International Scholarly Research Network
ISRN Applied Mathematics
Volume 2012, Article ID 324194, 34 pages
doi:10.5402/2012/324194

Review Article

Using Radial Basis Function Networks for Function Approximation and Classification

Yue Wu,[1] Hui Wang,[1] Biaobiao Zhang,[1] and K.-L. Du[1, 2]

[1] *Enjoyor Laboratories, Enjoyor Inc., Hangzhou 310030, China*
[2] *Department of Electrical and Computer Engineering, Concordia University, Montreal, QC, Canada H3G 1M8*

Correspondence should be addressed to K.-L. Du, kldu@ece.concordia.ca

Received 16 November 2011; Accepted 5 December 2011

Academic Editors: E. Gallopoulos, E. Kita, and M. Sun

The radial basis function (RBF) network has its foundation in the conventional approximation theory. It has the capability of universal approximation. The RBF network is a popular alternative to the well-known multilayer perceptron (MLP), since it has a simpler structure and a much faster training process. In this paper, we give a comprehensive survey on the RBF network and its learning. Many aspects associated with the RBF network, such as network structure, universal approximation capability, radial basis functions, RBF network learning, structure optimization, normalized RBF networks, application to dynamic system modeling, and nonlinear complex-valued signal processing, are described. We also compare the features and capability of the two models.

1. Introduction

The multilayer perceptron (MLP) trained with backpropagation (BP) rule [1] is one of the most important neural network models. Due to its universal function approximation capability, the MLP is widely used in system identification, prediction, regression, classification, control, feature extraction, and associative memory [2]. The RBF network model was proposed by Broomhead and Lowe in 1988 [3]. It has its foundation in the conventional approximation techniques, such as generalized splines and regularization techniques. The RBF network has equivalent capabilities as the MLP model, but with a much faster training speed, and thus it has become a good alternative to the MLP.

The RBF network has its origin in performing exact interpolation of a set of data points in a multidimensional space [4]. It can be considered one type of functional link nets [5]. It has a network architecture similar to the classical regularization network [6], where the basis functions are the Green's functions of the Gram's operator associated with the stabilizer. If the stabilizer exhibits radial symmetry, an RBF network is obtained. From the viewpoint of

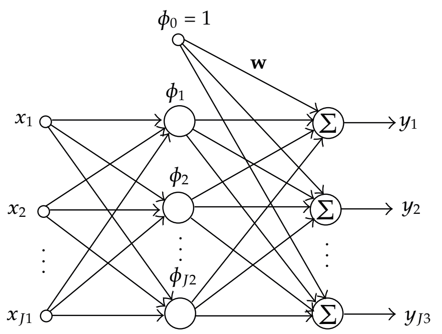

Figure 1: Architecture of the RBF network. The input, hidden, and output layers have J_1, J_2, and J_3 neurons, respectively. $\phi_0(\vec{x}) = 1$ corresponds to the bias in the output layer, while $\phi_i(\vec{x})$'s denote the nonlinearity at the hidden nodes.

approximation theory, the regularization network has three desirable properties [6, 7]. It can approximate any multivariate continuous function on a compact domain to an arbitrary accuracy, given a sufficient number of units; it has the best approximation property since the unknown coefficients are linear. The solution is optimal by minimizing a functional containing a regularization term.

1.1. Network Architecture

The RBF network is a three-layer (J_1-J_2-J_3) feedforward neural network, as shown in Figure 1. Each node in the hidden layer uses a radial basis function (RBF), denoted $\phi(r)$, as its nonlinear activation function. The hidden layer performs a nonlinear transform of the input, and the output layer is a linear combiner mapping the nonlinearity into a new space. Usually, the same RBF is applied on all nodes; that is, the RBF nodes have the nonlinearity $\phi_i(\vec{x}) = \phi(\vec{x} - \vec{c}_i)$, $i = 1, \dots, J_2$, where $\vec{c}_i$ is the prototype or center of the ith node and $\phi(\vec{x})$ is an RBF. The biases of the output layer neurons can be modeled by an additional neuron in the hidden layer, which has a constant activation function $\phi_0(r) = 1$. The RBF network achieves a global optimal solution to the adjustable weights in the minimum mean square error (MSE) sense by using the linear optimization method.

For input $\vec{x}$, the output of the RBF network is given by

$$y_i(\vec{x}) = \sum_{k=1}^{J_2} w_{ki}\phi(\|\vec{x} - \vec{c}_k\|), \quad i = 1, \dots, J_3, \tag{1.1}$$

where $y_i(\vec{x})$ is the ith output, w_{ki} is the connection weight from the kth hidden unit to the ith output unit, and $\|\cdot\|$ denotes the Euclidean norm. The RBF $\phi(\cdot)$ is typically selected as the Gaussian function, and such an RBF network is usually termed the Gaussian RBF network.

For a set of N pattern pairs $\{(\vec{x}_p, \vec{y}_p) \mid p = 1, \dots, N\}$, (1.1) can be expressed in the matrix form

$$\mathbf{Y} = \mathbf{W}^T\mathbf{\Phi}, \tag{1.2}$$

where $\mathbf{W} = [\vec{w}_1, \ldots, \vec{w}_{J_3}]$ is a $J_2 \times J_3$ matrix, $\vec{w}_i = (w_{1i}, \ldots, w_{J_2 i})^T$, $\mathbf{\Phi} = [\vec{\phi}_1, \ldots, \vec{\phi}_N]$ is a $J_2 \times N$ matrix, $\vec{\phi}_p = (\phi_{p,1}, \ldots, \phi_{p,J_2})^T$ is the output of the hidden layer for the pth sample, that is, $\phi_{p,k} = \phi(\|\vec{x}_p - \vec{c}_k\|)$, $\mathbf{Y} = [\vec{y}_1 \ \vec{y}_2 \ \ldots \ \vec{y}_N]$ is a $J_3 \times N$ matrix, and $\vec{y}_p = (y_{p,1}, \ldots, y_{p,J_3})^T$.

The RBF network with a localized RBF such as the Gaussian RBF network is a receptive-field or localized network. The localized approximation method provides the strongest output when the input is near the prototype of a node. For a suitably trained localized RBF network, input vectors that are close to each other always generate similar outputs, while distant input vectors produce nearly independent outputs. This is the intrinsic local generalization property. A receptive-field network is an associative neural network in that only a small subspace is determined by the input to the network. This property is particularly attractive since the modification of the receptive-field function produces local effect. Thus receptive-field networks can be conveniently constructed by adjusting the parameters of the receptive-field functions and/or adding or removing neurons. Another well-known receptive-field network is the cerebellar model articulation controller (CMAC) [8, 9]. The CMAC is a distributed LUT system suitable for VLSI realization. It can approximate slow-varying functions, but may fail in approximating highly nonlinear or rapidly oscillating functions [10, 11].

1.2. Universal Approximation

The RBF network has universal approximation and regularization capabilities. Theoretically, the RBF network can approximate any continuous function arbitrarily well, if the RBF is suitably chosen [6, 12, 13]. A condition for suitable $\phi(\cdot)$ is given by Micchelli's interpolation theorem [14]. A less restrictive condition is given in [15], where $\phi(\cdot)$ is continuous on $(0, \infty)$ and its derivatives satisfy $(-1)^l \phi^{(l)}(x) > 0$, for all $x \in (0, \infty)$ and $l = 0, 1, 2$. The choice of RBF is not crucial to the performance of the RBF network [4, 16]. The Gaussian RBF network can approximate, to any degree of accuracy, any continuous function by a sufficient number of centers $\vec{c}_i$, $i = 1, \ldots, J_2$, and a common standard deviation $\sigma > 0$ in the L_p norm, $p \in [1, \infty]$ [12]. A class of RBF networks can achieve universal approximation when the RBF is continuous and integrable [12]. The requirement of the integrability of the RBF is relaxed in [13]. For an RBF which is continuous almost everywhere, locally essentially bounded and nonpolynomial, the RBF network can approximate any continuous function with respect to the uniform norm [13]. Based on this result, such RBFs as $\phi(r) = \mathrm{e}^{-r/\sigma^2}$ and $\phi(r) = \mathrm{e}^{r/\sigma^2}$ also lead to universal approximation capability [13].

In [17], in an incremental constructive method three-layer feedforward networks with randomly generated hidden nodes are proved to be universal approximators when any bounded nonlinear piecewise continuous activation function is used, and only the weights linking the hidden layer and the output layer need to be adjusted. The proof itself gives an efficient incremental construction of the network. Theoretically the learning algorithms thus derived can be applied to a wide variety of activation functions no matter whether they are sigmoidal or nonsigmoidal, continuous or noncontinuous, or differentiable or nondifferentiable; it can be used to train threshold networks directly. The network learning process is fully automatic, and no user intervention is needed.

This paper is organized as follows. In Section 2, we give a general description to a variety of RBFs. Learning of the RBF network is treated in Section 3. Optimization of the RBF network structure and model selection is described in Section 4. In Section 5, we introduce the normalized RBF network. Section 6 describes the applications of the RBF network to dynamic systems and complex RBF networks to complex-valued signal processing. A comparison between the RBF network and the MLP is made in Section 7. A brief summary is given in Section 8, where topics such as generalizations of the RBF network, robust learning against outliers, and hardware implementation of the RBF network are also mentioned.

2. Radial Basis Functions

A number of functions can be used as the RBF [6, 13, 14]

$$\phi(r) = e^{-r^2/2\sigma^2}, \quad \text{Gaussian}, \tag{2.1}$$

$$\phi(r) = \frac{1}{(\sigma^2 + r^2)^{\alpha}}, \quad \alpha > 0, \tag{2.2}$$

$$\phi(r) = \left(\sigma^2 + r^2\right)^{\beta}, \quad 0 < \beta < 1, \tag{2.3}$$

$$\phi(r) = r, \quad \text{linear}, \tag{2.4}$$

$$\phi(r) = r^2 \ln(r), \quad \text{thin-plate spline}, \tag{2.5}$$

$$\phi(r) = \frac{1}{1 + e^{(r/\sigma^2)-\theta}}, \quad \text{logistic function}, \tag{2.6}$$

where $r > 0$ denotes the distance from a data point $\vec{x}$ to a center $\vec{c}$, σ in (2.1), (2.2), (2.3), and (2.6) is used to control the smoothness of the interpolating function, and θ in (2.6) is an adjustable bias. When β in (2.3) takes the value of $1/2$, the RBF becomes Hardy's multiquadric function, which is extensively used in surface interpolation with very good results [6]. When α in (2.2) is unity, $\phi(r)$ is suitable for DSP implementation [18].

Among these RBFs, (2.1), (2.2), and (2.6) are localized RBFs with the property that $\phi(r) \to 0$ as $r \to \infty$. Physiologically, there exist Gaussian-like receptive fields in cortical cells [6]. As a result, the RBF is typically selected as the Gaussian. The Gaussian is compact and positive. It is motivated from the point of view of kernel regression and kernel density estimation. In fitting data in which there is normally distributed noise with the inputs, the Gaussian is the optimal basis function in the least-squares (LS) sense [19]. The Gaussian is the only factorizable RBF, and this property is desirable for hardware implementation of the RBF network.

Another popular RBF for universal approximation is the thin-plate spline function (2.5), which is selected from a curve-fitting perspective [20]. The thin-plate spline is the solution when fitting a surface through a set of points and by using a roughness penalty [21]. It diverges at infinity and is negative over the region of $r \in (0, 1)$. However, for training purpose, the approximated function needs to be defined only over a specified range. There is some limited empirical evidence to suggest that the thin-plate spline better fits the data in high-dimensional settings [20]. The Gaussian and the thin-plate spline functions are illustrated in Figure 2.

A pseudo-Gaussian function in the one-dimensional space is introduced by selecting the standard deviation σ in the Gaussian (2.1) as two different positive values, namely, σ_- for $x < 0$ and σ_+ for $x > 0$ [22]. This function is extended to the multiple dimensional space by multiplying the pseudo-Gaussian function in each dimension. The pseudo-Gaussian function

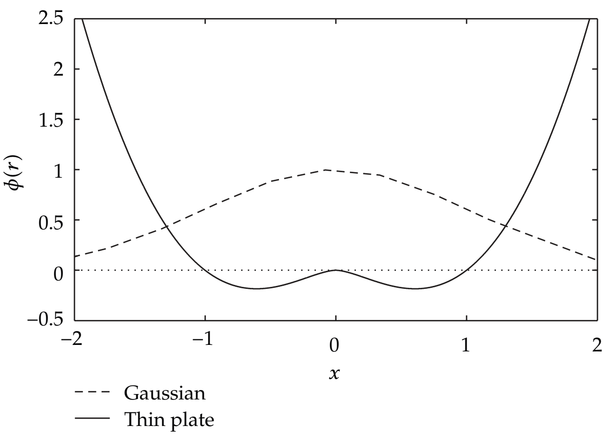

Figure 2: The Gaussian and thin-plate spline functions. $r = |x|$.

is not strictly an RBF due to its radial asymmetry, and this, however, provides the hidden units with a greater flexibility with respect to function approximation.

