from the National Technical University of Athens, Greece, in 1983, and the M.A.Sc. and Ph.D. degrees in electrical engineering from the University of Toronto, Canada, in 1987 and 1991, respectively.

From 1983 to 1984 he was a Research Assistant at the Nuclear Research Center "Democritos," Athens, Greece, where he was engaged in research on multidimensional signal processing. From 1984 to 1991 he worked as a Research and Teaching Assistant at the University of Toronto. He is currently an Associate Professor in the Department of Electrical and Computer Engineering, University of Houston, TX. He has published more than 80 papers, including 33 in technical journals, and is the coauthor of the book *Artificial Neural Networks: Learning Algorithms, Performance Evaluation, and Applications* (Boston, MA: Kluwer, 1993). His current research interests include supervised and unsupervised learning, applications of fuzzy logic in neural modeling, applications of artificial neural networks in image processing and communications, and learning vector quantization and its applications in image and video compression.

Dr. Karayiannis is the recipient of the W. T. Kittinger Outstanding Teacher Award. He is also a corecipient of a Theoretical Development Award for a paper presented at the Artificial Neural Networks in Engineering'94 Conference. He is an Associate Editor of the IEEE TRANSACTIONS ON NEURAL NETWORKS and the IEEE TRANSACTIONS ON FUZZY SYSTEMS. He also served as the General Chair of the 1997 International Conference on Neural Networks (ICNN'97), held in Houston, TX, on June 9–12, 1997. He is a member of the International Neural Network Society (INNS) and the Technical Chamber of Greece.