CHAPTER 2

NEW RADIAL BASIS NEURAL NETWORKS AND THEIR APPLICATION IN A LARGE-SCALE HANDWRITTEN DIGIT RECOGNITION PROBLEM

N.B. Karayiannis
Department of Electrical and Computer Engineering
University of Houston
Houston, Texas 77204-4793
U.S.A.
Karayiannis@UH.EDU

S. Behnke
Institute of Computer Science
Free University of Berlin
Takustr. 9, 14195 Berlin
Germany
behnke@inf.fu-berlin.de

This chapter presents an axiomatic approach for reformulating radial basis function (RBF) neural networks. With this approach the construction of admissible RBF models is reduced to the selection of generator functions that satisfy certain properties. The selection of specific generator functions is based on criteria which relate to their behavior when the training of reformulated RBF networks is performed by gradient descent. This chapter also presents batch and sequential learning algorithms developed for reformulated RBF networks using gradient descent. These algorithms are used to train reformulated RBF networks to recognize handwritten digits from the NIST databases.

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. From this point of view, learning is equivalent to finding a surface in a multidimensional space that provides the best fit for 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 [3] were the first to explore the use of radial basis functions in the design of neural networks and to show how RBF neural networks model nonlinear relationships and implement interpolation. Micchelli [33] showed that RBF neural networks can produce an interpolating surface which exactly passes through all the pairs of the training set. However, the exact fit is neither useful nor desirable in practice as it may produce anomalous interpolation surfaces. Poggio and Girosi [38] 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 uniquely reconstruct 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 [36], [37] proved that RBF neural networks with one layer of radial basis functions are capable of universal approximation. Under certain mild conditions on radial basis functions, RBF networks are capable of approximating arbitrarily well any function. Similar proofs also exist in the literature for feed-forward neural models with sigmoidal nonlinearities [7].

The performance of a RBF neural network depends on the number and positions of the radial basis functions, their shapes, 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 radial basis function centers randomly from the training data [3], 2) strategies employing unsupervised procedures for selecting radial basis function centers [5], [6], [25], [34], and 3) strategies employing su-

pervised procedures for selecting radial basis function centers [4], [13], [17], [20], [21], [38].

Broomhead and Lowe [3] suggested that, in the absence of *a priori* knowledge, the centers of the radial basis functions can either be distributed uniformly within the region of the input space for which there is data, or chosen to be a subset of training points by analogy with strict interpolation. This approach is sensible only if the training data are distributed in a representative manner for the problem under consideration, an assumption that is very rarely satisfied in practical applications. Moody and Darken [34] proposed a hybrid learning process for training RBF neural networks with Gaussian radial basis functions, which is widely used in practice. This learning procedure employs different schemes for updating the *output weights*, i.e., the weights that connect the radial basis functions with the output units, and the centers of the radial basis functions, i.e., the vectors in the input space that represent the *prototypes* of the input vectors included in the training set. Moody and Darken used the c-means (or k-means) clustering algorithm [2] and a "P-nearest-neighbor" heuristic to determine the positions and widths of the Gaussian radial basis functions, respectively. The output weights are updated using a supervised least-mean-squares learning rule. Poggio and Girosi [38] proposed a fully supervised approach for training RBF neural networks with Gaussian radial basis functions, which updates the radial basis function centers together with the output weights. Poggio and Girosi used Green's formulas to deduct an optimal solution with respect to the objective function and employed gradient descent to approximate the regularized solution. They also proposed that Kohonen's self-organizing feature map [29], [30] can be used for initializing the radial basis function centers before gradient descent is used to adjust all of the free parameters of the network. Chen *et al.* [5], [6] proposed a learning procedure for RBF neural networks based on the *orthogonal least squares* (OLS) method. The OLS method is used as a forward regression procedure to select a suitable set of radial basis function centers. In fact, this approach selects radial basis function centers one by one until an adequate RBF neural network has been constructed. Cha and Kassam [4] proposed a stochastic gradient training algorithm for RBF neural networks with Gaussian radial basis functions. This algorithm uses gradient descent to update all free parameters of RBF neural networks, which

include the radial basis function centers, the widths of the Gaussian radial basis functions, and the output weights. Whitehead and Choate [42] proposed an evolutionary training algorithm for RBF neural networks. In this approach, the centers of radial basis functions are governed by space-filling curves whose parameters evolve genetically. This encoding causes each group of co-determined basis functions to evolve in order to fit a region of the input space. Roy *et al.* [40] proposed a set of learning principles that led to a training algorithm for a network that contains "truncated" radial basis functions and other types of hidden units. This algorithm uses random clustering and linear programming to design and train the network with polynomial time complexity.

Despite the existence of a variety of learning schemes, RBF neural networks are frequently trained in practice using variations of the learning scheme proposed by Moody and Darken [34]. These hybrid learning schemes determine separately the prototypes that represent the radial basis function centers according to some *unsupervised* clustering or vector quantization algorithm and update the output weights by a *supervised* procedure to implement the desired input-output mapping. These approaches were developed as a natural reaction to the long training times typically associated with the training of traditional feed-forward neural networks using gradient descent [28]. In fact, these hybrid learning schemes achieve fast training of RBF neural networks as a result of the strategy they employ for learning the desired input-output mapping. However, the same strategy prevents the training set from participating in the formation of the radial basis function centers, with a negative impact on the performance of trained RBF neural networks [25]. This created a wrong impression about the actual capabilities of an otherwise powerful neural model. The training of RBF neural networks using gradient descent offers a solution to the trade-off between performance and training speed. Moreover, such training can make RBF neural networks serious competitors to feed-forward neural networks with sigmoidal hidden units.

Learning schemes attempting to train RBF neural networks by fixing the locations of the radial basis function centers are very slightly affected by the specific form of the radial basis functions used. On the other hand, the convergence of gradient descent learning and the performance of the

trained RBF neural networks are both affected rather strongly by the choice of radial basis functions. The search for admissible radial basis functions other than the Gaussian function motivated the development of an axiomatic approach for constructing reformulated RBF neural networks suitable for gradient descent learning [13], [17], [20], [21].

2 Function Approximation Models and RBF Neural Networks

There are many similarities between RBF neural networks and function approximation models used to perform interpolation. Such function approximation models attempt to determine a surface in a Euclidean space $\mathbb{R}^{n_i}$ that provides the best fit for the data $(\mathbf{x}_k, y_k)$, $1 \leq k \leq M$, where $\mathbf{x}_k \in \mathcal{X} \subset \mathbb{R}^{n_i}$ and $y_k \in \mathbb{R}$ for all $k = 1, 2, \ldots M$. Micchelli [33] considered the solution of the interpolation problem $s(\mathbf{x}_k) = y_k, 1 \leq k \leq M$, by functions $s : \mathbb{R}^{n_i} \rightarrow \mathbb{R}$ of the form:

$$s(\mathbf{x}) = \sum_{k=1}^{M} w_k \; g(\|\mathbf{x} - \mathbf{x}_k\|^2). \tag{1}$$

This formulation treats interpolation as a function approximation problem, with the function $s(\cdot)$ generated by the fitting procedure as the best approximation to this mapping. Given the form of the basis function $g(\cdot)$, the function approximation problem described by $s(\mathbf{x}_k) = y_k, \; 1 \leq k \leq M$, reduces to determining the weights $w_k, \; 1 \leq k \leq M$, associated with the model (1).

The model described by equation (1) is admissible for interpolation if the basis function $g(\cdot)$ satisfies certain conditions. Micchelli [33] showed 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 $(-1)^\ell \, g^{(\ell)}(x) \geq 0, \; \forall x \in (0, \infty)$, for $\ell = 0, 1, 2, \ldots$.

RBF neural network models can be viewed as the natural extension of this formulation. Consider the function approximation model described

by:

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

If the function $g(\cdot)$ satisfies certain conditions, the model (2) can be used to implement a desired mapping $\mathbb{R}^{n_i} \rightarrow \mathbb{R}$ specified by the training set $(\mathbf{x}_k, y_k)$, $1 \leq k \leq M$. This is usually accomplished by devising a learning procedure to determine its adjustable parameters. In addition to the weights w_j, $0 \leq j \leq c$, the adjustable parameters of the model (2) also include the vectors $\mathbf{v}_j \in \mathcal{V} \subset \mathbb{R}^{n_i}$, $1 \leq j \leq c$. These vectors are determined during learning as the prototypes of the input vectors $\mathbf{x}_k$, $1 \leq k \leq M$. The adjustable parameters of the model (2) are frequently updated by minimizing some measure of the discrepancy between the expected output y_k of the model to the corresponding input $\mathbf{x}_k$ and its actual response:

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

for all pairs $(\mathbf{x}_k, y_k)$, $1 \leq k \leq M$, included in the training set.

The function approximation model (2) can be extended to implement any mapping $\mathbb{R}^{n_i} \rightarrow \mathbb{R}^{n_o}$, $n_o \geq 1$, as:

$$\hat{y}_i = f\left(w_{i0} + \sum_{j=1}^{c} w_{ij} \, g(\|\mathbf{x} - \mathbf{v}_j\|^2)\right), 1 \leq i \leq n_o, \tag{4}$$

where $f(\cdot)$ is a non-decreasing, continuous and differentiable everywhere function. The model (4) describes a RBF neural network with inputs from $\mathbb{R}^{n_i}$, c radial basis function units, and n_o output units if:

$$g(x^2) = \phi(x), \tag{5}$$

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:

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

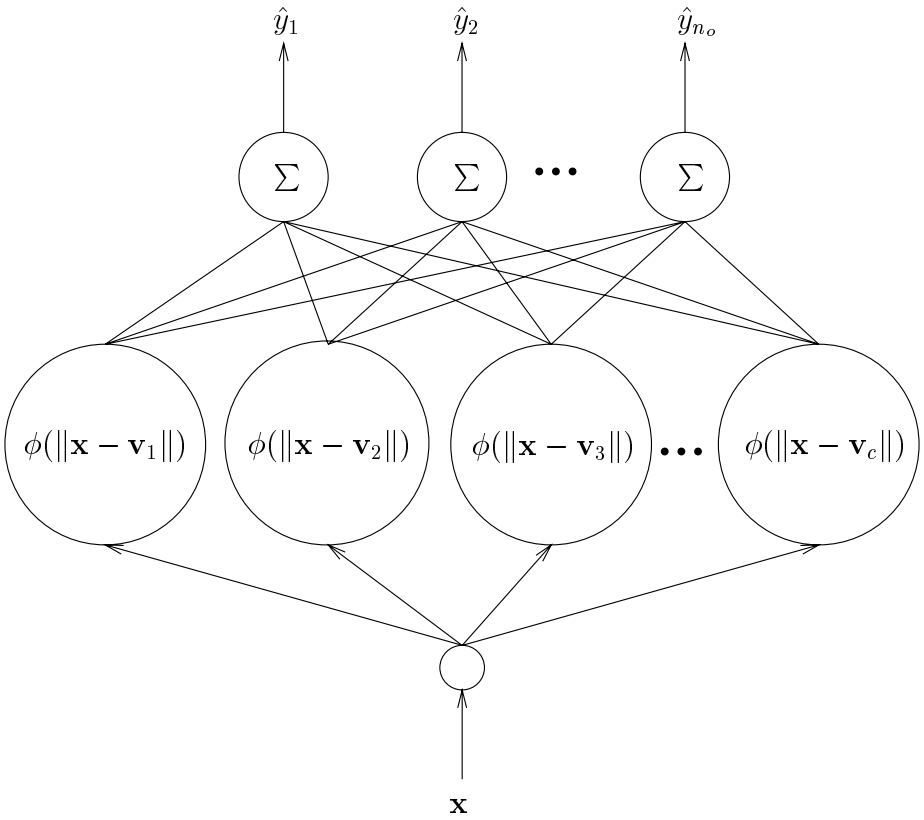


Figure 1. A radial basis function neural network.

where $h_{0,k} = 1, \forall k$, and $h_{j,k}$ represents the response of the radial basis function located at the jth prototype to the input vector $\mathbf{x}_k$, that is,

$$\begin{aligned} h_{j,k} &= \phi(\|\mathbf{x}_k - \mathbf{v}_j\|) \\ &= g(\|\mathbf{x}_k - \mathbf{v}_j\|^2),\ 1 \leq j \leq c. \end{aligned} \tag{7}$$

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 neural network is presented with $\mathbf{x}_k$, the input of the upper associative network is formed by the responses (7) of the radial basis functions located at the prototypes $\mathbf{v}_j, 1 \leq j \leq c$, as shown in Figure 1.

The models used in practice to implement RBF neural networks usually contain linear output units. An RBF model with linear output units can be seen as a special case of the model (4) that corresponds to $f(x) = x$. The

choice of linear output units was mainly motivated by the hybrid learning schemes originally developed for training RBF neural networks. Nevertheless, the learning process is only slightly affected by the form of $f(\cdot)$ if RBF neural networks are trained using learning algorithms based on gradient descent. Moreover, the form of an admissible function $f(\cdot)$ does not affect the function approximation capability of the model (4) or the conditions that must be satisfied by radial basis functions. Finally, the use of a nonlinear sigmoidal function $f(\cdot)$ could make RBF models stronger competitors to traditional feed-forward neural networks in certain applications, such as those involving pattern classification.

3 Reformulating Radial Basis Neural Networks

A 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 radial basis functions used. The interpretation often associated with RBF neural networks imposes some implicit constraints on the selection of radial basis functions. For example, RBF neural networks often employ decreasing Gaussian radial basis functions despite the fact that there exist both increasing and decreasing radial basis functions. The "neural" interpretation of the model (4) can be the basis of a systematic search for radial basis functions to be used for reformulating RBF neural networks [13], [17], [20], [21]. Such a systematic search is based on mathematical restrictions imposed on radial basis functions by their role in the formation of receptive fields.

The interpretation of a RBF neural network as a composition of receptive fields 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 an input vector as this input vector approaches its corresponding prototype.

Let $h_{j,k} = g\left(\|\mathbf{x}_k - \mathbf{v}_j\|^2\right)$ be the response of the jth radial basis function of a RBF neural network to the input $\mathbf{x}_k$. According to the above interpretation of RBF neural networks, any admissible radial basis function $\phi(x) = g(x^2)$ must satisfy the following three axiomatic requirements [13], [17], [20], [21]:

Axiom 1: $h_{j,k} > 0$ for all $\mathbf{x}_k \in \mathcal{X}$ and $\mathbf{v}_j \in \mathcal{V}$.

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

Axiom 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:

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

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

These basic axiomatic requirements impose some rather mild mathematical restrictions on the search for admissible radial basis functions. 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 radial basis functions 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:

$$\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2 > \|\nabla_{\mathbf{x}_\ell} h_{j,\ell}\|^2 \tag{8}$$

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

$$\frac{\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2}{\|\mathbf{x}_k - \mathbf{v}_j\|^2} > \frac{\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2}{\|\mathbf{x}_\ell - \mathbf{v}_j\|^2}. \tag{9}$$

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:

$$\frac{\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2}{\|\mathbf{x}_k - \mathbf{v}_j\|^2} > \frac{\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2}{\|\mathbf{x}_\ell - \mathbf{v}_j\|^2} > \frac{\|\nabla_{\mathbf{x}_\ell} h_{j,\ell}\|^2}{\|\mathbf{x}_\ell - \mathbf{v}_j\|^2}, \tag{10}$$

and the third axiomatic requirement is satisfied. This implies that condition (8) is stronger than that imposed by the third axiomatic requirement.