Approximating functions with nearly constant-valued segments using localized RBFs is most difficult, and the approximation is inefficient. The sigmoidal RBF, as a composite of a set of sigmoidal functions, can be used to deal with this problem [23]

$$\phi(x) = \frac{1}{1 + \mathrm{e}^{-\beta[(x-c)+\theta]}} - \frac{1}{1 + \mathrm{e}^{-\beta[(x-c)-\theta]}}, \tag{2.7}$$

where $\theta > 0$ and $\beta > 0$. $\phi(x)$ is radially symmetric with the maximum at c. β controls the steepness, and θ controls the width of the function. The shape of $\phi(x)$ is approximately rectangular or more exactly soft trapezoidal if $\beta \times \theta$ is large. For small β and θ it is bell shaped. $\phi(x)$ can be extended for n-dimensional approximation by multiplying the corresponding function in each dimension. To accommodate constant values of the desired output and to avoid diminishing the kernel functions, $\phi(\vec{x})$ can be modified by adding a compensating term to the product term $\phi_i(x_i)$ [24]. An alternative approach is to use the raised-cosine function as a one-dimensional RBF [25]. The raised-cosine RBF can represent a constant function exactly using two terms. This RBF can be generalized to n dimensions [25]. Some popular fuzzy membership functions can serve the same purpose by suitably constraining some parameters [2].

The popular Gaussian RBF is circular shaped. Many RBF nodes may be required for approximating a functional behavior with sharp noncircular features. In order to reduce the size of the RBF network, direction-dependent scaling, shaping, and rotation of Gaussian RBFs are introduced in [26] for maximal trend sensing with minimal parameter representations for function approximation, by using a directed graph-based algorithm.

3. RBF Network Learning

RBF network learning can be formulated as the minimization of the MSE function

$$E = \frac{1}{N}\sum_{i=1}^{N}\left\|\vec{y}_p - \mathbf{W}^T\vec{\phi}_p\right\|^2 = \frac{1}{N}\left\|\mathbf{Y} - \mathbf{W}^T\mathbf{\Phi}\right\|_F^2, \tag{3.1}$$

where $\mathbf{Y} = [\vec{y}_1, \vec{y}_2, \ldots, \vec{y}_N]$, $\vec{y}_i$ is the target output for the ith sample in the training set, and $\| \cdot \|_F^2$ is the Frobenius norm defined as $\|\mathbf{A}\|_F^2 = \operatorname{tr}(\mathbf{A}^T \mathbf{A})$.

RBF network learning requires the determination of the RBF centers and the weights. Selection of the RBF centers is most critical to RBF network implementation. The centers can be placed on a random subset or all of the training examples, or determined by clustering or via a learning procedure. One can also use all the data points as centers in the beginning and then selectively remove centers using the k-NN classification scheme [27]. For some RBFs such as the Gaussian, it is also necessary to determine the smoothness parameter σ. Existing RBF network learning algorithms are mainly derived for the Gaussian RBF network and can be modified accordingly when other RBFs are used.

3.1. Learning RBF Centers

RBF network learning is usually performed using a two-phase strategy: the first phase specifies suitable centers $\vec{c}_i$ and their respective standard deviations, also known as widths or radii, σ_i, and the second phase adjusts the network weights $\mathbf{W}$.

3.1.1. Selecting RBF Centers Randomly from Training Sets

A simple method to specify the RBF centers is to randomly select a subset of the input patterns from the training set if the training set is representative of the learning problem. Each RBF center is exactly situated at an input pattern. The training method based on a random selection of centers from a large training set of fixed size is found to be relatively insensitive to the use of pseudoinverse; hence the method itself may be a regularization method [28]. However, if the training set is not sufficiently large or the training set is not representative of the learning problem, learning based on the randomly selected RBF centers may lead to undesirable performance. If the subsequent learning using a selection of random centers is not satisfactory, another set of random centers has to be selected until a desired performance is achieved.

For function approximation, one heuristic is to place the RBF centers at the extrema of the second-order derivative of a function and to place the RBF centers more densely in areas of higher absolute second-order derivative than in areas of lower absolute second-order derivative [29]. As the second-order derivative of a function is associated with its curvature, this achieves a better function approximation than uniformly distributed center placement.

The Gaussian RBF network using the same σ for all RBF centers has universal approximation capability [12]. This global width can be selected as the average of all the Euclidian distances between the ith RBF center $\vec{c}_i$ and its nearest neighbor $\vec{c}_j$, $\sigma = \langle \|\vec{c}_i - \vec{c}_j\| \rangle$. Another simple method for selecting σ is given by $\sigma = d_{\max}/\sqrt{2J_2}$, where $d_{\max}$ is the maximum distance between the selected centers [3]. This choice makes the Gaussian RBF neither too steep nor too flat. The width of each RBF σ_i can be determined according to the data distribution in the region of the corresponding RBF center. A heuristics for selecting σ_i is to average the distances between the ith RBF center and its L nearest neighbors, or, alternatively, σ_i is selected according to the distance of unit i to its nearest neighbor unit j, $\sigma_i = a\|\vec{c}_i - \vec{c}_j\|$, where a is chosen between 1.0 and 1.5.

3.1.2. Selecting RBF Centers by Clustering

Clustering is a data analysis tool for characterizing the distribution of a data set and is usually used for determining the RBF centers. The training set is grouped into appropriate clusters whose prototypes are used as RBF centers. The number of clusters can be specified or determined automatically depending on the clustering algorithm. The performance of the clustering algorithm is important to the efficiency of RBF network learning.

Unsupervised clustering such as the C-means is popular for clustering RBF centers [30]. RBF centers determined by supervised clustering are usually more efficient for RBF network learning than those determined by unsupervised clustering [31], since the distribution of the output patterns is also considered. When the RBF network is trained for classification, the LVQ1 algorithm [32] is popular for clustering the RBF centers. Any unsupervised or supervised clustering algorithm can be used for clustering RBF centers. There are many papers that use clustering to select RBF centers, and these are described in [2]. A survey of clustering algorithms is given in [33].

After the RBF centers are determined, the covariance matrices of the RBFs are set to the covariances of the input patterns in each cluster. In this case, the Gaussian RBF network is extended to the generalized RBF network using the Mahalanobis distance, defined by the weighted norm [6]

$$\phi(\|\vec{x}-\vec{c}_k\|_{\mathbf{A}}) = e^{-(1/2)(\vec{x}-\vec{c}_k)^T\vec{\Sigma}^{-1}(\vec{x}-\vec{c}_k)}, \tag{3.2}$$

where the squared weighted norm $\|\vec{x}\|_{\mathbf{A}}^2 = (\mathbf{A}\vec{x})^T(\mathbf{A}\vec{x}) = \vec{x}^T\mathbf{A}^T\mathbf{A}\vec{x}$ and $\vec{\Sigma}^{-1} = 2\mathbf{A}^T\mathbf{A}$. When the Euclidean distance is employed, one can also select the width of the Gaussian RBF network according to Section 3.1.1.

3.2. Learning the Weights

After RBF centers and their widths or covariance matrices are determined, learning of the weights $\mathbf{W}$ is reduced to a linear optimization problem, which can be solved using the LS method or a gradient-descent method.

After the parameters related to the RBF centers are determined, $\mathbf{W}$ is then trained to minimize the MSE (3.1). This LS problem requires a complexity of $O(NJ_2^2)$ flops for $N > J_2$ when the popular orthogonalization techniques such as SVD and QR decomposition are applied [34]. A simple representation of the solution is given explicitly by [3]

$$\mathbf{W} = \left(\mathbf{\Phi}^T\right)^{\dagger}\mathbf{Y}^T = \left(\mathbf{\Phi}\mathbf{\Phi}^T\right)^{-1}\mathbf{\Phi}\mathbf{Y}^T, \tag{3.3}$$

where $[\cdot]^{\dagger}$ is the pseudoinverse of the matrix within. The over- or underdetermined linear LS system is an ill-conditioned problem. SVD is an efficient and numerically robust technique for dealing with such an ill-conditioned problem and is preferred. For regularly sampled inputs and exact interpolation, $\mathbf{W}$ can be computed by using the Fourier transform of the RBF network [35], which reduces the complexity to $O(N \ln N)$.

When the full data set is not available and samples are obtained on-line, the RLS method can be used to train the weights on-line [36]

$$\begin{aligned}
\vec{w}_i(t) &= \vec{w}_i(t-1) + \vec{k}(t)e_i(t), \\
\vec{k}(t) &= \frac{\mathbf{P}(t-1)\vec{\phi}_t}{\vec{\phi}_t^T\mathbf{P}(t-1)\vec{\phi}_t + \mu}, \\
e_i(t) &= y_{t,i} - \vec{\phi}_t^T \vec{w}_i(t-1), \\
\mathbf{P}(t) &= \frac{1}{\mu}\left[\mathbf{P}(t-1) - \vec{k}(t)\vec{\phi}_t^T\mathbf{P}(t-1)\right],
\end{aligned} \tag{3.4}$$

for $i = 1, \ldots, J_3$, where $0 < \mu \leq 1$ is the forgetting factor. Typically, $\mathbf{P}(0) = a_0\mathbf{I}_{J_2}$, a_0 being a sufficiently large number and $\mathbf{I}_{J_2}$ the $J_2 \times J_2$ identity matrix, and $\vec{w}_i(0)$ is selected as a small random matrix.

In order to eliminate the inversion operation given in (3.3), an efficient, noniterative weight learning technique has been introduced by applying the Gram-Schmidt orthogonalization (GSO) of RBFs [37]. The RBFs are first transformed into a set of orthonormal RBFs for which the optimum weights are computed. These weights are then recomputed in such a way that their values can be fitted back into the original RBF network structure, that is, with kernel functions unchanged. The requirement for computing the off-diagonal terms in the solution of the linear set of weight equations is thus eliminated. In addition, the method has low storage requirements since the weights can be computed recursively, and the computation can be organized in a parallel manner. Incorporation of new hidden nodes does not require recomputation of the network weights already calculated. This allows for a very efficient network training procedure, where network hidden nodes are added one at a time until an adequate error goal is reached. The contribution of each RBF to the overall network output can be evaluated.

3.3. RBF Network Learning Using Orthogonal Least Squares

The orthogonal least-squares (OLS) method [16, 38, 39] is an efficient way for subset model selection. The approach chooses and adds RBF centers one by one until an adequate network is constructed. All the training examples are considered as candidates for the centers, and the one that reduces the MSE the most is selected as a new hidden unit. The GSO is first used to construct a set of orthogonal vectors in the space spanned by the vectors of the hidden unit activation $\vec{\phi}_p$, and a new RBF center is then selected by minimizing the residual MSE. Model selection criteria are used to determine the size of the network.

The batch OLS method can not only determine the weights, but also choose the number and the positions of the RBF centers. The batch OLS can employ the forward [38–40] and the backward [41] center selection approaches. When the RBF centers are distinct, $\mathbf{\Phi}^T$ is of full rank. The orthogonal decomposition of $\mathbf{\Phi}^T$ is performed using QR decomposition

$$\mathbf{\Phi}^T = \mathbf{Q}\begin{bmatrix}\mathbf{R} \\ \mathbf{0}\end{bmatrix}, \tag{3.5}$$

where $\mathbf{Q} = [\vec{q}_1, \ldots, \vec{q}_N]$ is an $N \times N$ orthogonal matrix and $\mathbf{R}$ is a $J_2 \times J_2$ upper triangular matrix. By minimizing the MSE given by (3.1), one can make use of the invariant property of the Frobenius norm

$$E = \frac{1}{N}\left\|\mathbf{Q}^{\mathrm{T}}\mathbf{Y}^{\mathrm{T}} - \mathbf{Q}^{\mathrm{T}}\mathbf{\Phi}^{\mathrm{T}}\mathbf{W}\right\|_F^2. \tag{3.6}$$

Let $\mathbf{Q}^T\mathbf{Y}^T = \left[\begin{smallmatrix}\widetilde{\mathbf{B}}\\ \overline{\mathbf{B}}\end{smallmatrix}\right]$, where $\widetilde{\mathbf{B}} = [\widetilde{b}_{ij}]$ and $\overline{\mathbf{B}} = [\overline{b}_{ij}]$ are, respectively, a $J_2 \times J_3$ and an $(N - J_2) \times J_3$ matrix. We then have

$$E = \frac{1}{N}\left\|\begin{bmatrix}\widetilde{\mathbf{B}} - \mathbf{RW}\\ \overline{\mathbf{B}}\end{bmatrix}\right\|_F^2. \tag{3.7}$$

Thus, the optimal $\mathbf{W}$ is derived from

$$\mathbf{RW} = \widetilde{\mathbf{B}}. \tag{3.8}$$

In this case, the residual $E = (1/N)\|\overline{\mathbf{B}}\|_F^2$.

Due to the orthogonalization procedure, it is very convenient to implement the forward and backward center selection approaches. The forward selection approach is to build up a network by adding, one at a time, centers at the data points that result in the largest decrease in the network output error at each stage. Alternatively, the backward selection algorithm sequentially removes from the network, one at a time, those centers that cause the smallest increase in the residual.

The error reduction ratio (ERR) due to the kth RBF neuron is defined by [39]

$$\mathrm{ERR}_k = \frac{\left(\sum_{i=1}^{J_3}\widetilde{b}_{ki}^2\right)\vec{q}_k^{\,T}\vec{q}_k}{\mathrm{tr}\left(\mathbf{YY}^T\right)}, \quad k = 1, \ldots, N. \tag{3.9}$$

RBF network training can be in a constructive way, and the centers with the largest ERR values are recruited until

$$1 - \sum_{k=1}^{J_2}\mathrm{ERR}_k < \rho, \tag{3.10}$$

where $\rho \in (0, 1)$ is a tolerance.

ERR is a performance-oriented criterion. An alternative terminating criterion can be based on the Akaike information criterion (AIC) [39, 42], which balances between the performance and the complexity. The weights are determined at the same time. The criterion used to stop center selection is a simple threshold on the ERR. If the threshold chose results in very large variances for Gaussian functions, poor generalization performance may occur. To improve generalization, regularized forward OLS methods can be implemented by penalizing large weights [43, 44]. In [45], the training objective is defined by $E + \sum_i^M \lambda_i w_i^2$, where λ_i's are the local regularization parameter and M is the number of weights.

The computation complexity of the orthogonal decomposition of $\mathbf{\Phi}^T$ is $O(NJ_2^2)$. When the size of a training data set N is large, the batch OLS is computationally demanding and also needs a large amount of computer memory.

The RBF center clustering method based on the Fisher ratio class separability measure [46] is similar to the forward selection OLS algorithm [38, 39]. Both the methods employ the QR decomposition-based orthogonal transform to decorrelate the responses of the prototype neurons as well as the forward center selection procedure. The OLS evaluates candidate centers based on the approximation error reduction in the context of nonlinear approximation, while the Fisher ratio-based forward selection algorithm evaluates candidate centers using the Fisher ratio class separability measure for the purpose of classification. The two algorithms have similar computational cost.

Recursive OLS (ROLS) algorithms are proposed for updating the weights of single-input single-output [47] and multi-input multioutput systems [48, 49]. In [48], the ROLS algorithm determines the increment of the weight matrix. In [49], the full weight matrix is determined at each iteration, and this reduces the accumulated error in the weight matrix, and the ROLS has been extended for the selection of the RBF centers. After training with the ROLS, the final triangular system of equations in a form similar to (3.8) contains important information about the learned network and can be used to sequentially select the centers to minimize the network output error. Forward and backward center selection methods are developed from this information, and Akaike's FPE criterion [50] is used in the model selection [49]. The ROLS selection algorithms sort the selected centers in the order of their significance in reducing the MSE [48, 49].

3.4. Supervised Learning of All Parameters

The gradient-descent method provides the simplest solution. We now apply the gradient-descent method to supervised learning of the RBF network.

3.4.1. Supervised Learning for General RBF Networks

To derive the supervised learning algorithm for the RBF network with any useful RBF, we rewrite the error function (3.1) as

$$E = \frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{J_3}\left(e_{n,i}\right)^2, \tag{3.11}$$

where $e_{n,i}$ is the approximation error at the ith output node for the nth example

$$e_{n,i} = y_{n,i} - \sum_{m=1}^{J_2} w_{mi}\phi\left(\left\|\vec{x}_n - \vec{c}_m\right\|\right) = y_{n,i} - \vec{w}_i^T\vec{\phi}_n. \tag{3.12}$$

Taking the first-order derivative of E with respect to w_{mi} and $\vec{c}_m$, respectively, we have

$$\frac{\partial E}{\partial w_{mi}} = -\frac{2}{N}\sum_{n=1}^{N} e_{n,i}\phi(\|\vec{x}_n - \vec{c}_m\|), \quad m = 1,\ldots,J_2, i = 1,\ldots,J_3, \tag{3.13}$$

$$\frac{\partial E}{\partial \vec{c}_m} = \frac{2}{N} w_{mi}\sum_{n=1}^{N} e_{n,i}\dot{\phi}(\|\vec{x}_n - \vec{c}_m\|)\frac{\vec{x}_n - \vec{c}_m}{\|\vec{x}_n - \vec{c}_m\|}, \quad m = 1,\ldots,J_2, i = 1,\ldots,J_3, \tag{3.14}$$

where $\dot{\phi}(\cdot)$ is the first-order derivative of $\phi(\cdot)$.

The gradient-descent method is defined by the update equations

$$\Delta w_{mi} = -\eta_1 \frac{\partial E}{\partial w_{mi}}, \qquad \Delta \vec{c}_m = -\eta_2 \frac{\partial E}{\partial \vec{c}_m}, \tag{3.15}$$

where η_1 and η_2 are learning rates. To prevent the situation that two or more centers are too close or coincide with one another during the learning process, one can add a term such as $\sum_{\alpha \neq \beta} \psi(\|\vec{c}_\alpha - \vec{c}_\beta\|)$ to E, where $\psi(\cdot)$ is an appropriate repulsive potential. The gradient-descent method given by (3.15) needs to be modified accordingly.

Initialization can be based on a random selection of the RBF centers from the examples and $\mathbf{W}$ as a matrix with small random components. One can also use clustering to find the initial RBF centers and the LS to find the initial weights, and the gradient-descent procedure is then applied to refine the learning result. When the gradients given above are set to zero, the optimal solutions to the weights and centers can be derived. The gradient-descent procedure is the iterative approximation to the optimal solutions. For each sample n, if we set $e_{n,i} = 0$, then the right-hand side of (3.13) is zero, we then achieve the global optimum and accordingly get $\vec{y}_n = \mathbf{W}^T\vec{\phi}_n$. For all samples, the result is exactly the same as (1.2). The optimum solution to weights is given by (3.3). Equating (3.14) to zero leads to a formulation showing that the optimal centers are weighted sums of the data points, corresponding to a task-dependent clustering problem.

3.4.2. Supervised Learning for Gaussian RBF Networks

For the Gaussian RBF network, the RBF at each node can be assigned a different width σ_i. The RBFs can be further generalized to allow for arbitrary covariance matrices $\mathbf{\Sigma}_i$

$$\phi_i(\vec{x}) = e^{-(1/2)(\vec{x}-\vec{c}_i)^T \mathbf{\Sigma}_i^{-1}(\vec{x}-\vec{c}_i)}, \tag{3.16}$$

where $\mathbf{\Sigma}_i \in R^{J_1\times J_1}$ is positive definite, symmetric covariance matrix. When $\mathbf{\Sigma}_i^{-1}$ is in general form, the shape and orientation of the axes of the hyperellipsoid are arbitrary in the feature space. If $\mathbf{\Sigma}_i^{-1}$ is a diagonal matrix with nonconstant diagonal elements, it is completely defined by a vector $\vec{\sigma}_i \in R^{J_1}$, and each ϕ_i is a hyperellipsoid whose axes are along the axes of the feature space, $\mathbf{\Sigma}_i^{-1} = \mathrm{diag}(1/\sigma_{i,1}^2,\ldots,1/\sigma_{i,J_1}^2)$. For the J_1-dimensional input space, each RBF using diagonal $\mathbf{\Sigma}_i^{-1}$ has a total of $J_1(J_1+3)/2$ independent adjustable parameters, while each RBF using the same σ in all directions and each RBF using diagonal $\mathbf{\Sigma}_i^{-1}$ have only $J_1 + 1$ and $2J_1$ independent parameters, respectively. There is a trade-off between using a small network with many adjustable parameters and using a large network with fewer adjustable parameters.

When using the RBF using the same σ in all directions, we get the gradients as

$$\begin{aligned}\frac{\partial E}{\partial \vec{c}_m} &= -\frac{2}{N}\sum_{n=1}^{N}\phi_m(\vec{x}_n)\frac{\vec{x}_n-\vec{c}_m}{\sigma_m^2}\sum_{i=1}^{J_3}e_{n,i}w_{i,m},\\ \frac{\partial E}{\partial \sigma_m} &= -\frac{2}{N}\sum_{n=1}^{N}\phi_m(\vec{x}_n)\frac{\|\vec{x}_n-\vec{c}_m\|^2}{\sigma_m^3}\sum_{i=1}^{J_3}e_{n,i}w_{i,m}.\end{aligned} \tag{3.17}$$

Similarly, for the RBF using diagonal $\mathbf{\Sigma}_i^{-1}$, the gradients are given by

$$\begin{aligned}\frac{\partial E}{\partial c_{m,j}} &= -\frac{2}{N}\sum_{n=1}^{N}\phi_m(\vec{x}_n)\frac{x_{n,j}-c_{m,j}}{\sigma_{m,j}^2}\sum_{i=1}^{J_3}e_{n,i}w_{i,m},\\ \frac{\partial E}{\partial \sigma_{m,j}} &= -\frac{2}{N}\sum_{n=1}^{N}\phi_m(\vec{x}_n)\frac{\left(x_{n,j}-c_{m,j}\right)^2}{\sigma_{m,j}^3}\sum_{i=1}^{J_3}e_{n,i}w_{i,m}.\end{aligned} \tag{3.18}$$

Adaptations for $\vec{c}_i$ and $\mathbf{\Sigma}_i$ are along the negative gradient directions. $\mathbf{W}$ are updated by (3.13) and (3.15). To prevent unreasonable radii, the updating algorithms can also be derived by adding to the MSE E a constraint term that penalizes small radii, $E_c = \sum_i 1/\sigma_i$ or $E_c = \sum_{i,j} 1/\sigma_{i,j}$.

3.4.3. Remarks

The gradient-descent algorithms introduced so far are batch learning algorithms. By optimizing the error function E_p for each example $(\vec{x}_p, \vec{y}_p)$, one can update the parameters in the incremental learning model, which are typically much faster than their batch counterparts for suitably selected learning parameters.

Although the RBF network trained by the gradient-descent method is capable of providing equivalent or better performance compared to that of the MLP trained with the BP, the training time for the two methods are comparable [51]. The gradient-descent method is slow in convergence since it cannot efficiently use the locally tuned representation of the hidden-layer units. When the hidden-unit receptive fields, controlled by the widths σ_i, are narrow, for a given input only a few of the total number of hidden units will be activated and hence only these units need to be updated. However, the gradient-descent method may leads to large width, and then the original idea of using a number of local tuning units to approximate the target function cannot be maintained. Besides, the computational advantage of locality cannot be utilized anymore [30].

The gradient-descent method is prone to finding local minima of the error function. For reasonably well-localized RBF, an input will generate a significant activation in a small region, and the opportunity of getting stuck at a local minimum is small. Unsupervised methods can be used to determine σ_i. Unsupervised learning is used to initialize the network parameters, and supervised learning is usually used for fine-tuning the network parameters. The ultimate RBF network learning algorithm is typically a blend of unsupervised and supervised algorithms. Usually, the centers are selected by using a random subset of the training set or obtained by using clustering, the variances are selected using a heuristic, and the weights are solved by using a linear LS method or the gradient-descent method. This combination may yield a fast learning procedure with a sufficient accuracy.

3.5. Other Learning Methods

Actually, all general-purpose unconstrained optimization methods are applicable for RBF network learning by minimization of E, with no or very little modification. These include popular second-order approaches like the Levenberg-Marquardt (LM), conjugate gradient, BFGS, and extended Kalman filtering (EKF), and heuristic-based global optimization methods like evolutionary algorithms, simulated annealing, and Tabu search. These algorithms are described in detail in [2].

The objective is to find suitable network structure and the corresponding network parameters. Some complexity criteria such as the AIC [42] are used to control a trade-off between the learning error and network complexity. Heuristic-based global optimization is also widely used for neural network learning. The selection of the network structure and parameters can be performed simultaneously or separately. For implementation using evolutionary algorithm, the parameters associated with each node are usually coded together in the chromosomes. Some heuristic-based RBF network learning algorithms are described in [2].

The LM method is used for RBF network learning [52–54]. In [53, 54], the LM method is used for estimating nonlinear parameters, and the LS method is used for weight estimation at each iteration. All model parameters are optimized simultaneously. In [54], at each iteration the weights are updated many times during the process of looking for the search direction to update the nonlinear parameters. This further accelerates the convergence of the search process. RBF network learning can be viewed as a system identification problem. After the number of centers is chosen, the EKF simultaneously solves for the prototype vectors and the weight matrix [55]. A decoupled EKF further decreases the complexity of the training algorithm [55]. EKF training provides almost the same performance as gradient-descent training, but with only a fraction of the computational cost. In [56], a pair of parallel running extended Kalman filters are used to sequentially update both the output weights and the RBF centers.

In [57], BP with selective training [58] is applied to RBF network learning. The method improves the performance of the RBF network substantially compared to the gradient-descent method, in terms of convergence speed and accuracy. The method is quite effective when the dataset is error-free and nonoverlapping. In [15], the RBF network is reformulated by using RBFs formed in terms of admissible generator functions and provides a fully supervised gradient-descent training method. A learning algorithm is proposed in [59] for training a special class of reformulated RBF networks, known as cosine RBF networks. It trains reformulated RBF networks by updating selected adjustable parameters to minimize the class-conditional variances at the outputs of their RBFs so as to be capable of identifying uncertainty in data classification.

Linear programming models with polynomial time complexity are also employed to train the RBF network [60]. A multiplication-free Gaussian RBF network with a gradient-based nonlinear learning algorithm [61] is described for adaptive function approximation.

The expectation-maximization (EM) method [62] is an efficient maximum likelihood-based method for parameter estimation; it splits a complex problem into many separate small-scale subproblems. The EM method has also been applied for RBF network learning

[63–65]. The shadow targets algorithm [66], which employs a philosophy similar to that of the EM method, is an efficient RBF network training algorithm for topographic feature extraction.