The above discussion suggests two complementary axiomatic requirements for radial basis functions [17]:

Axiom 4: $h_{j,k} < \infty$ for all $\mathbf{x}_k \in \mathcal{X}$ and $\mathbf{v}_j \in \mathcal{V}$.

Axiom 5: 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:

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

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 admissible radial basis functions can be facilitated by the following theorem [17]:

Theorem 1: The model described by equation (4) represents a RBF neural network in accordance with all five axiomatic requirements if and only if $g(\cdot)$ 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., $g''(x) > 0, \forall x \in (0, \infty)$,
4. $\lim_{x \to 0_+} g(x) = L$, where L is a finite number.
5. $d(x) = g'(x) + 2\,x\,g''(x) > 0, \forall x \in (0, \infty)$.

A radial basis function is said to be *admissible in the wide sense* if it satisfies the three basic axiomatic requirements, that is, the first three conditions of Theorem 1 [13], [17], [20], [21]. If a radial basis function satisfies all five axiomatic requirements, that is, all five conditions of Theorem 1, then it is said to be *admissible in the strict sense* [17].

A systematic search for admissible radial basis functions can be facilitated by considering basis functions of the form $\phi(x) = g(x^2)$, with $g(\cdot)$ defined in terms of a *generator function* $g_0(\cdot)$ as $g(x) = (g_0(x))^{\frac{1}{1-m}}$, $m \neq 1$ [13], [17], [20], [21]. The selection of generator functions that lead to admissible radial basis functions can be facilitated by the following theorem [17]:

Theorem 2: Consider the model (4) and let $g(x)$ be defined in terms of the generator function $g_0(x)$ that is continuous on $(0, \infty)$ as:

$$g(x) = (g_0(x))^{\frac{1}{1-m}}, m \neq 1. \tag{11}$$

If $m > 1$, then this model represents a RBF neural network in accordance with all five 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. $r_0(x) = \frac{m}{m-1} (g_0'(x))^2 - g_0(x)\, g_0''(x) > 0$, $\forall x \in (0, \infty)$,
4. $\lim_{x \to 0_+} g_0(x) = L_1 > 0$,
5. $d_0(x) = g_0(x)\, g_0'(x) - 2\, x\, r_0(x) < 0$, $\forall x \in (0, \infty)$.

If $m < 1$, then this model represents a RBF neural network in accordance with all five 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. $r_0(x) = \frac{m}{m-1} (g_0'(x))^2 - g_0(x)\, g_0''(x) < 0$, $\forall x \in (0, \infty)$,
4. $\lim_{x \to 0_+} g_0(x) = L_2 < \infty$,
5. $d_0(x) = g_0(x)\, g_0'(x) - 2\, x\, r_0(x) > 0$, $\forall x \in (0, \infty)$.

Any generator function that satisfies the first three conditions of Theorem 2 leads to admissible radial basis functions in the wide sense [13], [17], [20], [21]. Admissible radial basis functions in the strict sense can be obtained from generator functions that satisfy all five conditions of Theorem 2 [17].

4 Admissible Generator Functions

This section investigates the admissibility in the wide and strict sense of linear and exponential generator functions.

4.1 Linear Generator Functions

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

$$r_0(x) = \frac{m}{m-1}\,a^2. \tag{12}$$

If $m > 1$, then $r_0(x) > 0, \forall x \in (0, \infty)$. Thus, $g_0(x) = a\,x + b$ is an admissible generator function in the wide sense (i.e., in the sense that it satisfies the three basic axiomatic requirements) for all $a > 0$ and $b \geq 0$. Certainly, all combinations of $a > 0$ and $b > 0$ also lead to admissible generator functions in the wide sense.

For $g_0(x) = a\,x + b$, the fourth axiomatic requirement is satisfied if:

$$\lim_{x \to 0_+} g_0(x) = b > 0. \tag{13}$$

For $g_0(x) = a\,x + b$,

$$d_0(x) = (a\,x + b)\,a - 2\,x\,\frac{m}{m-1}\,a^2. \tag{14}$$

If $m > 1$, the fifth axiomatic requirement is satisfied if $d_0(x) < 0, \forall x \in (0, \infty)$. For $a > 0$, the condition $d_0(x) < 0$ is satisfied by $g_0(x) = a\,x + b$ if:

$$x > \frac{m-1}{m+1}\,\frac{b}{a}. \tag{15}$$

Since $m > 1$, the fifth axiomatic requirement is satisfied only if $b = 0$ or, equivalently, if $g_0(x) = a\,x$. However, the value $b = 0$ violates the fourth axiomatic requirement. Thus, there exists no combination of $a > 0$ and $b > 0$ leading to an admissible generator function in the strict sense that has the form $g_0(x) = a\,x + b$.

If $a = 1$ and $b = \gamma^2$, then the linear generator function $g_0(x) = a\,x + b$ 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 radial basis function:

$$\phi(x) = g(x^2) = \frac{1}{(x^2 + \gamma^2)^{\frac{1}{2}}}. \tag{16}$$

For $g_0(x) = x + \gamma^2$, $\lim_{x\to 0_+} g_0(x) = \gamma^2$ and $\lim_{x\to 0_+} g(x) = \gamma^{\frac{2}{1-m}}$. Since $m > 1$, $g(\cdot)$ is a bounded function if γ takes nonzero values. However, the bound of $g(\cdot)$ increases and approaches infinity as γ decreases and approaches 0. If $m > 1$, the condition $d_0(x) < 0$ is satisfied by $g_0(x) = x + \gamma^2$ if:

$$x > \frac{m-1}{m+1}\,\gamma^2. \tag{17}$$

Clearly, the fifth axiomatic requirement is satisfied only for $\gamma = 0$, which leads to an unbounded function $g(\cdot)$ [13], [20], [21].

Another useful generator function for practical applications can be obtained from $g_0(x) = a\,x + b$ by selecting $b = 1$ and $a = \delta > 0$. For $g_0(x) = 1 + \delta\,x$, $\lim_{x\to 0_+} g(x) = \lim_{x\to 0_+} g_0(x) = 1$. For this choice of parameters, the corresponding radial basis function $\phi(x) = g(x^2)$ is bounded by 1, which is also the bound of the Gaussian radial basis function. If $m > 1$, the condition $d_0(x) < 0$ is satisfied by $g_0(x) = 1 + \delta\,x$ if:

$$x > \frac{m-1}{m+1}\,\frac{1}{\delta}. \tag{18}$$

For a fixed $m > 1$, the fifth axiomatic requirement is satisfied in the limit $\delta \to \infty$. Thus, a reasonable choice for δ in practical situations is $\delta \gg 1$.

The radial basis function that corresponds to the linear generator function $g_0(x) = a\,x + b$ and some value of $m > 1$ can also be obtained from the

decreasing function $g_0(x) = 1/(a\,x + b)$ combined with an appropriate value of $m < 1$. As an example, for $m = 3$, the generator function $g_0(x) = a\,x + b$ leads to $g(x) = (a\,x + b)^{-\frac{1}{2}}$. For $a = 1$ and $b = \gamma^2$, this generator function corresponds to the multiquadratic radial basis function (16). The multiquadratic radial basis function (16) can also be obtained using the decreasing generator function $g_0(x) = 1/(x+\gamma^2)$ with $m = -1$. In general, the function $g(x) = (g_0(x))^{\frac{1}{1-m}}$ corresponding to the increasing generator function $g_0(x) = a\,x + b$ and $m = m_i > 1$, is identical with the function $g(x) = (g_0(x))^{\frac{1}{1-m}}$ corresponding to the decreasing function $g_0(x) = 1/(a\,x + b)$ and $m = m_d$ if:

$$\frac{1}{1 - m_i} = \frac{1}{m_d - 1}, \tag{19}$$

or, equivalently, if:

$$m_i + m_d = 2. \tag{20}$$

Since $m_i > 1$, (20) implies that $m_d < 1$.

The admissibility of the decreasing generator function $g_0(x) = 1/(a\,x + b)$ can also be verified by using directly the results of Theorem 2. Consider the function $g(x) = (g_0(x))^{\frac{1}{1-m}}$, with $g_0(x) = 1/(a\,x + b)$ and $m < 1$. For all $a > 0$ and $b > 0$, $g_0(x) = 1/(a\,x + b) > 0$, $\forall x \in (0, \infty)$. Since $g_0'(x) = -a/(a\,x + b)^2 < 0$, $\forall x \in (0, \infty)$, $g_0(x) = 1/(a\,x + b)$ is a monotonically decreasing function for all $a > 0$. Since $g_0''(x) = 2a^2/(a\,x + b)^3$,

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

For $m < 1$, $r_0(x) < 0$, $\forall x \in (0, \infty)$, and $g_0(x) = 1/(a\,x + b)$ is an admissible generator function in the wide sense.

For $g_0(x) = 1/(a\,x + b)$,

$$\lim_{x \to 0_+} g_0(x) = \frac{1}{b}, \tag{22}$$

which implies that $g_0(x) = 1/(a\,x + b)$ satisfies the fourth axiomatic requirement unless b approaches 0. In such a case, $\lim_{x\to 0_+} g_0(x) = 1/b =$

∞. For $g_0(x) = 1/(a\,x + b)$,

$$d_0(x) = \frac{a}{(a\,x+b)^4}\left(a\,\frac{m-3}{m-1}\,x - b\right). \tag{23}$$

If $m < 1$, the fifth axiomatic requirement is satisfied if $d_0(x) > 0,\ \forall x \in (0, \infty)$. Since $a > 0$, the condition $d_0(x) > 0$ is satisfied by $g_0(x) = 1/(a\,x + b)$ if:

$$x > \frac{m-1}{m-3}\,\frac{b}{a}. \tag{24}$$

Once again, the fifth axiomatic requirement is satisfied for $b = 0$, a value that violates the fourth axiomatic requirement.

4.2 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 of $x \in (0, \infty)$. 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,

$$r_0(x) = \frac{1}{m-1}\,(\beta\,\exp(\beta x))^2. \tag{25}$$

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 in the wide sense for all $\beta > 0$.

For $g_0(x) = \exp(\beta x), \beta > 0$,

$$\lim_{x \to 0_+} g_0(x) = 1 > 0, \tag{26}$$

which implies that $g_0(x) = \exp(\beta x)$ satisfies the fourth axiomatic requirement. For $g_0(x) = \exp(\beta x), \beta > 0$,

$$d_0(x) = (\beta\,\exp(\beta x))^2\,\left(\frac{1}{\beta} - \frac{2}{m-1}\,x\right). \tag{27}$$

For $m > 1$, the fifth axiomatic requirement is satisfied if $d_0(x) < 0$, $\forall x \in (0, \infty)$. The condition $d_0(x) < 0$ is satisfied by $g_0(x) = \exp(\beta x)$ if:

$$x > \frac{m-1}{2\beta} = \frac{\sigma^2}{2} > 0, \tag{28}$$

where $\sigma^2 = (m-1)/\beta$. Regardless of the value $\beta > 0$, $g_0(x) = \exp(\beta x)$ is not an admissible generator function in the strict sense.

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 $g_0''(x) = \beta^2 \exp(-\beta x)$,

$$r_0(x) = \frac{1}{m-1} \left(\beta \exp(-\beta x)\right)^2. \tag{29}$$

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 in the wide sense for all $\beta > 0$.

For $g_0(x) = \exp(-\beta x)$, $\beta > 0$,

$$\lim_{x \to 0_+} g_0(x) = 1 < \infty, \tag{30}$$

which implies that $g_0(x) = \exp(-\beta x)$ satisfies the fourth axiomatic requirement. For $g_0(x) = \exp(-\beta x)$, $\beta > 0$,

$$d_0(x) = \left(\beta \exp(-\beta x)\right)^2 \left(-\frac{1}{\beta} + \frac{2}{1-m}\, x\right). \tag{31}$$

For $m < 1$, the fifth axiomatic requirement is satisfied if $d_0(x) > 0$, $\forall x \in (0, \infty)$. The condition $d_0(x) > 0$ is satisfied by $g_0(x) = \exp(-\beta x)$ if:

$$x > \frac{1-m}{2\,\beta} = \frac{\sigma^2}{2}, \tag{32}$$

where $\sigma^2 = (1-m)/\beta$. Once again, $g_0(x) = \exp(-\beta x)$ is not an admissible generator function in the strict sense regardless of the value of $\beta > 0$.

It must be emphasized that both increasing and decreasing exponential generator functions essentially lead to the same radial basis function. If $m > 1$, the increasing exponential generator function $g_0(x) = \exp(\beta x)$, $\beta > 0$, corresponds to the Gaussian radial basis function $\phi(x) = g(x^2) = \exp(-x^2/\sigma^2)$, with $\sigma^2 = (m-1)/\beta$. If $m < 1$, the decreasing exponential generator function $g_0(x) = \exp(-\beta x)$, $\beta > 0$, also corresponds to the Gaussian radial basis function $\phi(x) = g(x^2) = \exp(-x^2/\sigma^2)$, with $\sigma^2 = (1-m)/\beta$. In fact, the function $g(x) = (g_0(x))^{\frac{1}{1-m}}$ corresponding to the increasing generator function $g_0(x) = \exp(\beta x), \beta > 0$, with $m = m_i > 1$ is identical with the function $g(x) = (g_0(x))^{\frac{1}{1-m}}$ corresponding to the decreasing function $g_0(x) = \exp(-\beta x), \beta > 0$, with $m = m_d < 1$ if:

$$m_i - 1 = 1 - m_d, \tag{33}$$

or, equivalently, if:

$$m_i + m_d = 2. \tag{34}$$

5 Selecting Generator Functions

All possible generator functions considered in the previous section satisfy the three basic axiomatic requirements but none of them satisfies all five axiomatic requirements. In particular, the fifth axiomatic requirement is satisfied only by generator functions of the form $g_0(x) = a\,x$, which violate the fourth axiomatic requirement. Therefore, it is clear that at least one of the five axiomatic requirements must be compromised in order to select a generator function. Since the response of the radial basis functions must be bounded in some function approximation applications, generator functions can be selected by compromising the fifth axiomatic requirement. Although this requirement is by itself very restrictive, its implications can be used to guide the search for generator functions appropriate for gradient descent learning [17].

5.1 The Blind Spot

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

$$\begin{aligned} \nabla_{\mathbf{x}_k} h_{j,k} &= g'(\|\mathbf{x}_k - \mathbf{v}_j\|^2)\, \nabla_{\mathbf{x}_k}(\|\mathbf{x}_k - \mathbf{v}_j\|^2) \\ &= 2\, g'(\|\mathbf{x}_k - \mathbf{v}_j\|^2)\, (\mathbf{x}_k - \mathbf{v}_j). \end{aligned} \tag{35}$$

The norm of the gradient $\nabla_{\mathbf{x}_k} h_{j,k}$ can be obtained from (35) as:

$$\begin{aligned} \|\nabla_{\mathbf{x}_k} h_{j,k}\|^2 &= 4\,\|\mathbf{x}_k - \mathbf{v}_j\|^2 \left(g'(\|\mathbf{x}_k - \mathbf{v}_j\|^2)\right)^2 \\ &= 4\,t(\|\mathbf{x}_k - \mathbf{v}_j\|^2), \end{aligned} \tag{36}$$

where $t(x) = x\,(g'(x))^2$. According to Theorem 1, the fifth axiomatic requirement is satisfied if and only if $d(x) = g'(x) + 2\,x\,g''(x) > 0,\ \forall x \in (0, \infty)$. Since $t(x) = x\,(g'(x))^2$,

$$\begin{aligned} t'(x) &= g'(x)\,(g'(x) + 2\,x\,g''(x)) \\ &= g'(x)\,d(x). \end{aligned} \tag{37}$$

Theorem 1 requires that $g(x)$ is a decreasing function of $x \in (0, \infty)$, which implies that $g'(x) < 0,\ \forall x \in (0, \infty)$. Thus, (37) indicates that the fifth axiomatic requirement is satisfied if $t'(x) < 0, \forall x \in (0, \infty)$. If this condition is not satisfied, then $\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$ is not a monotonically decreasing function of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ in the interval $(0, \infty)$, as required by the fifth axiomatic requirement. Given a function $g(\cdot)$ satisfying the three basic axiomatic requirements, the fifth axiomatic requirement can be relaxed by requiring that $\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$ is a monotonically decreasing function of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ in the interval (B, ∞) for some $B > 0$. According to (36), this is guaranteed if the function $t(x) = x\,(g'(x))^2$ has a maximum at $x = B$ or, equivalently, if there exists a $B > 0$ such that $t'(B) = 0$ and $t''(B) < 0$. If $B \in (0, \infty)$ is a solution of $t'(x) = 0$ and $t''(B) < 0$, then $t'(x) > 0, \forall x \in (0, B)$, and $t'(x) < 0, \forall x \in (B, \infty)$. Thus, $\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$ is an increasing function of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ for $\|\mathbf{x}_k - \mathbf{v}_j\|^2 \in (0, B)$ and a decreasing function of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ for $\|\mathbf{x}_k - \mathbf{v}_j\|^2 \in (B, \infty)$. For all input vectors $\mathbf{x}_k$ that satisfy $\|\mathbf{x}_k - \mathbf{v}_j\|^2 < B$, the norm of the gradient $\nabla_{\mathbf{x}_k} h_{j,k}$ corresponding to the jth radial basis function decreases as $\mathbf{x}_k$ approaches its center that is located at the prototype $\mathbf{v}_j$. This is exactly the opposite behavior of what would intuitively be expected, given the interpretation of radial basis functions as receptive fields. As far as gradient descent learning is concerned, the hypersphere $\mathcal{R}_B = \{\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^{n_i} : \|\mathbf{x} - \mathbf{v}\|^2 \in (0, B)\}$ is a "blind spot" for the radial basis function located at the prototype $\mathbf{v}$. The blind spot provides a measure of the sensitivity of radial basis functions to input vectors close to their centers.

The blind spot $\mathcal{R}_{B_{\text{lin}}}$ corresponding to the linear generator function

$g_0(x) = a\,x + b$ is determined by:

$$B_{\text{lin}} = \frac{m-1}{m+1}\,\frac{b}{a}. \tag{38}$$

The effect of the parameter m to the size of the blind spot is revealed by the behavior of the ratio $(m-1)/(m+1)$ viewed as a function of m. Since $(m-1)/(m+1)$ increases as the value of m increases, increasing the value of m expands the blind spot. For a fixed value of $m > 1$, $B_{\text{lin}} = 0$ only if $b = 0$. For $b \neq 0$, B_{lin} decreases and approaches 0 as a increases and approaches infinity. If $a = 1$ and $b = \gamma^2$, B_{lin} approaches 0 as γ approaches 0. If $a = \delta$ and $b = 1$, B_{lin} decreases and approaches 0 as δ increases and approaches infinity.

The blind spot $\mathcal{R}_{B_{\exp}}$ corresponding to the exponential generator function $g_0(x) = \exp(\beta x)$ is determined by:

$$B_{\exp} = \frac{m-1}{2\beta}. \tag{39}$$

For a fixed value of β, the blind spot depends exclusively on the parameter m. Once again, the blind spot corresponding to the exponential generator function expands as the value of m increases. For a fixed value of $m > 1$, $B_{\exp}$ decreases and approaches 0 as β increases and approaches infinity. For $g_0(x) = \exp(\beta x)$, $g(x) = (g_0(x))^{\frac{1}{1-m}} = \exp(-x/\sigma^2)$ with $\sigma^2 = (m-1)/\beta$. As a result, the blind spot corresponding to the exponential generator function approaches 0 only if the width of the Gaussian radial basis function $\phi(x) = g(x^2) = \exp(-x^2/\sigma^2)$ approaches 0. Such a range of values of σ would make it difficult for Gaussian radial basis functions to behave as receptive fields that can cover the entire input space.

It is clear from (38) and (39) that the blind spot corresponding to the exponential generator function is much more sensitive to changes of m compared with that corresponding to the linear generator function. This can be quantified by computing for both generator functions the relative sensitivity of $B = B(m)$ in terms of m, defined as:

$$S_B^m = \frac{m}{B}\,\frac{\partial B}{\partial m}. \tag{40}$$

For the linear generator function $g_0(x) = a\,x + b$, $\partial B_{\mathrm{lin}}/\partial m = (2/(m+1)^2)\,(b/a)$ and

$$S^m_{B_{\mathrm{lin}}} = \frac{2\,m}{m^2-1}. \tag{41}$$

For the exponential generator function $g_0(x) = \exp(\beta x)$, $\partial B_{\mathrm{exp}}/\partial m = 1/(2\,\beta)$ and

$$S^m_{B_{\mathrm{exp}}} = \frac{m}{m-1}. \tag{42}$$

Combining (41) and (42) gives:

$$S^m_{B_{\mathrm{exp}}} = \frac{m+1}{2}\,S^m_{B_{\mathrm{lin}}}. \tag{43}$$

Since $m > 1$, $S^m_{B_{\mathrm{exp}}} > S^m_{B_{\mathrm{lin}}}$. As an example, for $m = 3$ the sensitivity with respect to m of the blind spot corresponding to the exponential generator function is twice that corresponding to the linear generator function.

5.2 Criteria for Selecting Generator Functions

The response of the radial basis function located at the prototype $\mathbf{v}_j$ to training vectors depends on their Euclidean distance from $\mathbf{v}_j$ and the shape of the generator function used. If the generator function does not satisfy the fifth axiomatic requirement, the response of the radial basis function located at each prototype exhibits the desired behavior only if the training vectors are located outside its blind spot. This implies that the training of a RBF model by a learning procedure based on gradient descent depends mainly on the sensitivity of the radial basis functions to training vectors outside their blind spots. This indicates that the criteria used for selecting generator functions should involve both the shapes of the radial basis functions relative to their blind spots and the sensitivity of the radial basis functions to input vectors outside their blind spots. The sensitivity of the response $h_{j,k}$ of the jth radial basis function to any input $\mathbf{x}_k$ can be measured by the norm of the gradient $\nabla_{\mathbf{x}_k} h_{j,k}$. Thus, the shape and sensitivity of the radial basis function located at the prototype $\mathbf{v}_j$ are mainly affected by:

1. the value $h_j^* = g(B)$ of the response $h_{j,k} = g(\|\mathbf{x}_k - \mathbf{v}_j\|^2)$ of the jth radial basis function at $\|\mathbf{x}_k - \mathbf{v}_j\|^2 = B$ and the rate at which $h_{j,k} = g(\|\mathbf{x}_k - \mathbf{v}_j\|^2)$ decreases as $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ increases above B and approaches infinity, and

2. the maximum value attained by the norm of the gradient $\nabla_{\mathbf{x}_k} h_{j,k}$ at $\|\mathbf{x}_k - \mathbf{v}_j\|^2 = B$ and the rate at which $\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$ decreases as $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ increases above B and approaches infinity.

The criteria that may be used for selecting radial basis functions can be established by considering the following extreme situation. Suppose the response $h_{j,k} = g(\|\mathbf{x}_k - \mathbf{v}_j\|^2)$ diminishes very quickly and the receptive field located at the prototype $\mathbf{v}_j$ does not extend far beyond the blind spot. This can have a negative impact on the function approximation ability of the corresponding RBF model since the region outside the blind spot contains the input vectors that affect the implementation of the input-output mapping as indicated by the sensitivity measure $\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$. Thus, a generator function must be selected in such a way that:

1. the response $h_{j,k}$ and the sensitivity measure $\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$ take substantial values outside the blind spot before they approach 0, and

2. the response $h_{j,k}$ is sizable outside the blind sport even after the values of $\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$ become negligible.

The rate at which the response $h_{j,k} = g(\|\mathbf{x}_k - \mathbf{v}_j\|^2)$ decreases relates to the "tails" of the functions $g(\cdot)$ that correspond to different generator functions. The use of a short-tailed function $g(\cdot)$ shrinks the receptive fields of the RBF model while the use of a long-tailed function $g(\cdot)$ increases the overlapping between the receptive fields located at different prototypes. If $g(x) = (g_0(x))^{\frac{1}{1-m}}$ and $m > 1$, the tail of $g(x)$ is determined by how fast the corresponding generator function $g_0(x)$ changes as a function of x. As x increases, the exponential generator function $g_0(x) = \exp(\beta x)$ increases faster than the linear generator function $g_0(x) = a\,x + b$. As a result, the response $g(x) = (g_0(x))^{\frac{1}{1-m}}$ diminishes quickly if $g_0(\cdot)$ is exponential and slower if $g_0(\cdot)$ is linear.

The behavior of the sensitivity measure $\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$ also depends on the properties of the function $g(\cdot)$. For $h_{j,k} = g(\|\mathbf{x}_k - \mathbf{v}_j\|^2)$, $\nabla_{\mathbf{x}_k} h_{j,k}$ can be

obtained from (35) as:

$$\nabla_{\mathbf{x}_k} h_{j,k} = -\alpha_{j,k} \, (\mathbf{x}_k - \mathbf{v}_j), \tag{44}$$

where

$$\alpha_{j,k} = -2 \, g'(\|\mathbf{x}_k - \mathbf{v}_j\|^2). \tag{45}$$

From (44),

$$\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2 = \|\mathbf{x}_k - \mathbf{v}_j\|^2 \, \alpha_{j,k}^2. \tag{46}$$

The selection of a specific function $g(\cdot)$ influences the sensitivity measure $\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$ through $\alpha_{j,k} = -2 \, g'(\|\mathbf{x}_k - \mathbf{v}_j\|^2)$. If $g(x) = (g_0(x))^{\frac{1}{1-m}}$, then:

$$\begin{aligned} g'(x) &= \frac{1}{1-m} \, (g_0(x))^{\frac{m}{1-m}} \, g_0'(x) \\ &= \frac{1}{1-m} \, (g(x))^m \, g_0'(x). \end{aligned} \tag{47}$$

Since $h_{j,k} = g\left(\|\mathbf{x}_k - \mathbf{v}_j\|^2\right)$, $\alpha_{j,k}$ is given by:

$$\alpha_{j,k} = \frac{2}{m-1} \, (h_{j,k})^m \, g_0'(\|\mathbf{x}_k - \mathbf{v}_j\|^2). \tag{48}$$

5.3 Evaluation of Linear and Exponential Generator Functions

The criteria presented above are used here for evaluating linear and exponential generator functions.

5.3.1 Linear Generator Functions

If $g(x) = (g_0(x))^{\frac{1}{1-m}}$, with $g_0(x) = a\,x + b$ and $m > 1$, the response $h_{j,k} = g(\|\mathbf{x}_k - \mathbf{v}_j\|^2)$ of the jth radial basis function to $\mathbf{x}_k$ is:

$$h_{j,k} = \left(\frac{1}{a \, \|\mathbf{x}_k - \mathbf{v}_j\|^2 + b} \right)^{\frac{1}{m-1}}. \tag{49}$$

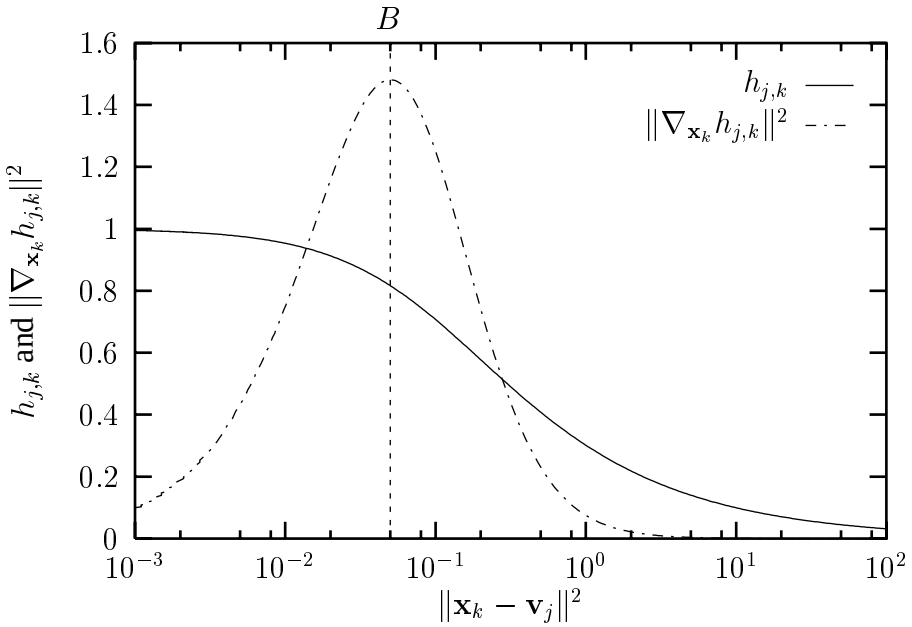


Figure 2. The response $h_{j,k} = g(\|\mathbf{x}_k - \mathbf{v}_j\|^2)$ of the jth radial basis function and the norm of the gradient $\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$ plotted as functions of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ for $g(x) = (g_0(x))^{\frac{1}{1-m}}$, with $g_0(x) = 1 + \delta\, x$, $m = 3$, and $\delta = 10$.