The RBF network using regression weights can significantly reduce the number of hidden units and is effectively used for approximating nonlinear dynamic systems [22, 25, 64]. For a J_1-J_2-1 RBF network, the weight from the ith hidden unit to the output unit w_i is defined by the linear regression [64] $w_i = \vec{a}_i^T \tilde{\vec{x}} + \xi_i$, where $\vec{a}_i = (a_{i,0}, a_{i,1}, \ldots, a_{i,J_1})^T$ is the regression parameter vector, $\tilde{\vec{x}} = (1, x_1, \ldots, x_{J_1})^T$ is the augmented input vector, and ξ_i is zero-mean Gaussian noise. For the Gaussian RBF network, the RBF centers $\vec{c}_i$ and their widths σ_i can be selected by the C-means and the nearest-neighbor heuristic, while the parameters of the regression weights are estimated by the EM method [64]. The RBF network with linear regression weights has also been studied [25], where a simple but fast computational procedure is achieved by using a high-dimensional raised-cosine RBF.

When approximating a given function $f(x)$, a parsimonious design of the Gaussian RBF network can be achieved based on the Gaussian spectrum of $f(x)$, $\gamma_G(f; \vec{c}, \sigma)$ [67]. According to the Gaussian spectrum, one can estimate the necessary number of RBF units and evaluate how appropriate the use of the Gaussian RBF network is. Gaussian RBFs are selected according to the peaks (negative as well as positive) of the Gaussian spectrum. Only the weights of the RBF network are needed to be tuned. Analogous to principal component analysis (PCA) of the data sets, the principal Gaussian components of $f(x)$ are extracted. If there are a few sharp peaks on the spectrum surface, the Gaussian RBF network is suitable for approximation with a parsimonious architecture. However, if there are many peaks with similar importance or small peaks situated in large flat regions, this method will be inefficient.

The Gaussian RBF network can be regarded as an improved alternative to the four-layer probabilistic neural network (PNN) [68]. In a PNN, a Gaussian RBF node is placed at the position of each training pattern so that the unknown density can be well interpolated and approximated. This technique yields optimal decision surfaces in the Bayes' sense. Training is to associate each node with its target class. This approach, however, severely suffers from the curse of dimensionality and results in a poor generalization. The probabilistic RBF network [69] constitutes a probabilistic version of the RBF network for classification that extends the typical mixture model approach to classification by allowing the sharing of mixture components among all classes. The probabilistic RBF network is an alternative approach for class-conditional density estimation. It provides output values corresponding to the class-conditional densities $p(\vec{x} \mid k)$ (for class k). The typical learning method of probabilistic RBF network for a classification task employs the EM algorithm, which highly depends on the initial parameter values. In [70], a technique for incremental training of the probabilistic RBF network for classification is proposed, based on criteria for detecting a region that is crucial for the classification task. After the addition of all components, the algorithm splits every component of the network into subcomponents, each one corresponding to a different class.

Extreme learning machine (ELM) [71] is a learning algorithm for single-hidden layer feedforward neural networks (SLFNs). ELM in an incremental method (I-ELM) is proved to be a universal approximator [17]. It randomly chooses hidden nodes and analytically determines the output weights of SLFNs. In theory, this algorithm tends to provide good generalization performance at extremely fast learning speed, compared to gradient-based algorithms and SVM. ELM is a simple and efficient three-step learning method, which does not require BP or iterative techniques. ELM can analytically determine all the parameters of SLFNs. Unlike the gradient-based algorithms, the ELM algorithm could be used to train SLFNs with many nondifferentiable activation functions (such as the threshold function) [72].

Theoretically the ELM algorithm can be used to train neural networks with threshold functions directly instead of approximating them with sigmoid functions [72]. In [73], an online sequential ELM (OS-ELM) algorithm is developed to learn data one by one or chunk by chunk with fixed or varying chunk size. In OS-ELM, the parameters of hidden nodes are randomly selected, and the output weights are analytically determined based on the sequentially arriving data. Apart from selecting the number of hidden nodes, no other control parameters have to be manually chosen.

4. Optimizing Network Structure

In order to achieve the optimum structure of an RBF network, learning can be performed by determining the number and locations of the RBF centers automatically using constructive and pruning methods.

4.1. Constructive Approach

The constructive approach gradually increases the number of RBF centers until a criterion is satisfied. The forward OLS algorithm [16] described in Section 3.3 is a well-known constructive algorithm. Based on the OLS algorithm, a constructive algorithm for the generalized Gaussian RBF network is given in [74]. RBF network learning based on a modification to the cascade-correlation algorithm [75] works in a way similar to the OLS method, but with a significantly faster convergence [76]. The OLS incorporated with the sensitivity analysis is also employed to search for the optimal RBF centers [77]. A review on constructive approaches to structural learning of feedforward networks is given in [78].

In [79], a new prototype is created in a region of the input space by splitting an existing prototype $\vec{c}_j$ selected by a splitting criterion, and splitting is performed by adding the perturbation vectors $\pm\vec{e}_j$ to $\vec{c}_j$. The resulting vectors $\vec{c}_j \pm \vec{e}_j$ together with the existing centers form the initial set of centers for the next growing cycle. $\vec{e}_j$ can be obtained by a deviation measure computed from $\vec{c}_j$ and the input vectors represented by $\vec{c}_j$, and $\|\vec{e}_j\| \ll \|\vec{c}_j\|$. Feedback splitting and purity splitting are two criteria for the selection of splitting centers. Existing algorithms for updating the centers $\vec{c}_j$, widths σ_j, and weights can be used. The process continues until a stopping criterion is satisfied.

In a heuristic incremental algorithm [80], the training phase is an iterative process that adds a hidden node $\vec{c}_t$ at each epoch t by an error-driven rule. Each epoch t consists of three phases. The data point with the worst approximation, denoted $\vec{x}_s$, is first recruited as a new hidden node $\vec{c}_t = \vec{x}_s$, and its weight to each output node j, w_{tj}, is fixed at the error at the jth output node performed by the network at the $(t-1)$th epoch on $\vec{x}_s$. The next two phases are, respectively, local tuning and fine tuning of the variances of the RBFs.

The incremental RBF network architecture using hierarchical gridding of the input space [81] allows for a uniform approximation without wasting resources. The centers and variances of the added nodes are fixed through heuristic considerations. Additional layers of Gaussians at lower scales are added where the residual error is higher. The number of Gaussians of each layer and their variances are computed from considerations based on linear filtering theory. The weight of each Gaussian is estimated through a maximum *a posteriori* estimate carried out locally on a subset of the data points. The method shows a high accuracy in the reconstruction, and it can deal with nonevenly spaced data points and is fully parallelizable. Similarly, the hierarchical RBF network [82] is a multiscale version of the

RBF network. It is constituted by hierarchical layers, each containing a Gaussian grid at a decreasing scale. The grids are not completely filled, but units are inserted only where the local error is over a threshold. The constructive approach is based only on the local operations, which do not require any iteration on the data. Like traditional wavelet-based multiresolution analysis (MRA), the hierarchical RBF network employs Riesz bases and enjoys asymptotic approximation properties for a very large class of functions.

The dynamic decay adjustment (DDA) algorithm is a fast constructive training method for the RBF network when used for classification [83]. It is motivated from the probabilistic nature of the PNN [68], the constructive nature of the restricted Coulomb energy (RCE) networks [84] as well as the independent adjustment of the decay factor or width σ_i of each prototype. The DDA method is faster and also achieves a higher classification accuracy than the conventional RBF network [30], the MLP trained with the Rprop, and the RCE.

Incremental RBF network learning is also derived based on the growing cell structures model [85] and based on a Hebbian learning rule adapted from the neural gas model [86]. The insertion strategy is on accumulated error of a subset of the data set. Another example of the constructive approach is the competitive RBF algorithm based on maximum-likelihood classification [87]. The resource-allocating network (RAN) [88] is a well-known RBF network construction method [88] and is introduced in Section 4.2.

4.2. Resource-Allocating Networks

The RAN is a sequential learning method for the localized RBF network, which is suitable for online modeling of nonstationary processes. The network begins with no hidden units. As the pattern pairs are received during the training, a new hidden unit may be recruited according to the novelty in the data. The novelty in the data is decided by two conditions

$$\|\vec{x}_t - \vec{c}_i\| > \varepsilon(t), \tag{4.1}$$

$$\|\vec{e}(t)\| = \left\|\vec{y}_t - f(\vec{x}_t)\right\| > e_{\min}, \tag{4.2}$$

where $\vec{c}_i$ is the center nearest to $\vec{x}_t$, the prediction error $\vec{e} = (e_1, \ldots, e_{J_3})^T$, and $\varepsilon(t)$ and $e_{\min}$ are two thresholds. The algorithm starts with $\varepsilon(t) = \varepsilon_{\max}$, where $\varepsilon_{\max}$ is chosen as the largest scale in the input space, typically the entire input space of nonzero probability. $\varepsilon(t)$ shrinks exponentially by $\varepsilon(t) = \max\{\varepsilon_{\max} e^{-t/\tau}, \varepsilon_{\min}\}$, where τ is a decay constant. $\varepsilon(t)$ is decayed until it reaches $\varepsilon_{\min}$.

Assuming that there are k nodes at time $t-1$, for the Gaussian RBF network, the newly added hidden unit at time t can be initialized as

$$\begin{gathered} \vec{c}_{k+1} = \vec{x}_t, \\ w_{(k+1)j} = e_j(t), \quad j = 1, \ldots, J_3, \\ \sigma_{k+1} = \alpha\|\vec{x}_t - \vec{c}_i\|, \end{gathered} \tag{4.3}$$

where the value for σ_{k+1} is based on the nearest-neighbor heuristic and α is a parameter defining the size of neighborhood. If a pattern pair $(\vec{x}_t, \vec{y}_t)$ does not pass the novelty criteria, no hidden unit is added and the existing network parameters are adapted using the LMS method [89].

The RAN method performs much better than the RBF network learning algorithm using random centers and that using the centers clustered by the C-means [30] in terms of network size and MSE. The RAN method achieves roughly the same performance as the MLP trained with the BP, but with much less computation.

In [90], an agglomerative clustering algorithm is used for RAN initialization, instead of starting from zero hidden node. In [91], the LMS method is replaced by the EKF method for the network parameter adaptation so as to generate a more parsimonious network. Two geometric criteria, namely, the prediction error criterion, which is the same as (4.2), and the angle criterion are also obtained from a geometric viewpoint. The angle criterion assigns RBFs that are nearly orthogonal to all the other existing RBFs. These criteria are proved to be equivalent to Platt's criteria [88]. In [92], the statistical novelty criterion is defined by using the result of the EKF method. By using the EKF method and using this criterion to replace the criteria (4.1) and (4.2), more compact networks and smaller MSEs are achieved than the RAN [88] and the EKF-based RAN [91].

Numerous improvements on the RAN have been made by integrating node-pruning procedure [93–99]. The minimal RAN [93, 94] is based on the EKF-based RAN [91] and achieves a more compact network with equivalent or better accuracy by incorporating a pruning strategy to remove inactive nodes and augmenting the basic growth criterion of the RAN. The output of each RBF unit is linearly scaled to $\hat{o}_i(\vec{x}) \in (0,1]$. If $\hat{o}_i(\vec{x})$ is below a predefined threshold δ for a given number of iterations, this node is idle and can be removed. For a given accuracy, the minimal RAN achieves a smaller complexity than the MLP trained with RProp [100]. In [95], the RAN is improved by using Givens QR decomposition-based RLS for the adaptation of the weights and integrating a node-pruning strategy. The ERR criterion in [38] is used to select the most important regressors. In [96], the RAN is improved by using in each iteration the combination of the SVD and QR-cp methods for determining the structure as well as for pruning the network. In the early phase of learning, the addition of RBFs is in small groups, and this leads to an increased rate of convergence. If a particular RBF is not considered for a given number of iterations, it is removed. The size of the network is more compact than the RAN.

In [97], the EKF and statistical novelty criterion-based method [92] is extended by incorporating an online pruning procedure, which is derived using the parameters and innovation statistics estimated from the EKF. The online pruning method is analogous to the saliency-based optimal brain surgeon (OBS) [101] and optimal brain damage (OBD) [102]. The IncNet and IncNet Pro [103] are RAN-EKF networks with statistically controlled growth criterion. The pruning method is similar to the OBS, but based on the result of the EKF algorithm.

The growing and pruning algorithm for RBF (GAP-RBF) [98] and the generalized GAP-RBF (GGAP-RBF) [99] are RAN-based sequential learning algorithms. These algorithms make use of the notion of significance of a hidden neuron, which is defined as a neuron's statistical contribution over all the inputs seen so far to the overall performance of the network. In addition to the two growing criteria of the RAN, a new neuron is added only when its significance is also above a chosen learning accuracy. If during the training the significance of a neuron becomes less than the learning accuracy, that neuron will be pruned. For each new pattern, only its nearest neuron is checked for growing, pruning, or updating using the EKF. The GGAP-RBF enhances the significance criterion such that it is applicable for training samples with arbitrary sampling density. Both the GAP-RBF and the GGAP-RBF outperform the RAN [88], the EKF-based RAN [91], and the minimal RAN [94] in terms of learning speed, network size, and generalization performance.

4.3. Constructive Methods with Pruning

In addition to the RAN algorithms with pruning strategy [93–99], there are some other constructive methods with pruning.

The normalized RBF network [22] can be sequentially constructed with pruning strategy based on the novelty of the data and the overall behaviour of the network using the gradient-descent method. The network starts from one neuron and adds a new neuron if an example passes two novelty criteria. The first criterion is the same as (4.2), and the second one deals with the activation of the nonlinear neurons, $\max_i \phi_i(\vec{x}_t) < \zeta$, where ζ is a threshold. The pseudo-Gaussian RBF is used, and RBF weights are linear regression functions of the input variables. After the whole pattern set is presented at an epoch, the algorithm starts to remove those neurons that meet any of the three cases, namely, neurons with a very small mean activation for the whole pattern set, neurons with a very small activation region, or neurons having an activation very similar to that of other neurons.