For this generator function, $g_0'(x) = a$ and (48) gives:

$$\begin{aligned} \alpha_{j,k} &= \frac{2a}{m-1} (h_{j,k})^m \\ &= \frac{2a}{m-1} \left(\frac{1}{a \|\mathbf{x}_k - \mathbf{v}_j\|^2 + b} \right)^{\frac{m}{m-1}} . \end{aligned} \tag{50}$$

Thus, $\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$ can be obtained from (46) as:

$$\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2 = \left(\frac{2\,a}{m-1} \right)^2 \|\mathbf{x}_k - \mathbf{v}_j\|^2 \left(\frac{1}{a \|\mathbf{x}_k - \mathbf{v}_j\|^2 + b} \right)^{\frac{2m}{m-1}} . \tag{51}$$

Figures 2 and 3 show the response $h_{j,k} = g(\|\mathbf{x}_k - \mathbf{v}_j\|^2)$ of the jth radial basis function to the input vector $\mathbf{x}_k$ and the sensitivity measure $\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$ plotted as functions of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ for $g(x) = (g_0(x))^{\frac{1}{1-m}}$, with $g_0(x) = 1 + \delta\, x$, $m = 3$, for $\delta = 10$ and $\delta = 100$, respectively. In accordance with the analysis, $\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$ increases monotonically as

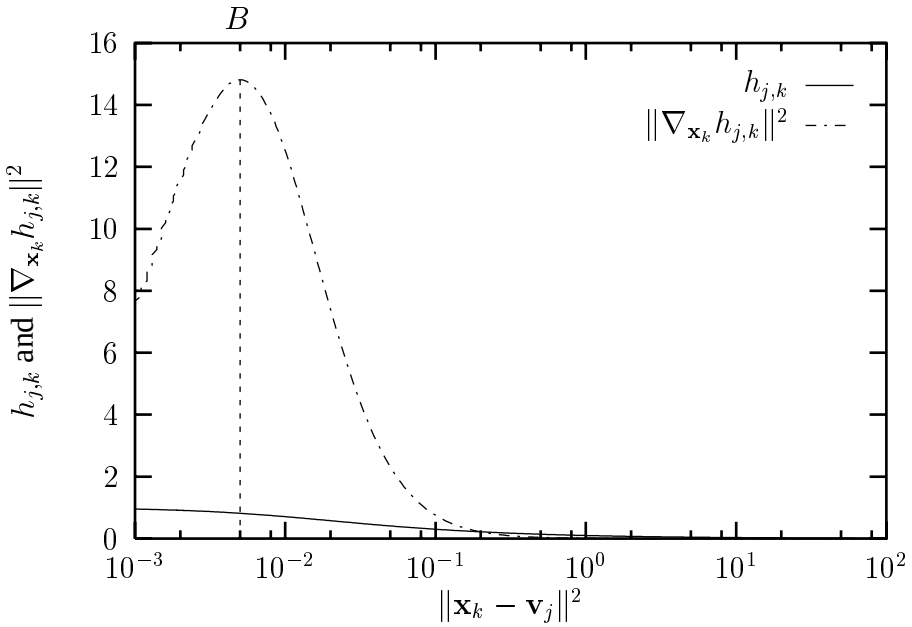


Figure 3. The response $h_{j,k} = g(\|\mathbf{x}_k - \mathbf{v}_j\|^2)$ of the jth radial basis function and the norm of the gradient $\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$ plotted as functions of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ for $g(x) = (g_0(x))^{\frac{1}{1-m}}$, with $g_0(x) = 1 + \delta\, x$, $m = 3$, and $\delta = 100$.

$\|\mathbf{x}_k - \mathbf{v}_j\|^2$ increases from 0 to $B = 1/(2\,\delta)$ and decreases monotonically as $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ increases above B and approaches infinity. Figures 2 and 3 indicate that, regardless of the value of δ, the response $h_{j,k}$ of the radial basis function located at the prototype $\mathbf{v}_j$ is sizable outside the blind spot even after the values of $\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$ become negligible. Thus, the radial basis function located at the prototype $\mathbf{v}_j$ is activated by all input vectors that correspond to substantial values of $\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$.

5.3.2 Exponential Generator Functions

If $g(x) = (g_0(x))^{\frac{1}{1-m}}$, with $g_0(x) = \exp(\beta x)$ and $m > 1$, the response $h_{j,k} = g(\|\mathbf{x}_k - \mathbf{v}_j\|^2)$ of the jth radial basis function to $\mathbf{x}_k$ is:

$$h_{j,k} = \exp\left(-\frac{\|\mathbf{x}_k - \mathbf{v}_j\|^2}{\sigma^2}\right), \tag{52}$$

where $\sigma^2 = (m-1)/\beta$. For this generator function, $g_0'(x) = \beta\, \exp(\beta x) = \beta\, g_0(x)$. In this case, $g_0'\left(\|\mathbf{x}_k - \mathbf{v}_j\|^2\right) = \beta\, (h_{j,k})^{1-m}$ and

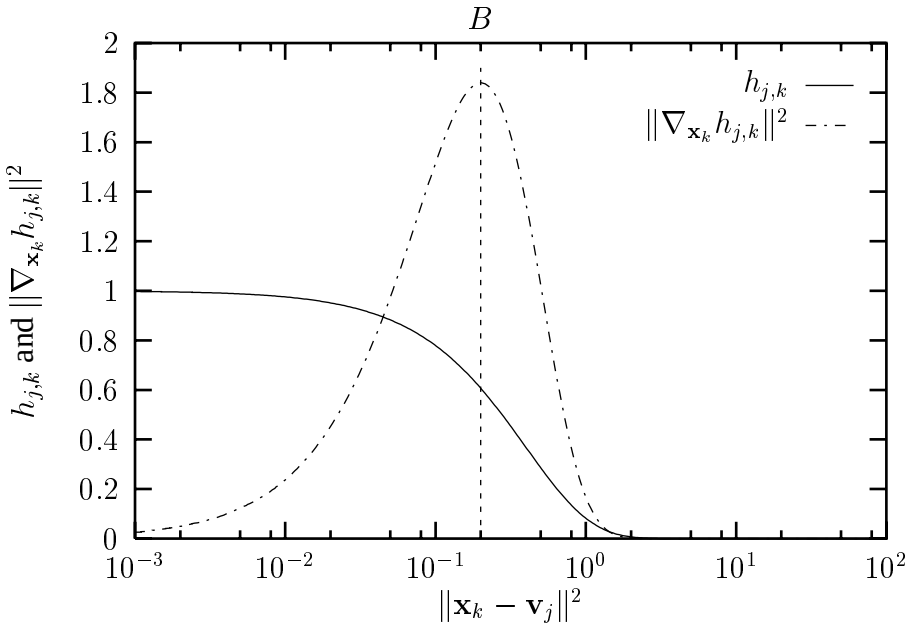


Figure 4. The response $h_{j,k} = g(\|\mathbf{x}_k - \mathbf{v}_j\|^2)$ of the jth radial basis function and the norm of the gradient $\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$ plotted as functions of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ for $g(x) = (g_0(x))^{\frac{1}{1-m}}$, with $g_0(x) = \exp(\beta\, x)$, $m = 3$, and $\beta = 5$.

(48) gives:

$$\begin{aligned} \alpha_{j,k} &= \frac{2\beta}{m-1}\, h_{j,k} \\ &= \frac{2}{\sigma^2} \exp\left(-\frac{\|\mathbf{x}_k - \mathbf{v}_j\|^2}{\sigma^2}\right). \end{aligned} \tag{53}$$

Thus, $\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$ can be obtained from (46) as:

$$\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2 = \left(\frac{2}{\sigma^2}\right)^2 \|\mathbf{x}_k - \mathbf{v}_j\|^2 \exp\left(-2\,\frac{\|\mathbf{x}_k - \mathbf{v}_j\|^2}{\sigma^2}\right). \tag{54}$$

Figures 4 and 5 show the response $h_{j,k} = g(\|\mathbf{x}_k - \mathbf{v}_j\|^2)$ of the jth radial basis function to the input vector $\mathbf{x}_k$ and the sensitivity measure $\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$ plotted as functions of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ for $g(x) = (g_0(x))^{\frac{1}{1-m}}$, with $g_0(x) = \exp(\beta x)$, $m = 3$, for $\beta = 5$ and $\beta = 10$, respectively. Once again, $\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$ increases monotonically as $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ increases from 0 to $B = 1/\beta$ and decreases monotonically as $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ increases

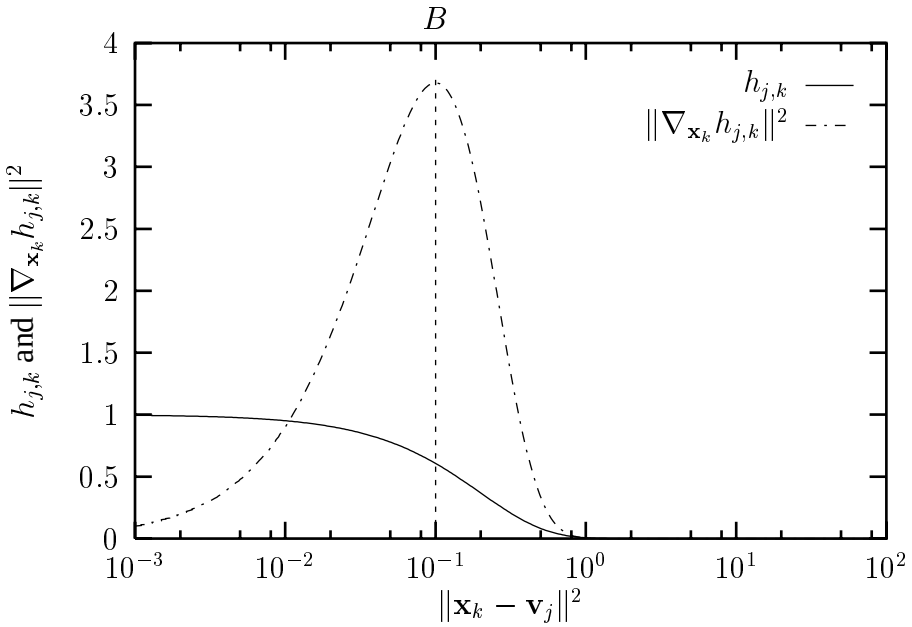


Figure 5. The response $h_{j,k} = g(\|\mathbf{x}_k - \mathbf{v}_j\|^2)$ of the jth radial basis function and the norm of the gradient $\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$ plotted as functions of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ for $g(x) = (g_0(x))^{\frac{1}{1-m}}$, with $g_0(x) = \exp(\beta\, x)$, $m = 3$, and $\beta = 10$.

above B and approaches infinity. Nevertheless, there are some significant differences between the response $h_{j,k}$ and the sensitivity measure $\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$ corresponding to linear and exponential generator functions. If $g_0(x) = \exp(\beta x)$, then the response $h_{j,k}$ is substantial for the input vectors inside the blind spot but diminishes very quickly for values of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ above B. In fact, the values of $h_{j,k}$ become negligible even before $\|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$ approaches asymptotically zero values. This is in direct contrast with the behavior of the same quantities corresponding to linear generator functions, which are shown in Figures 2 and 3.

6 Learning Algorithms Based on Gradient Descent

Reformulated RBF neural networks can be trained to map $\mathbf{x}_k \in \mathbb{R}^{n_i}$ into $\mathbf{y}_k = [y_{1,k}\, y_{2,k}\, \ldots\, y_{n_o,k}]^T \in \mathbb{R}^{n_o}$, where the vector pairs $(\mathbf{x}_k, \mathbf{y}_k)$, $1 \leq k \leq M$, form the training set. If $\mathbf{x}_k \in \mathbb{R}^{n_i}$ is the input to a reformulated RBF network, its response is $\hat{\mathbf{y}}_k = [\hat{y}_{1,k}\, \hat{y}_{2,k}\, \ldots\, \hat{y}_{n_o,k}]^T$, where $\hat{y}_{i,k}$ is the

actual response of the ith output unit to $\mathbf{x}_k$ given by:

$$\begin{aligned} \hat{y}_{i,k} &= f(\bar{y}_{i,k}) \\ &= f\left(\mathbf{w}_i^T\,\mathbf{h}_k\right) \\ &= f\left(\sum_{j=0}^{c} w_{ij}\,h_{j,k}\right), \end{aligned} \tag{55}$$

with $h_{0,k} = 1$, $1 \leq k \leq M$, $h_{j,k} = g\left(\|\mathbf{x}_k - \mathbf{v}_j\|^2\right)$, $1 \leq j \leq c$, $\mathbf{h}_k = [h_{0,k}\, h_{1,k}\, \ldots\, h_{c,k}]^T$, and $\mathbf{w}_i = [w_{i,0}\, w_{i,1}\, \ldots\, w_{i,c}]^T$. Training is typically based on the minimization of the error between the actual outputs of the network $\hat{\mathbf{y}}_k$, $1 \leq k \leq M$, and the desired responses $\mathbf{y}_k$, $1 \leq k \leq M$.

6.1 Batch Learning Algorithms

A reformulated RBF neural network can be trained by minimizing the error:

$$E = \frac{1}{2}\sum_{k=1}^{M}\sum_{i=1}^{n_o}(y_{i,k} - \hat{y}_{i,k})^2. \tag{56}$$

Minimization of (56) using gradient descent implies that all training examples are presented to the RBF network simultaneously. Such training strategy leads to *batch* learning algorithms. The update equation for the weight vectors of the upper associative network can be obtained using gradient descent as [21]:

$$\begin{aligned} \Delta\mathbf{w}_p &= -\eta\,\nabla_{\mathbf{w}_p} E \\ &= \eta\,\sum_{k=1}^{M}\varepsilon_{p,k}^{o}\,\mathbf{h}_k, \end{aligned} \tag{57}$$

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

$$\varepsilon_{p,k}^{o} = f'(\bar{y}_{p,k})\,(y_{p,k} - \hat{y}_{p,k}). \tag{58}$$

Similarly, the update equation for the prototypes can be obtained using gradient descent as [21]:

$$\begin{aligned} \Delta\mathbf{v}_q &= -\eta\,\nabla_{\mathbf{v}_q} E \\ &= \eta\,\sum_{k=1}^{M}\varepsilon_{q,k}^{h}\,(\mathbf{x}_k - \mathbf{v}_q), \end{aligned} \tag{59}$$

where η is the learning rate and $\varepsilon^h_{q,k}$ is the *hidden error*, defined as:

$$\varepsilon^h_{q,k} = \alpha_{q,k} \sum_{i=1}^{n_o} \varepsilon^o_{i,k}\, w_{iq}, \tag{60}$$

with $\alpha_{q,k} = -2\, g'\left(\|\mathbf{x}_k - \mathbf{v}_q\|^2\right)$. The selection of a specific function $g(\cdot)$ influences the update of the prototypes through $\alpha_{q,k} = -2\, g'(\|\mathbf{x}_k - \mathbf{v}_q\|^2)$, which is involved in the calculation of the corresponding hidden error $\varepsilon^h_{q,k}$. Since $h_{q,k} = g\left(\|\mathbf{x}_k - \mathbf{v}_q\|^2\right)$ and $g(x) = (g_0(x))^{\frac{1}{1-m}}$, $\alpha_{q,k}$ is given by (48) and the hidden error (60) becomes:

$$\varepsilon^h_{q,k} = \frac{2}{m-1}\,(h_{q,k})^m\, g_0'(\|\mathbf{x}_k - \mathbf{v}_q\|^2) \sum_{i=1}^{n_o} \varepsilon^o_{i,k}\, w_{iq}. \tag{61}$$

A RBF neural network can be trained according to the algorithm presented 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:

$$\mathbf{w}_p + \Delta\mathbf{w}_p = \mathbf{w}_p + \eta \sum_{k=1}^{M} \varepsilon^o_{p,k}\, \mathbf{h}_k. \tag{62}$$

Given the learning rate η and the responses $\mathbf{h}_k$ of the radial basis functions, these weight vectors are updated according to the output errors $\varepsilon^o_{p,k}, 1 \le p \le n_o$. Following the update of these weight vectors, the current estimate of each prototype $\mathbf{v}_q$, $1 \le q \le c$, is replaced by:

$$\mathbf{v}_q + \Delta\mathbf{v}_q = \mathbf{v}_q + \eta \sum_{k=1}^{M} \varepsilon^h_{q,k}\,(\mathbf{x}_k - \mathbf{v}_q). \tag{63}$$

For a given value of the learning rate η, the update of $\mathbf{v}_q$ depends on the hidden errors $\varepsilon^h_{q,k}, 1 \le k \le M$. The hidden error $\varepsilon^h_{q,k}$ is influenced by the output errors $\varepsilon^o_{i,k}, 1 \le i \le n_o$, and the weights $w_{iq}, 1 \le i \le n_o$, through the term $\sum_{i=1}^{n_o} \varepsilon^o_{i,k}\, w_{iq}$. Thus, the RBF neural network is trained according to this scheme by propagating back the output error.