In [104], training starts with zero hidden node and progressively builds the model as new data become available. A fuzzy partition of the input space defines a multidimensional grid, from which the RBF centers are selected, so that at least one selected RBF center is close enough to each input example. The method is capable of adapting on-line the structure of the RBF network, by adding new units when an input example does not belong to any of the subspaces that have been selected, or deleting old ones when no data have been assigned to the respective fuzzy subspaces for a long period of time. The weights are updated using the RLS algorithm. The method avoids selecting the centers only among the available data.

As an efficient and fast growing RBF network algorithm, the constructive nature of DDA [83] may result in too many neurons. The DDA with temporary neurons improves the DDA by introducing online pruning of neurons after each DDA training epoch [105]. After each training epoch, if the individual neurons cover a sufficient number of samples, they are marked as permanent; otherwise, they are deleted. This mechanism results in a significant reduction in the number of neurons. The DDA with selective pruning and model selection is another extension to the DDA [106], where only a portion of the neurons which cover only one training sample are pruned and pruning is carried out only after the last epoch of the DDA training. The method improves the generalization performance of the DDA [83] and the DDA with temporary neurons [105], but yields a larger network size than the DDA with temporary neurons. The pruning strategy proposed in [107] aims to detect and remove those neurons to improve generalization. When the dot product values of two nodes is beyond a threshold, one of the two nodes can be pruned.

4.4. Pruning Methods

Various pruning methods for feedforward networks have been discussed in [2]. These methods are applicable to the RBF network since the RBF network is a kind of feedforward network. Well-known pruning methods are the weight-decay technique [43, 45], the OBD [102], and OBS [101]. Pruning algorithms based on the regularization technique are also popular since additional terms that penalize the complexity of the network are incorporated into the MSE criterion.

With the flavor of weight-decay technique, some regularization techniques for improving the generalization capability of the MLP and the RBF network are also discussed in [108]. As in the MLP, the favored penalty term $\sum w_{ij}^2$ is also appropriate for the RBF network. The widths of the RBFs is a major source of ill-conditioning in RBF network learning, and large width parameters are desirable for better generalization. Some suitable penalty terms for widths are given in [108]. Fault-tolerance ability of RBF networks are also described based on a regularization technique. In [109], a Kullback-Leibler divergence-based objective function is defined for improving the fault-tolerance of RBF networks. The learning method achieves a better fault-tolerant ability, compared with weight-decay-based regularizers. In [110], a regularization-based objective function for training a functional link network to tolerate multiplicative weight noise is defined, and a simple learning algorithm is derived. The function link network is somewhat similar to the RBF network. Under some mild conditions the derived regularizer is essentially the same as a weight decay regularizer. This explains why applying weight decay can also improve the fault-tolerant ability of an RBF with multiplicative weight noise.

In [111], the pruning method starts from a large RBF network and achieves a compact network through an iterative procedure of training and selection. The training procedure adaptively changes the centers and the width of the RBFs and trains the linear weights. The selection procedure performs the elimination of the redundant RBFs using an objective function based on the MDL principle [112]. In [113], all the data vectors are initially selected as centers. Redundant centers in the RBF network are eliminated by merging two centers at each adaptation cycle by using an iterative clustering method. The technique is superior to the traditional RBF network algorithms, particularly in terms of the processing speed and solvability of nonlinear patterns.

4.5. Model Selection

A theoretically well-motivated criterion for describing the generalization error is developed by using Stein's unbiased risk estimator (SURE) [114]

$$\mathrm{Err}(J_2) = \mathrm{err}(J_2) - N\sigma_n^2 + 2\sigma_n^2(J_2 + 1), \tag{4.4}$$

where Err is the generalization error on the new data, err denotes the training error for each model, J_2 is the number of RBF nodes, N is the size of the pattern set, and σ_n^2 is the noise variance, which can be estimated from the MSE of the model. An empirical comparison among the SURE-based method, cross-validation, and the Bayesian information criterion (BIC) [115], is made in [114]. The generalization error of the models by the SURE-based method can be less than that of the models selected by cross-validation, but with much less computation. The SURE-based method has a behavior similar to the BIC. However, the BIC generally gives preference to simpler models since it penalizes complex models more harshly.

The generalization error of a trained network can be decomposed into two parts, namely, an approximation error that is due to the finite number of parameters of the approximation scheme and an estimation error that is due to the finite number of data available [116, 117]. For a feedforward network with J_1 input nodes and a single output node, a bound for the generalization error is given by [116, 117]

$$O\left(\frac{1}{P}\right) + O\left(\left[\frac{PJ_1 \ln(PN) - \ln\delta}{N}\right]^{1/2}\right), \tag{4.5}$$

with a probability greater than $1 - \delta$, where N is the number of examples, $\delta \in (0, 1)$ is the confidence parameter, and P is proportional to the number of parameters such as P hidden units in an RBF network. The first term in (4.5) corresponds to the bound on the approximation error, and the second on the estimation error. As P increases, the approximation error decreases since we are using a larger model; however, the estimation error increases due to overfitting (or alternatively, more data). The trade-off between the approximation and estimation errors is best maintained when P is selected as $P \propto N^{1/3}$ [116]. After suitably selecting P and N, the generalization error for feedforward networks should be $O(1/P)$. This result is similar to that for an MLP with sigmoidal functions [118]. The bound given by (4.5) has been considerably improved to $O((\ln N/N)^{1/2})$ in [119] for RBF network learning with the MSE function.

5. Normalized RBF Networks

The normalized RBF network is defined by normalizing the vector composing of the responses of all the RBF units [30]

$$y_i(\vec{x}) = \sum_{k=1}^{J_2} w_{ki} \widehat{\phi}_k(\vec{x}), \quad i = 1, \ldots, J_3, \tag{5.1}$$

where

$$\widehat{\phi}_k(\vec{x}) = \frac{\phi(\vec{x} - \vec{c}_k)}{\sum_{j=1}^{J_2} \phi(\vec{x} - \vec{c}_j)}. \tag{5.2}$$

Since the normalization operation is nonlocal, the convergence process is computationally costly.

A simple algorithm, called weighted averaging (WAV) [120], is inspired by the functional equivalence of the normalized RBF network of the form (5.1) and fuzzy inference systems (FISs) [121].

The normalized RBF network given by (5.1) can be reformulated such that normalization is performed in the output layer [64, 122]

$$y_i(\vec{x}) = \frac{\sum_{j=1}^{J_2} w_{ji} \phi(\vec{x} - \vec{c}_j)}{\sum_{j=1}^{J_2} \phi(\vec{x} - \vec{c}_j)}. \tag{5.3}$$

As it already receives information from all the hidden units, the locality of the computational processes is preserved. The two forms of the normalized RBF network (5.1) and (5.3) are equivalent, and their similarity with FISs has been pointed out in [121].

In the normalized RBF network of the form (5.3), the traditional roles of the weights and activities in the hidden layer are exchanged. In the RBF network, the weights determine as to how much each hidden node contributes to the output, while in the normalized RBF network the activities of the hidden nodes determine as to which weights contribute the most to the output. The normalized RBF network provides better smoothness than the RBF network. Due to the localized property of the receptive fields, for most data points, there is usually only one hidden node that contributes significantly to (5.3). The normalized RBF network can be trained using a procedure similar to that for the RBF network. The normalized Gaussian RBF network exhibits superiority in supervised classification due to its soft modification rule [123]. It is also a universal approximator in the space of continuous functions with compact support in the space $L^p(R^p, d\vec{x})$ [124].

The normalized RBF network loses the localized characteristics of the localized RBF network and exhibits excellent generalization properties, to the extent that hidden nodes need to be recruited only for training data at the boundaries of the class domains. This obviates the need for a dense coverage of the class domains, in contrast to the RBF network. Thus, the normalized RBF network softens the curse of dimensionality associated with the localized RBF network [122]. The normalized Gaussian RBF network outperforms the Gaussian RBF network in terms of the training and generalization errors, exhibits a more uniform error over the training domain, and is not sensitive to the RBF widths.

The normalized RBF network is an RBF network with a quasilinear activation function with a squashing coefficient decided by the actviations of all the hidden units. The output units can also employ the sigmoidal activation function. The RBF network with the sigmoidal function at the output nodes outperforms the case of linear or quasilinear function at the output nodes in terms of sensitivity to learning parameters, convergence speed as well as accuracy [57].

The normalized RBF network is found functionally equivalent to a class of Takagi-Sugeno-Kang (TSK) systems [121]. According to the output of the normalized RBF network given by (5.3), when the t-norm in the TSK model is selected as algebraic product and the membership functions (MFs) are selected the same as RBFs of the RBF network, the two models are mathematically equivalent [121, 125]. Note that each hidden unit corresponds to a fuzzy rule. In the normalized RBF network, w_{ij}'s typically take constant values; thus the normalized RBF network corresponds to the zero-order TSK model. When the RBF weights are linear regression functions of the input variables [22, 64], the model is functionally equivalent to the first-order TSK model.

6. Applications of RBF Networks

Tradionally, RBF networks are used for function approximation and classification. They are trained to approximate a nonlinear function, and the trained RBF networks are then used to generalize. All applications of the RBF network are based on its universal approximation capability.

RBF networks have now used in a vast variety of applications, such as face tracking and face recognition [126], robotic control [127], antenna design [128], channel equalizations [129, 130], computer vision and graphics [131–133], and solving partial differential equations with boundary conditions (Dirichlet or Neumann) [134].

Special RBFs are customized to match the data characteristics of some problems. For instance, in channel equalization [129, 130], the received signal is complex-valued, and hence complex-valued RBFs are considered.

6.1. Vision Applications

In the graphics or vision applications, the input domain is spherical. Hence, the spherical RBFs [131–133, 135–137] are required.

In a spherical RBF network, the kernel function of the ith RBF node $\phi_i(\vec{s}) = \exp\{-d(\vec{s}, \vec{c}_i)^2/2\Delta^2\}$ is a Gaussian function, where $\vec{s}$ is a unit input vector and Δ is the RBF angular width. The function $d(\vec{s}, \vec{c}_i) = \cos^{-1}(\vec{s} \circ \vec{c}_i)$, where "$\circ$" is the dot product operator, is the distance between two unit vectors on the unit sphere.

6.2. Modeling Dynamic Systems

The sequential RBF network learning algorithms, such as the RAN family and the works in [22, 104], are capable of modifying both the network structure and the output weights on line; thus, these algorithms are particularly suitable for modeling dynamical time-varying systems, where not only the dynamics but the operating region changes with time.

The state-dependent autoregressive (AR) model with functional coefficients is often used to model complex nonlinear dynamical systems. The RBF network can be used as a nonlinear AR time-series model for forecasting [138]. The RBF network can also be used to approximate the coefficients of a state-dependent AR model, yielding the RBF-AR model [139]. The RBF-AR model has the advantages of both the state-dependent AR model for describing nonlinear dynamics and the RBF network for function approximation. The RBF-ARX model is an RBF-AR model with an exogenous variable [54]; it usually uses far fewer RBF centers when compared with the RBF network. The time-delayed neural network (TDNN) model can be an MLP-based or an RBF network-based temporal neural network for nonlinear dynamics and time-series learning [2]. The RBF network-based TDNN [140] uses the same spatial representation of time as the MLP-based TDNN [141]. Learning of the RBF network-based TDNN uses the RBF network learning algorithms.

For time-series applications, the input to the network is $\vec{x}(t) = (y(t-1), \ldots, y(t-n_y))^T$ and the network output is $y(t)$. There are some problems with the RBF network when used as a time-series predictor [142]. First, the Euclidean distance measure is not always appropriate for measuring the similarity between the input vector $\vec{x}_t$ and the prototype $\vec{c}_i$ since $\vec{x}_t$ is itself highly autocorrelated. Second, the node response is radially symmetrical, whereas the data may be distributed differently in each dimension. Third, when the minimum lag n_y is a large number, if all lagged versions of the output $y(t)$ are concatenated as the input vector, the network is too complex, and the performance deteriorates due to irrelevant inputs and an oversized structure. The dual-orthogonal RBF network algorithm overcomes most of these limitations for nonlinear time-series prediction [142]. Motivated by the linear discriminant analysis (LDA) technique, a distance metric is defined based on a classification function of the set of input vectors in order to achieve improved clustering. The forward OLS is used first to determine the significant lags and then to select the RBF centers. In both the steps, the ERR is used for the selection of significant nodes [143].

For online adaptation of nonlinear systems, a constant exponential forgetting factor is commonly applied to all the past data uniformly. This is undesirable for nonlinear systems whose dynamics are different in different operating regions. In [144], online adaptation of the Gaussian RBF network is implemented using a localized forgetting method, which sets different forgetting factors in different regions according to the response of the local prototypes to the current input vector. The method is applied in conjunction with the ROLS [48], and the computing is very efficient.

Recurrent RBF networks, which combine features from the RNN and the RBF network, are suitable for the modeling of nonlinear dynamic systems [145–147]. Time is an internal mechanism and is implicit via recurrent connection. Training of recurrent RBF networks can be based on the RCE algorithm [147], gradient descent, or the EKF [146]. Some techniques for injecting finite state automata into the network have also been proposed in [145].

6.3. Complex RBF Networks

Complex RBF networks are more efficient than the RBF network, in the case of nonlinear signal processing involving complex-valued signals, such as equalization and modeling of nonlinear channels in communication systems. Digital channel equalization can be treated as a classification problem.

In the complex RBF network [130], the input, the output, and the output weights are complex values, whereas the activation function of the hidden nodes is the same as that for the RBF network. The Euclidean distance in the complex domain is defined by [130]

$$d(\vec{x}_t, \vec{c}_i) = \left[(\vec{x}_t - \vec{c}_i)^H (\vec{x}_t - \vec{c}_i)\right]^{1/2}, \tag{6.1}$$

where $\vec{c}_i$ is a J_1-dimensional complex center vector. The Mahalanobis distance (3.16) defined for the Gaussian RBF can be extended to the complex domain [148] by changing the transpose T in (3.16) into the Hermitian transpose H. Most existing RBF network learning algorithms can be easily extended for training various versions of the complex RBF network [129, 130, 148, 149]. When using clustering techniques to determine the RBF centers, the similarity measure can be based on the distance defined by (6.1). The Gaussian RBF is usually used in the complex RBF network. In [129], the minimal RAN algorithm [93] is extended to its complex-valued version.

Although the input and centers of the complex RBF network [129, 130, 148, 149] are complex valued, each RBF node has a real-valued response that can be interpreted as a conditional probability density function. This interpretation makes such a network particularly useful in the equalization application of communication channels with complex-valued signals. This complex RBF network is essentially two separate real-valued RBF networks.

Learning of the complex Gaussian RBF network can be performed in two phases, where the RBF centers are first selected by using the incremental C-means algorithm [33] and the weights are then solved by fixing the RBF parameters [148]. At each iteration t, the C-means first finds the winning node with index w by using the nearest-neighbor rule and then updates both the center and the variance of the winning node by

$$\begin{gathered} \vec{c}_w(t) = \vec{c}_w(t-1) + \eta[\vec{x}_t - \vec{c}_w(t-1)], \\ \mathbf{\Sigma}_w(t) = \mathbf{\Sigma}_w(t-1) + \eta[\vec{x}_t - \vec{c}_w(t-1)][\vec{x}_t - \vec{c}_w(t-1)]^H, \end{gathered} \tag{6.2}$$

where η is the learning rate. The C-means is repeated until the changes in all $\vec{c}_i(t)$ and $\mathbf{\Sigma}_i(t)$ are within a specified accuracy. After complex RBF centers are determined, the weight matrix $\mathbf{W}$ is determined using the LS or RLS algorithm.

In [150], the ELM algorithm is extended to the complex domain, yielding the fully complex ELM (C-ELM). For channel equalization, the C-ELM algorithm significantly outperforms the complex minimal RAN [129], complex RBF network [149], and complex backpropagation, in terms of symbol error rate (SER) and learning speed. In [151], a fully complex-valued RBF network is proposed for regression and classification applications, which is based on the locally regularised OLS (LROLS) algorithm aided with the D-optimality experimental design. These models [150, 151] have a complex-valued response at each RBF node.

7. RBF Networks versus MLPs

Both the MLP and the RBF networks are used for supervised learning. In the RBF network, the activation of an RBF unit is determined by the distance between the input vector and the prototype vector. For classification problems, RBF units map input patterns from a nonlinear separable space to a linear separable space, and the responses of the RBF units form new feature vectors. Each RBF prototype is a cluster serving mainly a certain class. When the MLP with a linear output layer is applied to classification problems, minimizing the error at the output of the network is equivalent to maximizing the so-called network discriminant function at the output of the hidden units [152]. A comparison between the MLP and the localized RBF network (assuming that all units have the RBF with the same width) is as follows.

7.1. Global Method versus Local Method

The MLP is a global method; for an input pattern, many hidden units will contribute to the network output. The localized RBF network is a local method; it satisfies the minimal disturbance principle [153]; that is, the adaptation not only reduces the output error for the current example, but also minimizes disturbance to those already learned. The localized RBF network is biologically plausible.

7.2. Local Minima

The MLP has very complex error surface, resulting in the problem of local minima or nearly flat regions. In contrast, the RBF network has a simple architecture with linear weights, and the LMS adaptation rule is equivalent to a gradient search of a quadratic surface, thus having a unique solution to the weights.

7.3. Approximation and Generalization

The MLP has greater generalization for each training example and is a good candidate for extrapolation. The extension of a localized RBF to its neighborhood is, however, determined by its variance. This localized property prevents the RBF network from extrapolation beyond the training data.

7.4. Network Resources and Curse of Dimensionality

The localized RBF network suffers from the curse of dimensionality. To achieve a specified accuracy, it needs much more data and more hidden units than the MLP. In order to approximate a wide class of smooth functions, the number of hidden units required for the three-layer MLP is polynomial with respect to the input dimensions, while the counterpart for the localized RBF network is exponential [118]. The curse of dimensionality can be alleviated by using smaller networks with more adaptive parameters [6] or by progressive learning [154].

7.5. Hyperlanes versus Hyperellipsoids

For the MLP, the response of a hidden unit is constant on a surface which consists of parallel $(J_1 - 1)$-dimensional hyperplanes in the J_1-dimensional input space. As a result, the MLP is preferable for linear separable problems. In the RBF network the activation of the hidden units is constant on concentric $(J_1 - 1)$-dimensional hyperspheres or hyperellipsoids. Thus, it may be more efficient for linear inseparable classification problems.

7.6. Training and Performing Speeds

The error surface of the MLP has many local minima or large flat regions called plateaus, which lead to slow convergence of the training process for gradient search. For the localized RBF network, only a few hidden units have significant activations for a given input; thus the network modifies the weights only in the vicinity of the sample point and retains constant weights in the other regions. The RBF network requires orders of magnitude less training time for convergence than the MLP trained with the BP rule for comparable performance [3, 30, 155]. For equivalent generalization performance, a trained MLP typically has much less hidden units than a trained localized RBF network and thus is much faster in performing.

Remark 7.1. Generally speaking, the MLP is a better choice if the training data is expensive. However, when the training data is cheap and plentiful or online training is required, the RBF network is very desirable. In addition, the RBF network is insensitive to the order of the presentation of the adjusted signals and hence more suitable for online or subsequent adaptive adjustment [156]. Some properties of the MLP and the RBF network are combined for improving the efficiency of modeling such as the centroid-based MLP [157], the conic section function network [158], a hybrid perceptron node/RBF node scheme [159], and a hybrid RBF sigmoid neural network [160].

8. Concluding Remarks

The RBF network is a good alternative to the MLP. It has a much faster training process compared to the MLP. In this paper, we have given a comprehensive survey of the RBF network. Various aspects of the RBF network have been described, with emphasis placed on RBF network learning and network structure optimization. Topics on normalized RBF networks, RBF networks in dynamic systems modeling, and complex RBF networks for handling nonlinear complex-valued signals are also described. The comparison of the RBF network and the MLP addresses the advantages of each of the two models.

In the support vector machine (SVM) and support vector regression (SVR) approaches, when RBFs are used as kernel function, SVM/SVR training automatically finds the important support vectors (RBF centers) and the weights. Of course, the training objective is not in the MSE sense.

Before we close this paper, we would like also to mention in passing some topics associated with the RBF network. Due to length restriction, we refer to the readers to [2] for detailed exhibition.

8.1. Two Generalizations of the RBF Network

The generalized single-layer network (GSLN) [161, 162] and the wavelet neural network (WNN) [163–165] are two generalizations of the RBF network and use the same three-layer architecture as the RBF network. The GSLN is also known as the generalized linear discriminant. Depending on the set of kernel functions used, the GSLN such as the RBF network and the Volterra network [166] may have broad approximation capabilities. The WNN uses wavelet functions for the hidden units, and it is a universal approximator. Due to the localized properties in both the time and frequency domains of wavelet functions, wavelets are locally receptive field functions which approximate discontinuous or rapidly changing functions due to the multiresolution property of wavelets.

8.2. Robust Learning of RBF Networks

When a training set contains outliers, robust statistics [167] can be applied for RBF network learning. Robust learning algorithms are usually derived from the M-estimator method [2]. The M-estimator replaces the conventional squared error terms by the so-called loss functions. The loss function is a subquadratic function and degrades the effects of those outliers in learning. The derivation of robust RBF network learning algorithms is typically based on the gradient-descent procedure [168, 169].

8.3. Hardware Implementations of RBF Networks

Hardware implementations of neural networks are commonly based on building blocks and thus allow for the inherent parallelism of neural networks. The properties of the MOS transistor are desirable for analog designs of the Gaussian RBF network. In the subthreshold or weak-inversion region, the drain current of the MOS transistor has an exponential dependence on the gate bias and dissipates very low power, and this is usually exploited for designing the Gaussian function [170]. On the other hand, the MOS transistor has a square-law dependence on the bias voltages in its strong-inversion or saturation region. The circuits for the Euclidean distance measure are usually based on the square-law property of the strong-inversion region [171, 172]. There are examples of analog circuits using building blocks [173], pulsed VLSI RBF network chips [172], direct digital VLSI implementations [174, 175], and hybrid VLSI/digital designs [170].

Acknowledgments

The authors acknowledge Professor Chi Sing Leung (Department of Electronic Engineering, City University of Hong Kong) and Professor M. N. S. Swamy (Department of Electrical and Computer Engineering, Concordia University) for their help in improving the quality of this paper. This work was supported in part by NSERC.

References

[1] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, “Learning internal representations by error propagation,” in *Parallel Distributed Processing: Explorations in the Microstructure of Cognition*, D. E. Rumelhart and J. L. McClelland, Eds., vol. 1, pp. 318–362, MIT Press, Cambridge, Mass, USA, 1986.

[2] K.-L. Du and M. N. S. Swamy, *Neural Networks in a Softcomputing Framework*, Springer, London, UK, 2006.

[3] D. S. Broomhead and D. Lowe, “Multivariable functional interpolation and adaptive networks,” *Complex Systems*, vol. 2, no. 3, pp. 321–355, 1988.

[4] M. J. D. Powell, “Radial basis functions for multivariable interpolation: a review,” in *Algorithms for Approximation*, J. C. Mason and M. G. Cox, Eds., vol. 10 of *Institute of Mathematics & Its Applications Conference Series*, pp. 143–167, Oxford University Press, Oxford, UK, 1987.

[5] M. Klaseen and Y. H. Pao, “The functional link net in structural pattern recognition,” in *Proceedings of the IEEE Region Conference on Computer and Communication Systems (IEEE TENCON ’90)*, vol. 2, pp. 567–571, Hong Kong, September 1990.

[6] T. Poggio and F. Girosi, “Networks for approximation and learning,” *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1481–1497, 1990.

[7] F. Girosi and T. Poggio, “Networks and the best approximation property,” *Biological Cybernetics*, vol. 63, no. 3, pp. 169–176, 1990.

[8] J. S. Albus, “A new approach to manipulator control: cerebellar model articulation control (CMAC),” *Journal of Dynamic Systems, Measurement and Control*, vol. 97, no. 3, pp. 220–227, 1975.

[9] W. T. Miller, F. H. Glanz, and L. G. Kraft, “CMAC: an associative neural network alternative to backpropagation,” *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1561–1567, 1990.

[10] N. E. Cotter and T. J. Guillerm, “The CMAC and a theorem of Kolmogorov,” *Neural Networks*, vol. 5, no. 2, pp. 221–228, 1992.

[11] M. Brown, C. J. Harris, and P. C. Parks, “The interpolation capabilities of the binary CMAC,” *Neural Networks*, vol. 6, no. 3, pp. 429–440, 1993.

[12] J. Park and I. W. Sanberg, “Universal approximation using radial-basis-function networks,” *Neural Computation*, vol. 2, pp. 246–257, 1991.

[13] Y. Liao, S. C. Fang, and H. L. W. Nuttle, “Relaxed conditions for radial-basis function networks to be universal approximators,” *Neural Networks*, vol. 16, no. 7, pp. 1019–1028, 2003.

[14] C. A. Micchelli, “Interpolation of scattered data: distance matrices and conditionally positive definite functions,” *Constructive Approximation*, vol. 2, no. 1, pp. 11–22, 1986.

[15] N. B. Karayiannis, “Reformulated radial basis neural networks trained by gradient descent,” *IEEE Transactions on Neural Networks*, vol. 10, no. 3, pp. 657–671, 1999.

[16] S. Chen, S. A. Billings, C. F. N. Cowan, and P. M. Grant, “Practical identification of NARMAX models using radial basis functions,” *International Journal of Control*, vol. 52, no. 6, pp. 1327–1350, 1990.

[17] G. B. Huang, L. Chen, and C. K. Siew, “Universal approximation using incremental constructive feedforward networks with random hidden nodes,” *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 879–892, 2006.

[18] K. L. Du, K. K. M. Cheng, and M. N. S. Swamy, “A fast neural beamformer for antenna arrays,” in *Proceedings of the International Conference on Communications (ICC ’02)*, vol. 1, pp. 139–144, New York, NY, USA, May 2002.

[19] A. R. Webb, “Functional approximation by feed-forward networks: a least-squares approach to generalization,” *IEEE Transactions on Neural Networks*, vol. 5, no. 3, pp. 363–371, 1994.

[20] D. Lowe, “On the use of nonlocal and non positive definite basis functions in radial basis function networks,” in *Proceedings of the 4th International Conference on Artificial Neural Networks*, pp. 206–211, Cambridge, UK, 1995.