This algorithm can be summarized as follows:

1. Select η and ϵ; initialize $\{w_{ij}\}$ with zero values; initialize the prototypes $\mathbf{v}_j, 1 \le j \le c$; set $h_{0,k} = 1, \forall k$.

2. Compute the initial response:

   - $h_{j,k} = \left(g_0\left(\|\mathbf{x}_k - \mathbf{v}_j\|^2\right)\right)^{\frac{1}{1-m}}, \forall j,k.$
   - $\mathbf{h}_k = [h_{0,k}\, h_{1,k}\, \ldots\, h_{c,k}]^T, \forall k.$
   - $\hat{y}_{i,k} = f(\mathbf{w}_i^T\, \mathbf{h}_k), \forall i,k.$

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:

   - $\varepsilon_{i,k}^o = f'(\bar{y}_{i,k})(y_{i,k} - \hat{y}_{i,k}), \forall i,k.$
   - $\mathbf{w}_i \leftarrow \mathbf{w}_i + \eta\sum_{k=1}^{M}\varepsilon_{i,k}^o\,\mathbf{h}_k, \forall i.$
   - $\varepsilon_{j,k}^h = \frac{2}{m-1}\, g_0'\left(\|\mathbf{x}_k - \mathbf{v}_j\|^2\right)(h_{j,k})^m \sum_{i=1}^{n_o}\varepsilon_{i,k}^o\, w_{ij}, \forall j,k.$
   - $\mathbf{v}_j \leftarrow \mathbf{v}_j + \eta\sum_{k=1}^{M}\varepsilon_{j,k}^h\,(\mathbf{x}_k - \mathbf{v}_j), \forall j.$

6. Compute the current response:

   - $h_{j,k} = \left(g_0\left(\|\mathbf{x}_k - \mathbf{v}_j\|^2\right)\right)^{\frac{1}{1-m}}, \forall j,k.$
   - $\mathbf{h}_k = [h_{0,k}\, h_{1,k}\, \ldots\, h_{c,k}]^T, \forall k.$
   - $\hat{y}_{i,k} = f(\mathbf{w}_i^T\, \mathbf{h}_k), \forall i,k.$

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 4.

6.2 Sequential Learning Algorithms

Reformulated RBF neural networks can also be trained "on-line" by *sequential* learning algorithms. Such algorithms can be developed by using gradient descent to minimize the errors:

$$E_k = \frac{1}{2}\sum_{i=1}^{n_o}(y_{i,k} - \hat{y}_{i,k})^2, \tag{64}$$

for $k = 1, 2, \ldots, M$. The update equation for the weight vectors of the upper associative network can be obtained using gradient descent as [21]:

$$\begin{aligned} \Delta \mathbf{w}_{p,k} &= \mathbf{w}_{p,k} - \mathbf{w}_{p,k-1} \\ &= -\eta \nabla_{\mathbf{w}_p} E_k \\ &= \eta \, \varepsilon^o_{p,k} \, \mathbf{h}_k, \end{aligned} \tag{65}$$

where $\mathbf{w}_{p,k-1}$ and $\mathbf{w}_{p,k}$ are the estimates of the weight vector $\mathbf{w}_p$ before and after the presentation of the training example $(\mathbf{x}_k, \mathbf{y}_k)$, η is the learning rate, and $\varepsilon^o_{p,k}$ is the output error defined in (58). Similarly, the update equation for the prototypes can be obtained using gradient descent as [21]:

$$\begin{aligned} \Delta \mathbf{v}_{q,k} &= \mathbf{v}_{q,k} - \mathbf{v}_{q,k-1} \\ &= -\eta \nabla_{\mathbf{v}_q} E_k \\ &= \eta \, \varepsilon^h_{q,k} \, (\mathbf{x}_k - \mathbf{v}_q), \end{aligned} \tag{66}$$

where $\mathbf{v}_{q,k-1}$ and $\mathbf{v}_{q,k}$ are the estimates of the prototype $\mathbf{v}_q$ before and after the presentation of the training example $(\mathbf{x}_k, \mathbf{y}_k)$, η is the learning rate, and $\varepsilon^h_{q,k}$ is the hidden error defined in (61).

When an adaptation cycle begins, the current estimates of the weight vectors $\mathbf{w}_p$ and the prototypes $\mathbf{v}_q$ are stored in $\mathbf{w}_{p,0}$ and $\mathbf{v}_{q,0}$, respectively. After an example $(\mathbf{x}_k, \mathbf{y}_k)$, $1 \leq k \leq M$, is presented to the network, each weight vector $\mathbf{w}_p$, $1 \leq p \leq n_o$, is updated as:

$$\mathbf{w}_{p,k} = \mathbf{w}_{p,k-1} + \Delta \mathbf{w}_{p,k} = \mathbf{w}_{p,k-1} + \eta \, \varepsilon^o_{p,k} \, \mathbf{h}_k. \tag{67}$$

Following the update of all the weight vectors $\mathbf{w}_p$, $1 \leq p \leq n_o$, each prototype $\mathbf{v}_q, 1 \leq q \leq c$, is updated as:

$$\mathbf{v}_{q,k} = \mathbf{v}_{q,k-1} + \Delta \mathbf{v}_{q,k} = \mathbf{v}_{q,k-1} + \eta \, \varepsilon^h_{q,k} \, (\mathbf{x}_k - \mathbf{v}_{q,k-1}). \tag{68}$$

An adaptation cycle is completed in this case after the sequential presentation to the network of all the examples included in the training set. Once again, the RBF neural network is trained according to this scheme by propagating back the output error.

This algorithm can be summarized as follows:

1. Select η and ϵ; initialize $\{w_{ij}\}$ with zero values; initialize the prototypes $\mathbf{v}_j, 1 \leq j \leq c$; set $h_{0,k} = 1, \forall k$.

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

6.3 Initialization of Supervised Learning

The training of reformulated RBF neural networks using gradient descent can be initialized by randomly generating the set of prototypes that determine the locations of the radial basis function centers in the input space. Such an approach relies on the supervised learning algorithm to determine appropriate locations for the radial basis function centers by updating the prototypes during learning. Nevertheless, the training of reformulated RBF neural networks by gradient descent algorithms can be facilitated by initializing the supervised learning process using a set of prototypes specifically determined to represent the input vectors included

in the training set. This can be accomplished by computing the initial set of prototypes using unsupervised clustering or learning vector quantization (LVQ) algorithms.

According to the learning scheme often used for training conventional RBF neural networks [34], the locations of the radial basis function centers are determined from the input vectors included in the training set using the c-means (or k-means) algorithm. The c-means algorithm begins from an initial set of c prototypes, which implies the partition of the input vectors into c clusters. Each cluster is represented by a prototype, which is evaluated at subsequent iterations as the centroid of the input vectors belonging to that cluster. Each input vector is assigned to the cluster whose prototype is its closest neighbor. In mathematical terms, the indicator function $u_{ij} = u_j(\mathbf{x}_i)$ that assigns the input vector $\mathbf{x}_i$ to the jth cluster is computed as [9]:

$$u_{ij} = \begin{cases} 1, & \text{if } \|\mathbf{x}_i - \mathbf{v}_j\|^2 < \|\mathbf{x}_i - \mathbf{v}_\ell\|^2, \, \forall \ell \neq j, \\ 0, & \text{otherwise.} \end{cases} \tag{69}$$

For a given set of indicator functions, the new set of prototypes is calculated as [9]:

$$\mathbf{v}_j = \frac{\sum_{i=1}^{M} u_{ij} \, \mathbf{x}_i}{\sum_{i=1}^{M} u_{ij}}, \; 1 \leq j \leq c. \tag{70}$$

The c-means algorithm partitions the input vectors into clusters represented by a set of prototypes based on *hard* or *crisp* decisions. In other words, each input vector is assigned to the cluster represented by its closest prototype. Since this strategy fails to quantify the uncertainty typically associated with partitioning a set of input vectors, the performance of the c-means algorithm depends rather strongly on its initialization [8], [26]. When this algorithm is initialized randomly, it often converges to shallow local minima and produces empty clusters.

Most of the disadvantages of the c-means algorithm can be overcome by employing a prototype splitting procedure to produce the initial set of prototypes. Such a procedure is employed by a variation of the c-means algorithm often referred to in the literature as the LBG (Linde-Buzo-Gray) algorithm [31], which is often used for codebook design in image and video compression approaches based on vector quantization.

The LBG algorithm employs an initialization scheme to compensate for the dependence of the c-means algorithm on its initialization [8]. More specifically, this algorithm generates the desired number of clusters by successively splitting the prototypes and subsequently employing the c-means algorithm. The algorithm begins with a single prototype that is calculated as the centroid of the available input vectors. This prototype is split into two vectors, which provide the initial estimate for the c-means algorithm that is used with $c = 2$. Each of the resulting vectors is then split into two vectors and the above procedure is repeated until the desired number of prototypes is obtained. Splitting is performed by adding the perturbation vectors $\pm\mathbf{e}_i$ to each vector $\mathbf{v}_i$ producing two vectors: $\mathbf{v}_i + \mathbf{e}_i$ and $\mathbf{v}_i - \mathbf{e}_i$. The perturbation vector $\mathbf{e}_i$ can be calculated from the variance between the input vectors and the prototypes [8].

7 Generator Functions and Gradient Descent Learning

The effect of the generator function on gradient descent learning algorithms developed for reformulated RBF neural networks essentially relates to the criteria established in Section 5 for selecting generator functions. These criteria were established on the basis of the response $h_{j,k}$ of the jth radial basis function to an input vector $\mathbf{x}_k$ and the norm of the gradient $\nabla_{\mathbf{x}_k} h_{j,k}$, that can be used to measure the sensitivity of the radial basis function response $h_{j,k}$ to an input vector $\mathbf{x}_k$. Since $\nabla_{\mathbf{x}_k} h_{j,k} = -\nabla_{\mathbf{v}_j} h_{j,k}$, (46) gives

$$\|\nabla_{\mathbf{v}_j} h_{j,k}\|^2 = \|\mathbf{x}_k - \mathbf{v}_j\|^2 \alpha_{j,k}^2. \tag{71}$$

According to (71), the quantity $\|\mathbf{x}_k - \mathbf{v}_j\|^2 \alpha_{j,k}^2$ can also be used to measure the sensitivity of the response of the jth radial basis function to changes in the prototype $\mathbf{v}_j$ that represents its location in the input space.

The gradient descent learning algorithms presented in Section 6 attempt to train a 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 radial basis functions 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 [28], such updates alone are unlikely to implement non-trivial input-output mappings. Thus, the ability of the network to implement a desired input-output mapping depends to a large extent on the sensitivity of the responses of the radial basis functions to incremental changes of their corresponding prototypes. This discussion indicates that the sensitivity measure $\|\nabla_{\mathbf{v}_j} h_{j,k}\|^2$ is relevant to gradient descent learning algorithms developed for reformulated RBF neural networks. Moreover, the form of this sensitivity measure in (71) underlines the significant role of the generator function, whose selection affects $\|\nabla_{\mathbf{v}_j} h_{j,k}\|^2$ as indicated by the definition of $\alpha_{j,k}$ in (48). The effect of the generator function on gradient descent learning is revealed by comparing the response $h_{j,k}$ and the sensitivity measure $\|\nabla_{\mathbf{v}_j} h_{j,k}\|^2 = \|\nabla_{\mathbf{x}_k} h_{j,k}\|^2$ corresponding to the linear and exponential generator functions.

According to Figures 2 and 3, the response $h_{j,k}$ of the jth radial basis function to the input $\mathbf{x}_k$ diminishes very slowly outside the blind spot, i.e., as $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ increases above B. This implies that the training vector $\mathbf{x}_k$ has a non-negligible effect on the response $h_{j,k}$ of the radial basis function located at this prototype. The behavior of the sensitivity measure $\|\nabla_{\mathbf{v}_j} h_{j,k}\|^2$ outside the blind spot indicates that the update of the prototype $\mathbf{v}_j$ produces significant variations in the input of the upper associative network, which is trained to implement the desired input-output mapping by updating the output weights. Figures 2 and 3 also reveal the trade-off involved in the selection of the free parameter δ in practice. As the value of δ increases, $\|\nabla_{\mathbf{v}_j} h_{j,k}\|^2$ attains significantly higher values. This implies that the jth radial basis function is more sensitive to updates of the prototype $\mathbf{v}_j$ due to input vectors outside its blind spot. The blind spot shrinks as the value of δ increases but $\|\nabla_{\mathbf{v}_j} h_{j,k}\|^2$ approaches 0 quickly outside the blind spot, i.e., as the value of $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ increases above B. This implies that the receptive fields located at the prototypes shrink, which can have a negative impact on the gradient descent learning. Increasing the value of δ can also affect the number of radial basis functions required for the implementation of the desired input-output mapping. This is due to the fact that more radial basis functions are required to cover the input space. The receptive fields located at the prototypes can be expanded by decreasing the value of δ. However, $\|\nabla_{\mathbf{v}_j} h_{j,k}\|^2$

becomes flat as the value of δ decreases. This implies that very small values of δ can decrease the sensitivity of the radial basis functions to the input vectors included in their receptive fields.

According to Figures 4 and 5, the response of the jth radial basis function to the input $\mathbf{x}_k$ diminishes very quickly outside the blind spot, i.e., as $\|\mathbf{x}_k - \mathbf{v}_j\|^2$ increases above B. This behavior indicates that if a RBF network is constructed using exponential generator functions, the inputs $\mathbf{x}_k$ corresponding to high values of $\|\nabla_{\mathbf{v}_j} h_{j,k}\|^2$ have no significant effect on the response of the radial basis function located at the prototype $\mathbf{v}_j$. As a result, the update of this prototype due to $\mathbf{x}_k$ does not produce significant variations in the input of the upper associative network that implements the desired input-output mapping. Figures 4 and 5 also indicate that the blind spot shrinks as the value of β increases while $\|\nabla_{\mathbf{v}_j} h_{j,k}\|^2$ reaches higher values. Decreasing the value of β expands the blind spot but $\|\nabla_{\mathbf{v}_j} h_{j,k}\|^2$ reaches lower values. In other words, the selection of the value of β in practice involves a trade-off similar to that associated with the selection of the free parameter δ when the radial basis functions are formed by linear generator functions.