[21] J. Meinguet, “Multivariate interpolation at arbitrary points made simple,” *Journal of Applied Mathematics and Physics*, vol. 30, no. 2, pp. 292–304, 1979.

[22] I. Rojas, H. Pomares, J. L. Bernier et al., “Time series analysis using normalized PG-RBF network with regression weights,” *Neurocomputing*, vol. 42, pp. 267–285, 2002.

[23] C. C. Lee, P. C. Chung, J. R. Tsai, and C. I. Chang, “Robust radial basis function neural networks,” *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 29, no. 6, pp. 674–685, 1999.

[24] S. J. Lee and C. L. Hou, “An ART-based construction of RBF networks,” *IEEE Transactions on Neural Networks*, vol. 13, no. 6, pp. 1308–1321, 2002.

[25] R. J. Schilling, J. J. Carroll, and A. F. Al-Ajlouni, “Approximation of nonlinear systems with radial basis function neural networks,” *IEEE Transactions on Neural Networks*, vol. 12, no. 1, pp. 1–15, 2001.

[26] P. Singla, K. Subbarao, and J. L. Junkins, “Direction-dependent learning approach for radial basis function networks,” *IEEE Transactions on Neural Networks*, vol. 18, no. 1, pp. 203–222, 2007.

[27] P. A. Devijver and J. Kittler, *Pattern Recognition: A Statistical Approach*, Prentice-Hall International, Englewood Cliffs, NJ, USA, 1982.

[28] S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, Upper Saddle River, NJ, USA, 2nd edition, 1999.

[29] A. V. D. Sanchez, "Second derivative dependent placement of RBF centers," *Neurocomputing*, vol. 7, no. 3, pp. 311–317, 1995.

[30] J. Moody and C. J. Darken, "Fast learning in networks of locally-tuned processing units," *Neural Computation*, vol. 1, no. 2, pp. 281–294, 1989.

[31] C. L. Chen, W. C. Chen, and F. Y. Chang, "Hybrid learning algorithm for Gaussian potential function networks," *IEE Proceedings D*, vol. 140, no. 6, pp. 442–448, 1993.

[32] T. Kohonen, *Self-Organization and Associative Memory*, vol. 8 of *Springer Series in Information Sciences*, Springer, Berlin, Germany, 3rd edition, 1989.

[33] K. L. Du, "Clustering: a neural network approach," *Neural Networks*, vol. 23, no. 1, pp. 89–107, 2010.

[34] G. H. Golub and C. F. van Loan, *Matrix Computation*, John Hopkins University Press, Baltimore, Md, USA, 2nd edition, 1989.

[35] Y. Abe and Y. Iiguni, "Fast computation of RBF coefficients for regularly sampled inputs," *Electronics Letters*, vol. 39, no. 6, pp. 543–544, 2003.

[36] K. J. Astrom and B. Wittenmark, *Adaptive Control*, Addison-Wesley, Reading, Mass, USA, 1995.

[37] W. Kaminski and P. Strumillo, "Kernel orthonormalization in radial basis function neural networks," *IEEE Transactions on Neural Networks*, vol. 8, no. 5, pp. 1177–1183, 1997.

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

[39] S. Chen, P. M. Grant, and C. F. N. Cowan, "Orthogonal least-squares algorithm for training multioutput radial basis function networks," *IEE Proceedings, Part F*, vol. 139, no. 6, pp. 378–384, 1992.

[40] S. Chen and J. Wigger, "Fast orthogonal least squares algorithm for efficient subset model selection," *IEEE Transactions on Signal Processing*, vol. 43, no. 7, pp. 1713–1715, 1995.

[41] X. Hong and S. A. Billings, "Givens rotation based fast backward elimination algorithm for RBF neural network pruning," *IEE Proceedings: Control Theory and Applications*, vol. 144, no. 5, pp. 381–384, 1997.

[42] H. Akaike, "A new look at the statistical model identification," *Institute of Electrical and Electronics Engineers*, vol. 19, pp. 716–723, 1974.

[43] M. J. L. Orr, "Regularization in the selection of radial basis function centers," *Neural Computation*, vol. 7, no. 3, pp. 606–623, 1995.

[44] S. Chen, E. S. Chng, and K. Alkadhimi, "Regularized orthogonal least squares algorithm for constructing radial basis function networks," *International Journal of Control*, vol. 64, no. 5, pp. 829–837, 1996.

[45] S. Chen, X. Hong, C. J. Harris, and P. M. Sharkey, "Sparse modelling using orthogonal forward regression with press statistic and regularization," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 34, no. 2, pp. 898–911, 2004.

[46] K. Z. Mao, "RBF neural network center selection based on Fisher ratio class separability measure," *IEEE Transactions on Neural Networks*, vol. 13, no. 5, pp. 1211–1217, 2002.

[47] J. E. Bobrow and W. Murray, "An algorithm for RLS identification of parameters that vary quickly with time," *Institute of Electrical and Electronics Engineers*, vol. 38, no. 2, pp. 351–354, 1993.

[48] D. L. Yu, J. B. Gomm, and D. Williams, "A recursive orthogonal least squares algorithm for training RBF networks," *Neural Processing Letters*, vol. 5, no. 3, pp. 167–176, 1997.

[49] J. B. Gomm and D. L. Yu, "Selecting radial basis function network centers with recursive orthogonal least squares training," *IEEE Transactions on Neural Networks*, vol. 11, no. 2, pp. 306–314, 2000.

[50] H. Akaike, "Fitting autoregressive models for prediction," *Annals of the Institute of Statistical Mathematics*, vol. 21, pp. 243–247, 1969.

[51] D. Wettschereck and T. Dietterich, "Improving the performance of radial basis function networks by learning center locations," in *Advances in Neural Information Processing Systems*, J. E. Moody, S. J. Hanson, and R. P. Lippmann, Eds., vol. 4, pp. 1133–1140, Morgan Kauffmann, San Mateo, Calif, USA, 1992.

[52] D. Gorinevsky, "An approach to parametric nonlinear least square optimization and application to task-level learning control," *Institute of Electrical and Electronics Engineers*, vol. 42, no. 7, pp. 912–927, 1997.

[53] S. McLoone, M. D. Brown, G. Irwin, and G. Lightbody, "A hybrid linear/nonlinear training algorithm for feedforward neural networks," *IEEE Transactions on Neural Networks*, vol. 9, no. 4, pp. 669–684, 1998.

[54] H. Peng, T. Ozaki, V. Haggan-Ozaki, and Y. Toyoda, "A parameter optimization method for radial basis function type models," *IEEE Transactions on Neural Networks*, vol. 14, no. 2, pp. 432–438, 2003.

[55] D. Simon, "Training radial basis neural networks with the extended Kalman filter," *Neurocomputing*, vol. 48, pp. 455–475, 2002.

[56] I. B. Ciocoiu, "RBF networks training using a dual extended Kalman filter," *Neurocomputing*, vol. 48, pp. 609–622, 2002.

[57] M.-T. Vakil-Baghmisheh and N. Pavešić, "Training RBF networks with selective backpropagation," *Neurocomputing*, vol. 62, no. 1–4, pp. 39–64, 2004.

[58] M. T. Vakil-Baghmisheh and N. Pavešić, "A fast simplified fuzzy ARTMAP network," *Neural Processing Letters*, vol. 17, no. 3, pp. 273–316, 2003.

[59] N. B. Karayiannis and Y. Xiong, "Training reformulated radial basis function neural networks capable of identifying uncertainty in data classification," *IEEE Transactions on Neural Networks*, vol. 17, no. 5, pp. 1222–1233, 2006.

[60] A. Roy, S. Govil, and R. Miranda, "An algorithm to generate radial basis function (RBF)-like nets for classification problems," *Neural Networks*, vol. 8, no. 2, pp. 179–201, 1995.

[61] M. Heiss and S. Kampl, "Multiplication-free radial basis function network," *IEEE Transactions on Neural Networks*, vol. 7, no. 6, pp. 1461–1464, 1996.

[62] A. P. Dempster, N. M. Laird, and D. B. Rubin, "Maximum likelihood from incomplete data via the EM algorithm," *Journal of the Royal Statistical Society. Series B. Methodological*, vol. 39, no. 1, pp. 1–38, 1977.

[63] M. Lázaro, I. Santamaría, and C. Pantaleón, "A new EM-based training algorithm for RBF networks," *Neural Networks*, vol. 16, no. 1, pp. 69–77, 2003.

[64] R. Langari, L. Wang, and J. Yen, "Radial basis function networks, regression weights, and the expectation—maximization algorithm," *IEEE Transactions on Systems, Man, and Cybernetics Part A*, vol. 27, no. 5, pp. 613–623, 1997.

[65] A. Ukrainec and S. Haykin, "A mutual information-based learning strategy and its application to radar," in *Fuzzy Logic and Neural Network Handbook*, C. H. Chen, Ed., pp. 12.3–12.26, McGraw-Hill, New York, NY, USA, 1996.

[66] M. E. Tipping and D. Lowe, "Shadow targets: a novel algorithm for topographic projections by radial basis functions," *Neurocomputing*, vol. 19, no. 1–3, pp. 211–222, 1998.

[67] P. András, "Orthogonal RBF neural network approximation," *Neural Processing Letters*, vol. 9, no. 2, pp. 141–151, 1999.

[68] D. F. Specht, "Probabilistic neural networks," *Neural Networks*, vol. 3, no. 1, pp. 109–118, 1990.

[69] M. K. Titsias and A. C. Likas, "Shared kernel models for class conditional density estimation," *IEEE Transactions on Neural Networks*, vol. 12, no. 5, pp. 987–997, 2001.

[70] C. Constantinopoulos and A. Likas, "An incremental training method for the probabilistic RBF network," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 966–974, 2006.

[71] G. B. Huang, Q. Y. Zhu, and C. K. Siew, "Extreme learning machine: theory and applications," *Neurocomputing*, vol. 70, no. 1–3, pp. 489–501, 2006.

[72] G. B. Huang, Q. Y. Zhu, K. Z. Mao, C. K. Siew, P. Saratchandran, and N. Sundararajan, "Can threshold networks be trained directly?" *IEEE Transactions on Circuits and Systems II*, vol. 53, no. 3, pp. 187–191, 2006.

[73] N. Y. Liang, G. B. Huang, P. Saratchandran, and N. Sundararajan, "A fast and accurate online sequential learning algorithm for feedforward networks," *IEEE Transactions on Neural Networks*, vol. 17, no. 6, pp. 1411–1423, 2006.

[74] X. X. Wang, S. Chen, and D. J. Brown, "An approach for constructing parsimonious generalized Gaussian kernel regression models," *Neurocomputing*, vol. 62, no. 1–4, pp. 441–457, 2004.

[75] S. E. Fahlman and C. Lebiere, "The cascade-correlation learning architecture," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, Ed., vol. 2, pp. 524–532, Morgan Kauffmann, San Mateo, Calif, USA, 1990.

[76] M. Lehtokangas, J. Saarinen, and K. Kaski, "A ccelerating training of radial basis function networks with Cascade-Correlation algorithm," *Neurocomputing*, vol. 9, no. 2, pp. 207–213, 1995.

[77] D. Shi, J. Gao, D. S. Yeung, and F. Chen, "Radial basis function network pruning by sensitivity analysis," in *Proceedings of the 17th Conference of the Canadian Society for Computational Studies of Intelligence, Advances in Artificial Intelligence*, A. Y. Tawfik and S. D. Goodwin, Eds., vol. 3060 of *Lecture Notes in Artificial Intelligence*, pp. 380–390, Springer, Ontario, Canada, May 2004.

[78] T. Y. Kwok and D. Y. Yeung, "Constructive algorithms for structure learning in feedforward neural networks for regression problems," *IEEE Transactions on Neural Networks*, vol. 8, no. 3, pp. 630–645, 1997.

[79] N. B. Karayiannis and G. W. Mi, "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, 1997.

[80] A. Esposito, M. Marinaro, D. Oricchio, and S. Scarpetta, "Approximation of continuous and discontinuous mappings by a growing neural RBF-based algorithm," *Neural Networks*, vol. 13, no. 6, pp. 651–665, 2000.

[81] N. A. Borghese and S. Ferrari, "Hierarchical RBF networks and local parameters estimate," *Neurocomputing*, vol. 19, no. 1–3, pp. 259–283, 1998.

[82] S. Ferrari, M. Maggioni, and N. A. Borghese, "Multiscale approximation with hierarchical radial basis functions networks," *IEEE Transactions on Neural Networks*, vol. 15, no. 1, pp. 178–188, 2004.

[83] M .R. Berthold and J. Diamond, "Boosting the performance of RBF networks with dynamic decay adjustment," in *Advances In Neural Information Processing Systems*, G. Tesauro, D. S. Touretzky, and T. Leen, Eds., vol. 7, pp. 521–528, MIT Press, New York, NY, USA, 1995.

[84] D. L. Reilly, L. N. Cooper, and C. Elbaum, "A neural model for category learning," *Biological Cybernetics*, vol. 45, no. 1, pp. 35–41, 1982.

[85] B. Fritzke, "Supervised learning with growing cell structures," in *Advances in Neural Information Processing Systems*, J. D. Cowan, G. Tesauro, and J. Alspector, Eds., vol. 6, pp. 255–262, Morgan Kauffmann, San Mateo, Calif, USA, 1994.

[86] B. Fritzke, "Fast learning with incremental RBF networks," *Neural Processing Letters*, vol. 1, no. 1, pp. 2–5, 1994.