8 Handwritten Digit Recognition

8.1 The NIST Databases

Reformulated RBF neural networks were tested and compared with competing techniques on a large-scale handwritten digit recognition problem. The objective of a classifier in this application is the recognition of the digit represented by a binary image of a handwritten numeral. Recognition of handwritten digits is the key component of automated systems developed for a great variety of real-world applications, including mail sorting and check processing. Automated recognition of handwritten digits is not a trivial task due to the high variance of handwritten digits caused by different writing styles, pens, etc. Thus, the development of a reliable system for handwritten digit recognition requires large databases containing a great variety of samples. Such a collection of handwritten digits is contained in the *NIST Special Databases 3*, which contain about 120000 isolated binary digits that have been extracted from sample forms. These

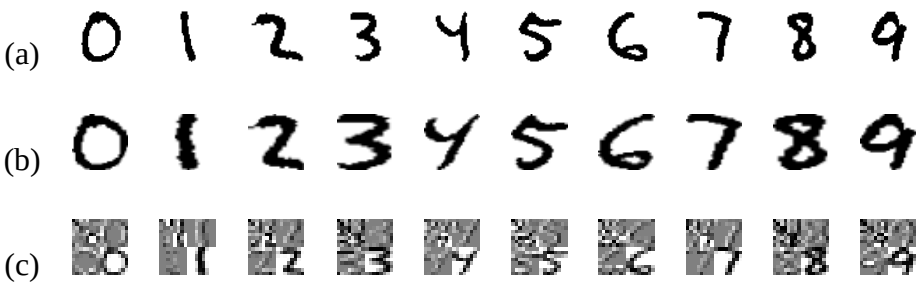


Figure 6. Digits from the NIST Databases: (a) original binary images, (b) 32×32 binary images after one stage of preprocessing (slant and size normalization), and (c) 16×16 images of the digits after two stages of preprocessing (slant and size normalization followed by wavelet decomposition).

digits were handwritten by about 2100 field representatives of the United States Census Bureau. The isolated digits were scanned to produce binary images of size 40×60 pixels, which are centered in a 128×128 box. Figure 6(a) shows some sample digits from 0 to 9 from the NIST databases used in these experiments. The data set was partitioned in three subsets as follows: 58646 digits were used for training, 30367 digits were used for testing, and the remaining 30727 digits constituted the validation set.

8.2 Data Preprocessing

The raw data from the NIST databases were preprocessed in order to reduce the variance of the images that is not relevant to classification. The first stage of the preprocessing scheme produced a slant and size normalized version of each digit. The slant of each digit was found by first determining its center of gravity, which defines an upper and lower half of it. The centers of gravity of each half were subsequently computed and provided an estimate of the vertical main axis of the digit. This axis was then made exactly vertical using a horizontal shear transformation. In the next step, the minimal bounding box was determined and the digit was scaled into a 32×32 box. This scaling may slightly distort the aspect ratio of the digits by centering, if necessary, the digits in the box. Figure 6(b) shows the same digits shown in Figure 6(a) after slant and size normalization.

The second preprocessing stage involved a 4-level wavelet decomposition of the 32×32 digit representation produced by the first preprocessing stage. Each decomposition level includes the application of a 2-D Haar wavelet filter in the decomposed image, followed by downsampling by a factor of 2 along the horizontal and vertical directions. Because of downsampling, each decomposition level produces four subbands of lower resolution, namely a subband that carries background information (containing the low-low frequency components of the original subband), two subbands that carry horizontal and vertical details (containing low-high and high-low frequency components of the original subband), and a subband that carries diagonal details (containing the high-high frequency components of the original subband). As a result, the 4-level decomposition of the original 32×32 image produced three subbands of sizes 16×16, 8×8, and 4×4, and four subbands of size 2×2. The 32×32 image produced by wavelet decomposition was subsequently reduced to an image of size 16×16 by representing each 2×2 window by the average of the four pixels contained in it. This step reduces the amount of data by 3/4 and has a smoothing effect that suppresses the noise present in the 32×32 image [1]. Figure 6(c) shows the images representing the digits shown in Figures 6(a) and 6(b), resulting after the second preprocessing stage described above.

8.3 Classification Tools for NIST Digits

This section begins with a brief description of the variants of the k-*nearest neighbor* (k-NN) classifier used for benchmarking the performance of the neural networks tested in the experiments and also outlines the procedures used for classifying the digits from the NIST databases using neural networks. These procedures involve the formation of the desired input-output mapping and the strategies used to recognize the NIST digits by interpreting the responses of the trained neural networks to the input samples.

The k-NN classifier uses feature vectors from the training set as a reference to classify examples from the testing set. Given an input example from the testing set, the k-NN classifier computes its Euclidean distance from all the examples included in the training set. The k-NN classifier can be implemented to classify all input examples (no rejections allowed)

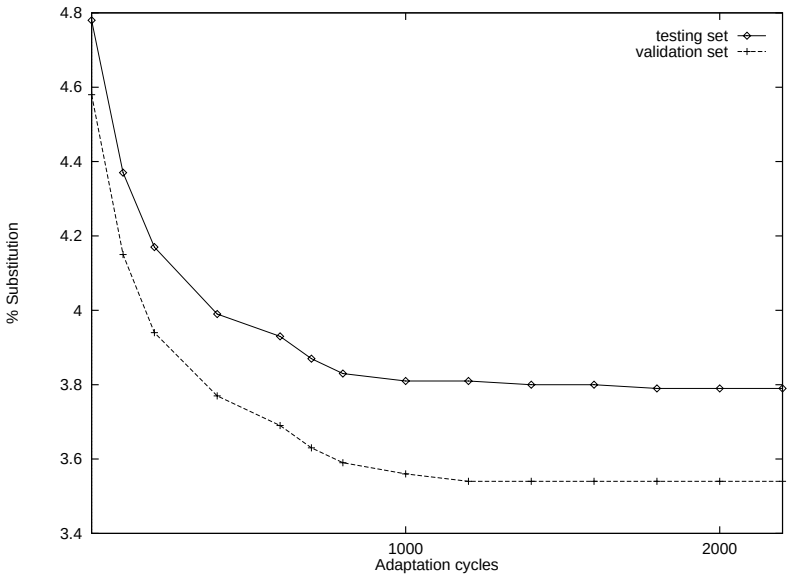
or to selectively reject some ambiguous examples. The k-NN classifier can be implemented using two alternative classification strategies: According to the first and most frequently used classification strategy, each of the k closest training examples to the input example has a vote with a weight equal to 1. According to the second classification strategy, the ith closest training example to the input example has a vote with weight $1/i$; that is, the weight of the closest example is 1, the weight of the second closest example is $1/2$, etc. When no rejections are allowed, the class that receives the largest sum of votes wins the competition and the input example is assigned the corresponding label. The input example is *recognized* if the label assigned by the classifier and the actual label are identical or *substituted* if the assigned and actual labels are different. When rejections are allowed and the k closest training examples to the input example have votes equal to 1, the example is *rejected* if the largest sum of votes is less than k. Otherwise, the input example is classified according to the strategy described above.

The reformulated RBF neural networks and feed-forward neural networks (FFNNs) tested in these experiments consisted of $256 = 16 \times 16$ inputs and 10 linear output units, each representing a digit from 0 to 9. The inputs of the networks were normalized to the interval $[0, 1]$. The learning rate in all these experiments was $\eta = 0.1$. 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 $\mathbf{x}_k \in \mathcal{X}$ corresponding to the digit represented by the ith output unit. The assignment of input vectors to classes was based on a winner-takes-all strategy. More specifically, each input vector was assigned to the class represented by the output unit of the trained RBF neural network with the maximum response. In an attempt to improve the reliability of the neural-network-based classifiers, label assignment was also implemented by employing an alternative scheme that allows the rejection of some ambiguous digits according to the strategy described below: Suppose one of the trained networks is presented with an input vector $\mathbf{x}$ representing a digit from the testing or the validation set and let $\hat{y}_i$, $1 \leq i \leq n_o$, be the responses of its output units. Let $\hat{y}^{(1)} = \hat{y}_{i_1}$ be the maximum among all responses of the output units, that is, $\hat{y}^{(1)} = \hat{y}_{i_1} = \max_{i \in \mathcal{I}_1}\{\hat{y}_i\}$, with $\mathcal{I}_1 = \{1, 2, \ldots, n_o\}$. Let $\hat{y}^{(2)} = \hat{y}_{i_2}$ be the maximum among the responses of the rest of the output units, that is, $\hat{y}^{(2)} = \hat{y}_{i_2} = \max_{i \in \mathcal{I}_2}\{\hat{y}_i\}$, with $\mathcal{I}_2 = \mathcal{I}_1 - \{i_1\}$. The simplest classi-

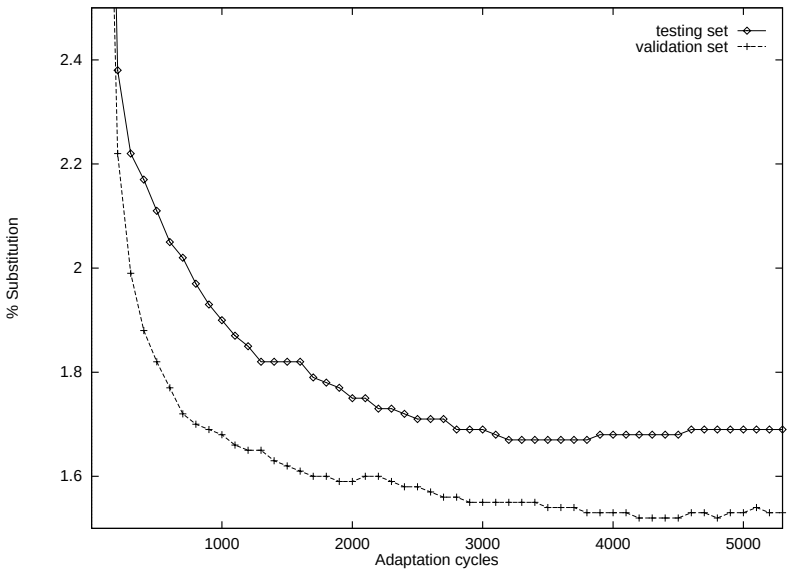
fication scheme would be to assign the digit represented by $\mathbf{x}$ to the i_1th class, which implies that none of the digits would be rejected by the network. Nevertheless, the reliability of this assignment can be improved by comparing the responses $\hat{y}^{(1)}$ and $\hat{y}^{(2)}$ of the two output units that claim the digit for their corresponding classes. If the responses $\hat{y}^{(1)}$ and $\hat{y}^{(2)}$ are sufficiently close, then the digit represented by $\mathbf{x}$ probably lies in a region of the input space where the classes represented by the i_1th and i_2th output units overlap. This indicates that the reliability of classification can be improved by rejecting this digit. This rejection strategy can be implemented by comparing the difference $\Delta\hat{y} = \hat{y}^{(1)} - \hat{y}^{(2)}$ with a *rejection parameter* $r \geq 0$. The digit corresponding to $\mathbf{x}$ is *accepted* if $\Delta\hat{y} \geq r$ and *rejected* otherwise. An accepted digit is *recognized* if the output unit with the maximum response represents the desired class and *substituted* otherwise. The rejection rate depends on the selection of the rejection parameter $r \geq 0$. If $r = 0$, then the digit corresponding to $\mathbf{x}$ is accepted if $\Delta\hat{y} = \hat{y}^{(1)} - \hat{y}^{(2)} \geq 0$, which is by definition true. This implies that none of the input digits is rejected by the network if $r = 0$. The rejection rate increases as the value of the rejection parameter increases above 0.

8.4 Role of the Prototypes in Gradient Descent Learning

RBF neural networks are often trained to implement the desired input-output mapping by updating the output weights, that is, the weights that connect the radial basis functions and the output units, in order to minimize the output error. The radial basis functions are centered at a fixed set of prototypes that define a partition of the input space. In contrast, the gradient descent algorithm presented in Section 6 updates the prototypes representing the centers of the radial basis functions together with the output weights every time training examples are presented to the network. This set of experiments investigated the importance of updating the prototypes during the learning process in order to implement the desired input-output mapping. The reformulated RBF neural networks tested in these experiments contained $c = 256$ radial basis functions obtained in terms of the generator function $g_0(x) = 1 + \delta\, x$, with $g(x) = (g_0(x))^{\frac{1}{1-m}}$, $m = 3$ and $\delta = 10$. In all these experiments the prototypes of the



(a)



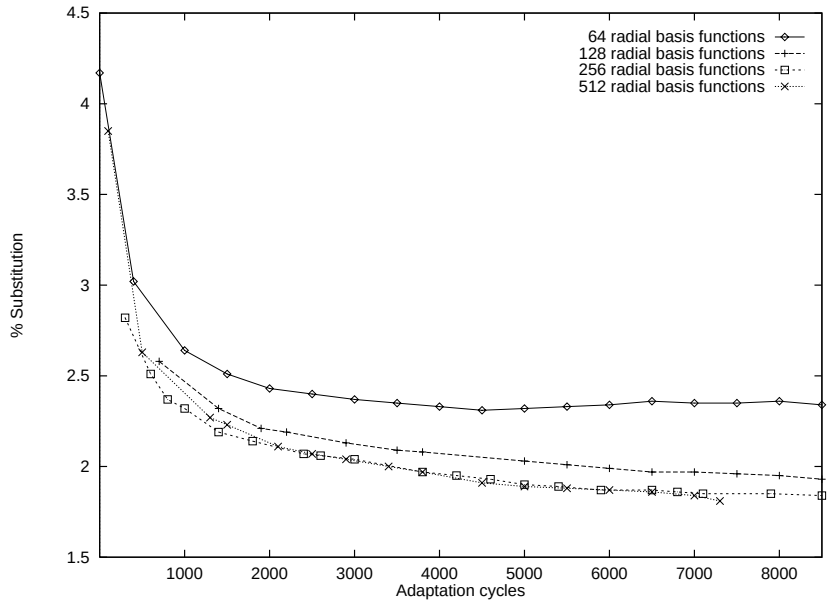
(b)

Figure 7. Performance of a reformulated RBF neural network tested on the testing and validation sets during its training. The network was trained (a) by updating only its output weights, and (b) by updating its output weights and prototypes.

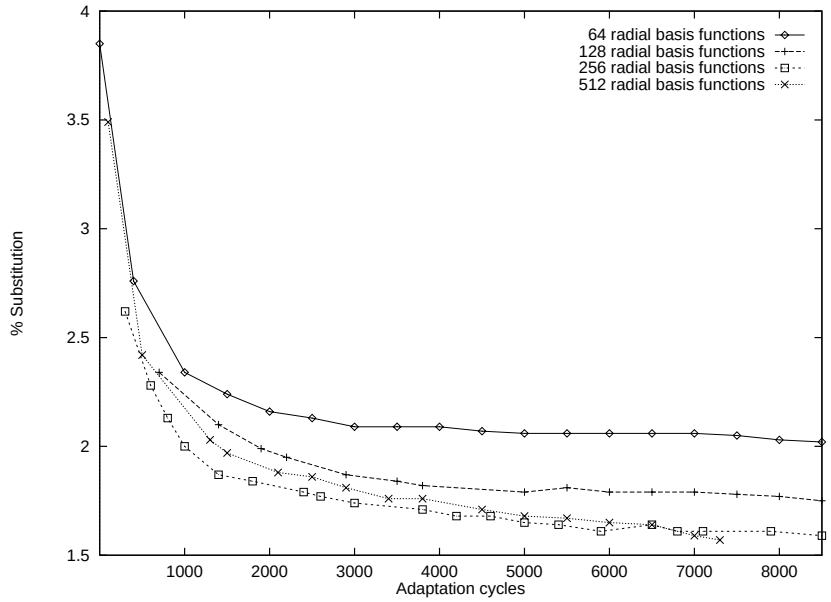
RBF neural networks were determined by employing the initialization scheme described in Section 6, which involves prototype splitting followed by the c-means algorithm. Figure 7 summarizes the performance of the networks trained in these experiments at different stages of the learning process. Figure 7(a) shows the percentage of digits from the testing and validation sets substituted by the RBF network trained by updating only the output weights while keeping the prototypes fixed during the learning process. Figure 7(b) shows the percentage of digits from the testing and validation sets substituted by the reformulated RBF neural network trained by updating the prototypes and the output weights according to the sequential gradient descent learning algorithm presented in Section 6. In both cases, the percentage of substituted digits decreased with some fluctuations during the initial adaptation cycles and remained almost constant after a certain number of adaptation cycles. When the prototypes were fixed during learning, the percentage of substituted digits from both testing and validation sets remained almost constant after 1000 adaptation cycles. In contrast, the percentage of substituted digits decreased after 1000 adaptation cycles and remained almost constant after 3000 adaptation cycles when the prototypes were updated together with the output weights using gradient descent. In this case, the percentage of substituted digits reached 1.69% on the testing set and 1.53% on the validation set. This outcome can be compared with the substitution of 3.79% of the digits from the testing set and 3.54% of the digits from the validation set produced when the prototypes remained fixed during learning. This experimental outcome verifies that the performance of RBF neural networks can be significantly improved by updating all their free parameters during learning according to the training set, including the prototypes that represent the centers of the radial basis functions in the input space.

8.5 Effect of the Number of Radial Basis Functions

This set of experiments evaluated the performance on the testing and validation sets formed from the NIST data of various reformulated RBF neural networks at different stages of their training. The reformulated RBF neural networks contained $c = 64$, $c = 128$, $c = 256$, and $c = 512$ radial basis functions obtained in terms of the generator func-



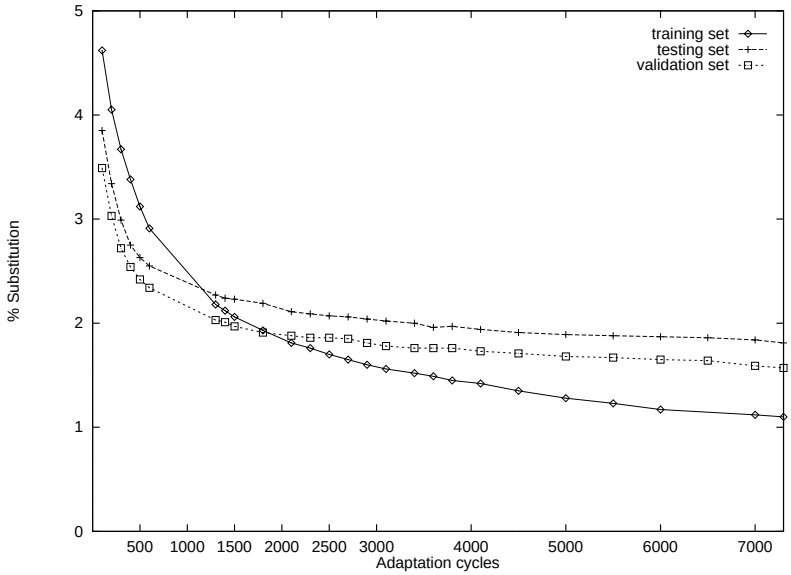
(a)



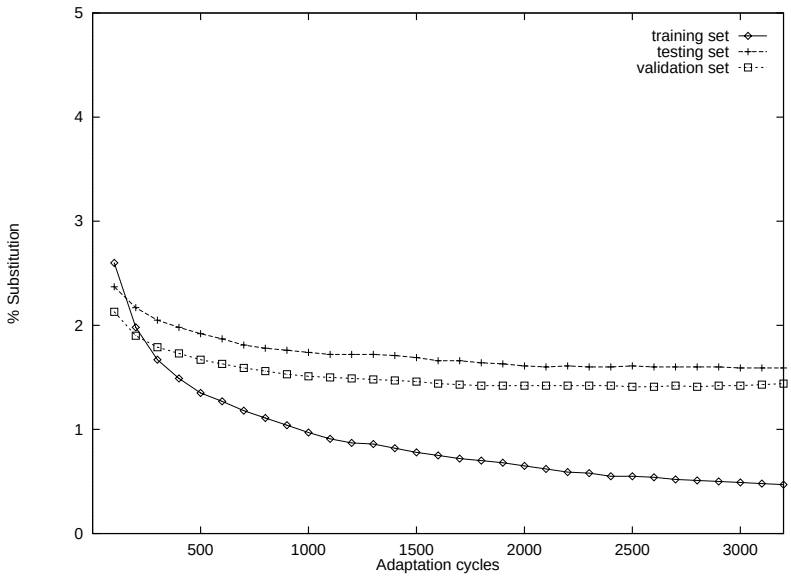
(b)

Figure 8. Performance of reformulated RBF neural networks with different numbers of radial basis functions during their training. The substitution rate was computed (a) on the testing set, and (b) on the validation set.

tion $g_0(x) = 1 + \delta\, x$ as $\phi(x) = g(x^2)$, with $g(x) = (g_0(x))^{\frac{1}{1-m}}$, $m = 3$ and $\delta = 10$. All networks were trained using the sequential gradient descent algorithm described in Section 6. The initial prototypes were computed using the initialization scheme involving prototype splitting. Figures 8(a) and 8(b) plot the percentage of digits from the testing and validation sets, respectively, that were substituted by all four reformulated RBF neural networks as a function of the number of adaptation cycles. Regardless of the number of radial basis functions contained by the reformulated RBF neural networks, their performance on both testing and validation sets improved as the number of adaptation cycles increased. The improvement of the performance was significant during the initial adaptation cycles, which is consistent with the behavior and convergence properties of the gradient descent algorithm used for training. Figures 8(a) and 8(b) also indicate that the number of radial basis functions had a rather significant effect on the performance of reformulated RBF neural networks. The performance of reformulated RBF neural networks on both testing and validation sets improved as the number of radial basis functions increased from $c = 64$ to $c = 128$. The best performance on both sets was achieved by the reformulated RBF neural networks containing $c = 256$ and $c = 512$ radial basis functions. It must be noted that there are some remarkable differences in the performance of these two networks on the testing and validation sets. According to Figure 8(a), the reformulated RBF neural networks with $c = 256$ and $c = 512$ radial basis functions substituted almost the same percentage of digits from the testing set after 1000 adaptation cycles. However, the network with $c = 512$ radial basis functions performed slightly better on the testing set than that containing $c = 256$ radial basis functions when the training continued beyond 7000 adaptation cycles. According to Figure 8(b), the reformulated RBF network with $c = 256$ radial basis functions outperformed consistently the network containing $c = 512$ radial basis functions on the validation set for the first 6000 adaptation cycles. However, the reformulated RBF network with $c = 512$ radial basis functions substituted a smaller percentage of digits from the validation set than the network with $c = 256$ radial basis functions when the training continued beyond 7000 adaptation cycles.



(a)



(b)

Figure 9. Performance of reformulated RBF neural networks with 512 radial basis functions during their training. The substitution rates were computed on the training, testing, and validation sets when gradient descent training was initialized (a) randomly, and (b) by prototype splitting.

8.6 Effect of the Initialization of Gradient Descent Learning

This set of experiments evaluated the effect of the initialization of the supervised learning on the performance of reformulated RBF neural networks trained by gradient descent. The reformulated RBF neural network tested in these experiments contained $c = 512$ radial basis functions constructed as $\phi(x) = g(x^2)$, with $g(x) = (g_0(x))^{\frac{1}{1-m}}$, $g_0(x) = 1 + \delta\, x$, $m = 3$, and $\delta = 10$. The network was trained by the sequential gradient descent algorithm described in Section 6. Figures 9(a) and 9(b) show the percentage of digits from the training, testing, and validation sets substituted during the training process when gradient descent learning was initialized by randomly selecting the prototypes and by prototype splitting, respectively. When the initial prototypes were determined by prototype splitting, the percentage of substituted digits from the training set decreased below 1% after 1000 adaptation cycles and reached values below 0.5% after 3000 adaptation cycles. In contrast, the percentage of substituted digits from the training set decreased much slower and never reached values below 1% when the initial prototypes were produced by a random number generator. When the initial prototypes were initialized by prototype splitting, the percentage of substituted digits from the testing and validation sets decreased to values around 1.5% after the first 1000 adaptation cycles and changed very slightly as the training progressed. When the supervised training was initialized randomly, the percentage of substituted digits from the testing and validation sets decreased much slower during training and reached values higher than those shown in Figure 9(b) even after 7000 adaptation cycles. This experimental outcome indicates that initializing gradient descent learning by prototype splitting improves the convergence rate of gradient descent learning and leads to trained networks that achieve superior performance.

8.7 Benchmarking Reformulated RBF Neural Networks

The last set of experiments compared the performance of reformulated RBF neural networks trained by gradient descent with that of FFNNs

Table 1. Substitution rates on the testing set (S_{test}) and the validation set (S_{val}) produced for different values of k by two variants of the k-NN classifier when no rejections were allowed.

	k-NN classifier (equal vote weights)		k-NN classifier (unequal vote weights)	
k	S_{test}	S_{val}	S_{test}	S_{val}
2	2.351	2.128	2.210	2.018
4	2.025	1.917	2.029	1.852
8	2.055	1.959	1.969	1.836
16	2.259	2.099	1.897	1.832
32	2.496	2.353	1.923	1.875
64	2.869	2.724	2.002	1.949

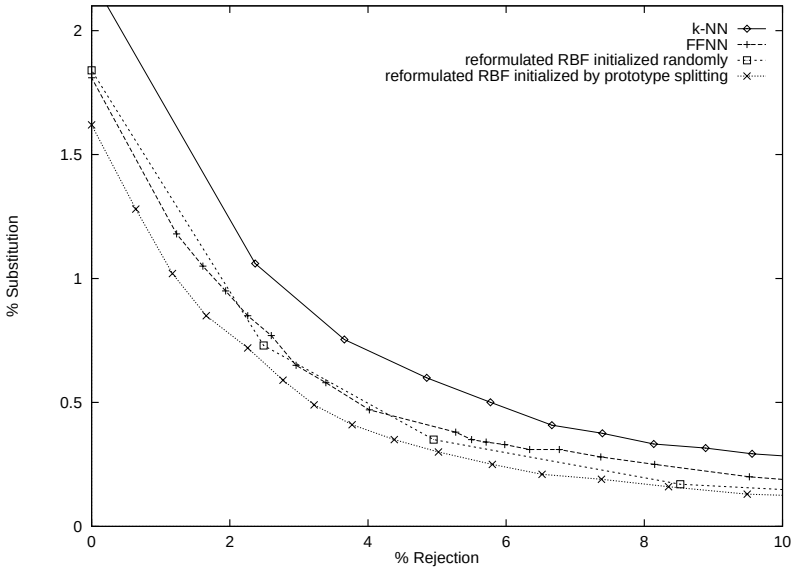
Table 2. Substitution rates on the testing set (S_{test}) and the validation set (S_{val}) produced by different neural-network-based classifiers when no rejections were allowed. FFNNs and reformulated RBF neural networks (RBFNNs) were trained with different numbers c of hidden units by gradient descent. The training of reformulated RBF neural networks was initialized randomly and by prototype splitting.

	FFNN		RBFNN		RBFNN (+ splitting)	
c	S_{test}	S_{val}	S_{test}	S_{val}	S_{test}	S_{val}
64	2.63	2.40	2.31	2.02	2.24	2.03
128	1.82	1.74	1.92	1.75	1.93	1.75
256	1.81	1.59	1.84	1.59	1.62	1.47
512	1.89	1.63	1.81	1.57	1.60	1.41

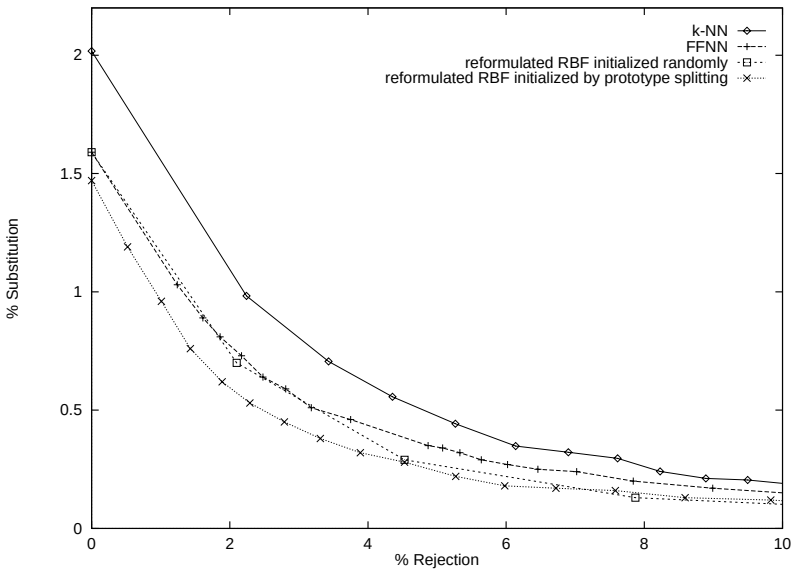
with sigmoidal hidden units and the k-NN classifier. The success rate was first measured when these classifiers were required to assign class labels to all input vectors corresponding to the digits from the testing and validation sets. Table 1 summarizes the substitution rates produced on the testing and validation sets by the two variants of the k-NN algorithm used for recognition when no rejections were allowed. The values of k were powers of two varying from 2 to 64. When each of the k closest training examples voted with weight 1, the smallest substitution rate was recorded for $k = 4$. When each of the k closest training examples voted according to their distance from the input example, the smallest substitution rate was recorded for $k = 16$. In this case, increasing the value of k up to 16 decreased the substitution rate. This can be attributed to

the fact that the votes of all k training examples were weighted with values that decreased from 1 to $1/k$, which reduced the contribution of the most distant among the k training examples. This weighting strategy improved the performance of the k-NN classifier, as indicated by Table 1. When no rejections were allowed, the performance of both variants of the k-NN classifier was inferior to that of the neural networks tested in these experiments. This is clearly indicated by Table 2, which summarizes the substitution rates produced on the testing and validation sets by FFNNs and reformulated RBF neural networks. The number of hidden units varied in these experiments from 64 to 512. The sets of prototypes used for initializing the supervised training of reformulated RBF neural networks were produced by a random number generator and by the prototype splitting procedure outlined in Section 6. The performance of the trained FFNNs on both testing and validation sets improved consistently as the number of hidden units increased from 64 to 256 but degraded when the number of hidden units increased from 256 to 512. In contrast, the performance of reformulated RBF neural networks on both testing and validation sets improved consistently as the number of radial basis function units increased from $c = 64$ to $c = 512$. Both reformulated RBF neural networks trained with $c = 512$ radial basis functions outperformed the best FFNN. Moreover, the performance of the best FFNN was inferior to that of the reformulated RBF neural network trained with $c = 256$ radial basis functions using the initialization scheme employing prototype splitting. The best overall performance among all classifiers evaluated in this set of experiments was achieved by the reformulated RBF neural network trained with $c = 512$ radial basis functions by gradient descent initialized by prototype splitting.

The success rate of the k-NN classifier and the neural-network-based classifiers was also measured when these classifiers were allowed to reject some ambiguous digits in order to improve their reliability. The k-NN classifier was implemented in these experiments by assigning votes equal to 1 to the k closest training examples. This variant of the k-NN classifier does not reject any digit if $k = 1$. The percentage of digits rejected by this variant of the k-NN classifier increases as the value of k increases. The rejection of digits by the FFNN and reformulated RBF neural networks was based on the strategy outlined above. According to this strategy, the percentage of the rejected digits increases as the rejec-



(a)



(b)

Figure 10. Performance of the k-NN classifier, a feed-forward neural network and two reformulated RBF neural networks tested on the NIST digits. The substitution rate is plotted versus the rejection rate (a) on the testing set, and (b) on the validation set.

tion parameter r increases above 0. For $r = 0$, no digits are rejected and classification is based on a winner-takes-all strategy. Figure 10 plots the percentage of digits from the testing and validation sets substituted at different rejection rates by the k-NN classifier, an FFNN with 256 hidden units, and two reformulated RBF neural networks with 256 radial basis functions. Both RBF neural networks were trained by the sequential gradient descent algorithm presented in Section 6. The supervised learning process was initialized in one case by randomly generating the initial set of prototypes and in the other case by determining the initial set of prototypes using prototype splitting. The training of all neural networks tested was terminated based on their performance on the testing set. When no rejections were allowed, all neural networks tested in these experiments performed better than various classification schemes tested on the same data set [10], none of which exceeded the recognition rate of 97.5%. In this case, all neural networks outperformed the k-NN classifier, which classified correctly 97.79% of the digits from the testing set and 97.98% of the digits from the validation set. When no rejections were allowed, the best performance was achieved by the reformulated RBF neural network whose training was initialized by the prototype splitting procedure outlined in Section 6. This network classified correctly 98.38% of the digits from the testing set and 98.53% of the digits from the validation set. According to Figure 10, the percentage of digits from the testing and validation sets substituted by all classifiers tested in these experiments decreased as the rejection rate increased. This experimental outcome verifies that the strategy employed for rejecting ambiguous digits based on the outputs of the trained neural networks is a simple and effective way of dealing with uncertainty. Regardless of the rejection rate, all three neural networks tested in these experiments outperformed the k-NN classifier, which substituted the largest percentage of digits from both testing and validation sets. The performance of the reformulated RBF neural network whose training was initialized by randomly generating the prototypes was close to that of the FFNN. In fact, the FFNN performed better at low rejection rates while this reformulated RBF neural network outperformed the FFNN at high rejection rates. The reformulated RBF neural network initialized by the prototype splitting procedure outlined in Section 6 performed consistently better on the testing and validation sets than the FFNN. The same RBF network outperformed the reformulated RBF neural network initialized randomly on the testing set and on

the validation set for low rejection rates. However, the two reformulated RBF neural networks achieved the same digit recognition rates on the validation set as the rejection rate increased. Among the three networks tested, the best overall performance was achieved by the reformulated RBF neural network whose training was initialized using prototype splitting.

9 Conclusions

This chapter presented an axiomatic approach for reformulating RBF neural networks trained by gradient descent. According to this approach, the development of admissible RBF models reduces to the selection of admissible generator functions that determine the form and properties of the radial basis functions. The reformulated RBF neural networks generated by linear and exponential generator functions can be trained by gradient descent and perform considerably better than conventional RBF neural networks. The criteria proposed for selecting generator functions indicated that linear generator functions have certain advantages over exponential generator functions, especially when reformulated RBF neural networks are trained by gradient descent. Given that exponential generator functions lead to Gaussian radial basis functions, the comparison of linear and exponential generator functions indicated that Gaussian radial basis functions are not the only, and perhaps not the best, choice for constructing RBF neural models. Reformulated RBF neural networks were originally constructed using linear functions of the form $g_0(x) = x + \gamma^2$, which lead to a family of radial basis functions that includes inverse multiquadratic radial basis functions [13], [20], [21]. Subsequent studies, including that presented in the chapter, indicated that linear functions of the form $g_0(x) = 1 + \delta\, x$ facilitate the training and improve the performance of reformulated RBF neural networks [17].

The experimental evaluation of reformulated RBF neural networks presented in this chapter showed that the association of RBF neural networks with erratic behavior and poor performance is unfair to this powerful neural architecture. The experimental results also indicated that the disadvantages often associated with RBF neural networks can only be attributed to the learning schemes used for their training and not to the

models themselves. If the learning scheme used to train RBF neural networks decouples the determination of the prototypes and the updates of the output weights, then the prototypes are simply determined to satisfy the optimization criterion behind the unsupervised algorithm employed. Nevertheless, the satisfaction of this criterion does not necessarily guarantee that the partition of the input space by the prototypes facilitates the implementation of the desired input-output mapping. The simple reason for this is that the training set does not participate in the formation of the prototypes. In contrast, the update of the prototypes during the learning process produces a partition of the input space that is specifically designed to facilitate the input-output mapping. In effect, this partition leads to trained reformulated RBF neural networks that are strong competitors to other popular neural models, including feed-forward neural networks with sigmoidal hidden units.

The results of the experiments on the NIST digits verified that reformulated RBF neural networks trained by gradient descent are strong competitors to classical classification techniques, such as the k-NN, and alternative neural models, such as FFNNs. The digit recognition rates achieved by reformulated RBF neural networks were consistently higher than those of feed-forward neural networks. The classification accuracy of reformulated RBF neural networks was also found to be superior to that of the k-NN classifier. In fact, the k-NN classifier was outperformed by all neural networks tested in these experiments. Moreover, the k-NN classifier was computationally more demanding than all the trained neural networks, which classified examples much faster than the k-NN classifier. The time required by the k-NN to classify an example increased with the problem size (number of examples in the training set), which had absolutely no effect on the classification of digits by the trained neural networks. The experiments on the NIST digits also indicated that the reliability and classification accuracy of trained neural networks can be improved by a recall strategy that allows the rejection of some ambiguous digits.

The experiments indicated that the performance of reformulated RBF neural networks improves when their supervised training by gradient descent is initialized by using an effective unsupervised procedure to determine the initial set of prototypes from the input vectors included in the

training set. An alternative to employing the variation of the c-means algorithm employed in these experiments would be the use of unsupervised algorithms that are not significantly affected by their initialization. The search for such codebook design techniques led to soft clustering [2], [11], [14], [18], [19] and soft learning vector quantization algorithms [12], [15], [16], [18], [19], [24], [27], [35], [41]. Unlike crisp clustering and vector quantization techniques, these algorithms form the prototypes on the basis of *soft* instead of crisp decisions. As a result, this strategy reduces significantly the effect of the initial set of prototypes on the partition of the input vectors produced by such algorithms. The use of soft clustering and LVQ algorithms for initializing the training of reformulated RBF neural networks is a particularly promising approach currently under investigation. Such an initialization approach is strongly supported by recent developments in unsupervised competitive learning, which indicated that the same generator functions used for constructing reformulated RBF neural networks can also be used to generate soft LVQ and clustering algorithms [19], [20], [22].

The generator function can be seen as the concept that establishes a direct relationship between reformulated RBF models and soft LVQ algorithms [20]. 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 [39]. 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.

References

[1] Behnke, S. and Karayiannis, N.B. (1998), "Competitive neural trees for pattern classification," *IEEE Transactions on Neural Networks*, vol. 9, no. 6, pp. 1352-1369.

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

[3] Broomhead, D.S. and Lowe, D. (1988), "Multivariable functional

interpolation and adaptive networks," *Complex Systems*, vol. 2, pp. 321-355.

[4] Cha, I. and Kassam, S.A. (1995), "Interference cancellation using radial basis function networks," *Signal Processing*, vol. 47, pp. 247-268.

[5] Chen, S., Cowan, C.F.N., and Grant, P.M. (1991), "Orthogonal least squares learning algorithm for radial basis function networks," *IEEE Transactions on Neural Networks*, vol. 2, no. 2, pp. 302-309.

[6] Chen, S., Gibson, G.J., Cowan, C.F.N., and Grant, P.M. (1991), "Reconstruction of binary signals using an adaptive radial-basis-function equalizer," *Signal Processing*, vol. 22, pp. 77-93.

[7] Cybenko, G. (1989), "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, pp. 303-314.

[8] Gersho, A. and Gray, R.M. (1992), *Vector Quantization and Signal Compression*, Kluwer Academic, Boston, MA.

[9] Gray, R.M. (1984), "Vector quantization," *IEEE ASSP Magazine*, vol. 1, pp. 4-29.

[10] Grother, P.J. and Candela, G.T. (1993), "Comparison of handprinted digit classifiers," *Technical Report NISTIR 5209*, National Institute of Standards and Technology, Gaithersburg, MD.

[11] Karayiannis, N.B. (1996), "Generalized fuzzy c-means algorithms," *Proceedings of Fifth International Conference on Fuzzy Systems*, New Orleans, LA, pp. 1036-1042.

[12] Karayiannis, N.B. (1997), "Entropy constrained learning vector quantization algorithms and their application in image compression," *SPIE Proceedings vol. 3030: Applications of Artificial Neural Networks in Image Processing II*, San Jose, CA, pp. 2-13.

[13] Karayiannis, N.B. (1997), "Gradient descent learning of radial basis neural networks," *Proceedings of 1997 IEEE International Conference on Neural Networks*, Houston, TX, pp. 1815-1820.

[14] Karayiannis, N.B. (1997), "Fuzzy partition entropies and entropy constrained clustering algorithms," *Journal of Intelligent & Fuzzy Systems*, vol. 5, no. 2, pp. 103-111.

[15] Karayiannis, N.B. (1997), "Learning vector quantization: A review," *International Journal of Smart Engineering System Design*, vol. 1, pp. 33-58.

[16] Karayiannis, N.B. (1997), "A methodology for constructing fuzzy algorithms for learning vector quantization," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 505-518.

[17] Karayiannis, N.B. (1998), "Learning algorithms for reformulated radial basis neural networks," *Proceedings of 1998 International Joint Conference on Neural Networks*, Anchorage, AK, pp. 2230-2235.

[18] Karayiannis, N.B. (1998), "Ordered weighted learning vector quantization and clustering algorithms," *Proceedings of 1998 International Conference on Fuzzy Systems*, Anchorage, AK, pp. 1388-1393.

[19] Karayiannis, N.B. (1998), "Soft learning vector quantization and clustering algorithms based in reformulation," *Proceedings of 1998 International Conference on Fuzzy Systems*, Anchorage, AK, pp. 1441-1446.

[20] Karayiannis, N.B. (1999), "Reformulating learning vector quantization and radial basis neural networks," *Fundamenta Informaticae*, vol. 37, pp. 137-175.

[21] Karayiannis, N.B. (1999), "Reformulated radial basis neural networks trained by gradient descent," *IEEE Transactions on Neural Networks*, vol. 10, no. 3, pp. 657-671.

[22] Karayiannis, N.B. (1999), "An axiomatic approach to soft learning vector quantization and clustering," *IEEE Transactions on Neural Networks*, vol. 10, no. 5, pp. 1153-1165.

[23] Karayiannis, N.B. and Bezdek, J.C. (1997), "An integrated approach to fuzzy learning vector quantization and fuzzy c-means

clustering," *IEEE Transactions on Fuzzy Systems*, vol. 5, no. 4, pp. 622-628.

[24] Karayiannis, N.B., Bezdek, J.C., Pal, N.R., Hathaway, R.J., and Pai, P.-I (1996), "Repairs to GLVQ: A new family of competitive learning schemes," *IEEE Transactions on Neural Networks*, vol. 7, no. 5, pp. 1062-1071.

[25] Karayiannis, N.B. and Mi, W. (1997), "Growing radial basis neural networks: Merging supervised and unsupervised learning with network growth techniques," *IEEE Transactions on Neural Networks*, vol. 8, no. 6, pp. 1492-1506.

[26] Karayiannis, N.B. and Pai, P.-I (1995), "Fuzzy vector quantization algorithms and their application in image compression," *IEEE Transactions on Image Processing*, vol. 4, no. 9, pp. 1193-1201.

[27] Karayiannis, N.B. and Pai, P.-I (1996), "Fuzzy algorithms for learning vector quantization," *IEEE Transactions on Neural Networks*, vol. 7, no. 5, pp. 1196-1211.

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

[29] Kohonen, T. (1989), *Self-Organization and Associative Memory*, 3rd Edition, Springer-Verlag, Berlin.

[30] Kohonen, T. (1990), "The self-organizing map," *Proceeding of the IEEE*, vol. 78, no. 9, pp. 1464-1480.

[31] Linde, Y., Buzo, A., and Gray, R.M. (1980), "An algorithm for vector quantization design," *IEEE Transactions on Communications*, vol. 28, no. 1, pp. 84-95.

[32] Lippmann, R.P. (1989), "Pattern classification using neural networks," *IEEE Communications Magazine*, vol. 27, pp. 47-54.

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

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

[35] Pal, N.R., Bezdek, J.C., and Tsao, E.C.-K. (1993), "Generalized clustering networks and Kohonen's self-organizing scheme," *IEEE Transactions on Neural Networks*, vol. 4, no. 4, pp. 549-557.

[36] Park, J. and Sandberg, I.W. (1991), "Universal approximation using radial-basis-function networks," *Neural Computation*, vol. 3, pp. 246-257.

[37] Park, J. and Sandberg, I.W. (1993), "Approximation and radial-basis-function networks," *Neural Computation*, vol. 5, pp. 305-316.

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

[39] Purushothaman, G. and Karayiannis, N.B. (1997), "Quantum Neural Networks (QNNs): Inherently fuzzy feedforward neural networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 679-693.

[40] Roy, A., Govil, S., and Miranda, R. (1997), "A neural-network learning theory and a polynomial time RBF algorithm," *IEEE Transactions on Neural Networks*, vol. 8, no. 6, pp. 1301-1313.

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

[42] Whitehead, B.A. and Choate, T.D. (1994), "Evolving space-filling curves to distribute radial basis functions over an input space," *IEEE Transactions on Neural Networks*, vol. 5, no. 1, pp. 15-23.