[87] Q. Zhu, Y. Cai, and L. Liu, "A global learning algorithm for a RBF network," *Neural Networks*, vol. 12, no. 3, pp. 527–540, 1999.

[88] J. Platt, "A resource allocating network for function interpolation," *Neural Computation*, vol. 3, no. 2, pp. 213–225, 1991.

[89] B. Widrow and M. E. Hoff, "Adaptive switching circuits," in *IRE Eastern Electronic Show & Convention (WESCON1960), Convention Record*, vol. 4, pp. 96–104, 1960.

[90] M. Wallace, N. Tsapatsoulis, and S. Kollias, "Intelligent initialization of resource allocating RBF networks," *Neural Networks*, vol. 18, no. 2, pp. 117–122, 2005.

[91] V. Kadirkamanathan and M. Niranjan, "A function estimation approach to sequential learning with neural network," *Neural Computation*, vol. 5, no. 6, pp. 954–975, 1993.

[92] V. Kadirkamanathan, "A statistical inference based growth criterion for the RBF network," in *Proceedings of the IEEE Workshop Neural Networks for Signal Processing*, pp. 12–21, Ermioni, Greece, 1994.

[93] L. Yingwei, N. Sundararajan, and P. Saratchandran, "A sequential learning scheme for function approximation using minimal radial basis function neural networks," *Neural Computation*, vol. 9, no. 2, pp. 461–478, 1997.

[94] L. Yingwei, N. Sundararajan, and P. Saratchandran, "Performance evaluation of a sequential minimal Radial Basis Function (RBF) neural network learning algorithm," *IEEE Transactions on Neural Networks*, vol. 9, no. 2, pp. 308–318, 1998.

[95] R. Rosipal, M. Koska, and I. Farkaš, "Prediction of chaotic time-series with a resource-allocating RBF network," *Neural Processing Letters*, vol. 7, no. 3, pp. 185–197, 1998.

[96] M. Salmerón, J. Ortega, C. G. Puntonet, and A. Prieto, "Improved RAN sequential prediction using orthogonal techniques," *Neurocomputing*, vol. 41, pp. 153–172, 2001.

[97] B. Todorović and M. Stanković, "Sequential growing and pruning of radial basis function network," in *Proceedings of the International Joint Conference on Neural Networks (IJCNN '01)*, vol. 3, pp. 1954–1959, Washington, DC, USA, 2001.

[98] G. B. Huang, P. Saratchandran, and N. Sundararajan, "An efficient sequential learning algorithm for growing and pruning RBF (GAP-RBF) networks," *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, vol. 34, no. 6, pp. 2284–2292, 2004.

[99] G. B. Huang, P. Saratchandran, and N. Sundararajan, "A generalized growing and pruning RBF (GGAP-RBF) neural network for function approximation," *IEEE Transactions on Neural Networks*, vol. 16, no. 1, pp. 57–67, 2005.

[100] M. Riedmiller and H. Braun, "Direct adaptive method for faster backpropagation learning: the RPROP algorithm," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 586–591, San Francisco, Calif, USA, April 1993.

[101] B. Hassibi, D. G. Stork, and G. J. Wolff, "Optimal brain surgeon and general network pruning," in *Proceedings of the IEEE International Conference on Neural Networks*, pp. 293–299, San Francisco, Calif, USA, April 1993.

[102] Y. Le Cun, J. S. Denker, and S. A. Solla, "Optimal brain damage," in *Advances in Neural Information Processing Systems*, D. S. Touretzky, Ed., vol. 2, pp. 598–605, Morgan Kauffmann, San Mateo, Calif, USA, 1990.

[103] N. Jankowski and V. Kadirkamanathan, "Statistical control of growing and pruning in RBF-like neural networks," in *Proceedings of the 3rd Conference Neural Networks and Their Applications*, pp. 663–670, Kule, Poland, 1997.

[104] A. Alexandridis, H. Sarimveis, and G. Bafas, "A new algorithm for online structure and parameter adaptation of RBF networks," *Neural Networks*, vol. 16, no. 7, pp. 1003–1017, 2003.

[105] J. Paetz, "Reducing the number of neurons in radial basis function networks with dynamic decay adjustment," *Neurocomputing*, vol. 62, no. 1–4, pp. 79–91, 2004.

[106] A. L. Oliveira, B. J. Melo, and S. R. Meira, "Improving constructive training of RBF networks through selective pruning and model selection," *Neurocomputing*, vol. 64, no. 1–4, pp. 537–541, 2005.

[107] E. Ricci and R. Perfetti, "Improved pruning strategy for radial basis function networks with dynamic decay adjustment," *Neurocomputing*, vol. 69, no. 13–15, pp. 1728–1732, 2006.

[108] S. Mc Loone and G. Irwin, "Improving neural network training solutions using regularisation," *Neurocomputing*, vol. 37, pp. 71–90, 2001.

[109] C. S. Leung and J. P. F. Sum, "A fault-tolerant regularizer for RBF networks," *IEEE Transactions on Neural Networks*, vol. 19, no. 3, pp. 493–507, 2008.

[110] J. P. F. Sum, C. S. Leung, and K. I. J. Ho, "On objective function, regularizer, and prediction error of a learning algorithm for dealing with multiplicative weight noise," *IEEE Transactions on Neural Networks*, vol. 20, no. 1, pp. 124–138, 2009.

[111] A. Leonardis and H. Bischof, "An efficient MDL-based construction of RBF networks," *Neural Networks*, vol. 11, no. 5, pp. 963–973, 1998.

[112] J. Rissanen, "Modeling by shortest data description," *Automatica*, vol. 14, no. 5, pp. 465–471, 1978.

[113] M. T. Musavi, W. Ahmed, K. H. Chan, K. B. Faris, and D. M. Hummels, "On the training of radial basis function classifiers," *Neural Networks*, vol. 5, no. 4, pp. 595–603, 1992.

[114] A. Ghodsi and D. Schuurmans, "Automatic basis selection techniques for RBF networks," *Neural Networks*, vol. 16, no. 5-6, pp. 809–816, 2003.

[115] G. Schwarz, "Estimating the dimension of a model," *The Annals of Statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[116] P. Niyogi and F. Girosi, "Generalization bounds for function approximation from scattered noisy data," *Advances in Computational Mathematics*, vol. 10, no. 1, pp. 51–80, 1999.

[117] P. Niyogi and F. Girosi, "On the relationship between generalization error, hypothesis complexity, and sample complexity for radial basis functions," *Neural Computation*, vol. 8, no. 4, pp. 819–842, 1996.

[118] A. R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *Institute of Electrical and Electronics Engineers*, vol. 39, no. 3, pp. 930–945, 1993.

[119] A. Krzyzak and T. Linder, "Radial basis function networks and complexity regularization in function learning," *IEEE Transactions on Neural Networks*, vol. 9, no. 2, pp. 247–256, 1998.

[120] Y. Li and J. M. Deng, "WAV—a weight adaptation algorithm for normalized radial basis function networks," in *Proceedings of the IEEE International Conference on Electronics, Circuits, and Systems*, vol. 2, pp. 117–120, Lisbon, Portugal, 1998.

[121] J. S. R. Jang and C. T. Sun, "Functional equivalence between radial basis function networks and fuzzy inference systems," *IEEE Transactions on Neural Networks*, vol. 4, no. 1, pp. 156–159, 1993.

[122] G. Bugmann, “Normalized Gaussian radial basis function networks,” *Neurocomputing*, vol. 20, no. 1–3, pp. 97–110, 1998.

[123] S. J. Nowlan, “Maximum likelihood competitive learning,” in *Advances in Neural Information Processing Systems*, D. S. Touretzky, Ed., vol. 2, pp. 574–582, Morgan Kauffmann, Boston, Mass, USA, 1990.

[124] M. Benaim, “On functional approximation with Normalized Gaussian units,” *Neural Computation*, vol. 6, no. 2, pp. 319–333, 1994.

[125] J. S. R. Jang, “ANFIS: adaptive-network-based fuzzy inference system,” *IEEE Transactions on Systems, Man and Cybernetics*, vol. 23, no. 3, pp. 665–685, 1993.

[126] F. Yang and M. Paindavoine, “Implementation of an RBF neural network on embedded systems: real-time face tracking and identity verification,” *IEEE Transactions on Neural Networks*, vol. 14, no. 5, pp. 1162–1175, 2003.

[127] G. Loreto and R. Garrido, “Stable neurovisual servoing for robot manipulators,” *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 953–965, 2006.

[128] S. Chen, A. Wolfgang, C. J. Harris, and L. Hanzo, “Symmetric RBF classifier for nonlinear detection in multiple-antenna-aided systems,” *IEEE Transactions on Neural Networks*, vol. 19, no. 5, pp. 737–745, 2008.

[129] D. Jianping, N. Sundararajan, and P. Saratchandran, “Communication channel equalization using complex-valued minimal radial basis function neural networks,” *IEEE Transactions on Neural Networks*, vol. 13, no. 3, pp. 687–697, 2002.

[130] S. Chen, S. McLaughlin, and B. Mulgrew, “Complex-valued radial basic function network, Part I: network architecture and learning algorithms,” *Signal Processing*, vol. 35, no. 1, pp. 19–31, 1994.

[131] C. S. Leung, T. T. Wong, P. M. Lam, and K. H. Choy, “An RBF-based compression method for image-based relighting,” *IEEE Transactions on Image Processing*, vol. 15, no. 4, pp. 1031–1041, 2006.

[132] S. Y. Cho and T. W. S. Chow, “Neural computation approach for developing a 3-D shape reconstruction model,” *IEEE Transactions on Neural Networks*, vol. 12, no. 5, pp. 1204–1214, 2001.

[133] Y. T. Tsai and Z. C. Shih, “All-frequency precomputed radiance transfer using spherical radial basis functions and clustered tensor approximation,” *ACM Transactions on Graphics*, vol. 25, no. 3, pp. 967–976, 2006.

[134] I. E. Lagaris, A. C. Likas, and D. G. Papageorgiou, “Neural-network methods for boundary value problems with irregular boundaries,” *IEEE Transactions on Neural Networks*, vol. 11, no. 5, pp. 1041–1049, 2000.

[135] R. L. Jenison and K. Fissell, “A spherical basis function neural network for modeling auditory space,” *Neural Computation*, vol. 8, no. 1, pp. 115–128, 1996.

[136] P. M. Lam, T. Y. Ho, C. S. Leung, and T. T. Wong, “All-frequency lighting with multiscale spherical radial basis functions,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 16, no. 1, pp. 43–56, 2010.

[137] T. Y. Ho, C. S. Leung, P. M. Lam, and T. T. Wong, “Efficient relighting of RBF-based illumination adjustable images,” *IEEE Transactions on Neural Networks*, vol. 20, no. 12, pp. 1987–1993, 2009.

[138] A. F. Sheta and K. De Jong, “Time-series forecasting using GA-tuned radial basis functions,” *Information Sciences*, vol. 133, no. 3-4, pp. 221–228, 2001.

[139] J. M. Vesin, “Amplitude-dependent autoregressive model based on a radial basis functions expansion,” in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP ’93)*, vol. 3, pp. 129–132, Minneapolis, Minn, USA, April 1993.

[140] M. R. Berthold, “Time delay radial basis function network for phoneme recognition,” in *Proceedings of the IEEE International Conference on Neural Networks*, vol. 7, pp. 4470–4472, Orlando, Fla, USA, June 1994.

[141] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, “Phoneme recognition using time-delay neural networks,” *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, pp. 328–339, 1989.

[142] S. A. Billings and X. Hong, “Dual-orthogonal radial basis function networks for nonlinear time series prediction,” *Neural Networks*, vol. 11, no. 3, pp. 479–493, 1998.

[143] S. Chen, S. A. Billings, and W. Luo, “Orthogonal least squares methods and their application to non-linear system identification,” *International Journal of Control*, vol. 50, no. 5, pp. 1873–1896, 1989.

[144] D. L. Yu, “A localized forgetting method for Gaussian RBFN model adaptation,” *Neural Processing Letters*, vol. 20, pp. 125–135, 2004.

[145] P. Frasconi, M. Cori, M. Maggini, and G. Soda, "Representation of finite state automata in recurrent radial basis function networks," *Machine Learning*, vol. 23, no. 1, pp. 5–32, 1996.

[146] B. Todorovic, M. Stankovic, and C. Moraga, "Extended Kalman filter trained recurrent radial basis function network in nonlinear system identification," in *Proceedings of the International Conference on Artificial Neural Networks (ICANN '95)*, J. R. Dorronsoro, Ed., pp. 819–824, Madrid, Spain, 2002.

[147] Z. Ryad, R. Daniel, and Z. Noureddine, "The RRBF dynamic representation of time in radial basis function network," in *Proceedings of the 8th IEEE International Conference Emerging Technologies and Factory Automation (ETFA '01)*, vol. 2, pp. 737–740, Juan-les-Pins Antibes, France, 2001.

[148] K. Y. Lee and S. Jung, "Extended complex RBF and its application to M-QAM in presence of co-channel interference," *Electronics Letters*, vol. 35, no. 1, pp. 17–19, 1999.

[149] I. Cha and S. A. Kassam, "Channel equalization using adaptive complex radial basis function networks," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 1, pp. 122–131, 1995.

[150] M. B. Li, G. B. Huang, P. Saratchandran, and N. Sundararajan, "Fully complex extreme learning machine," *Neurocomputing*, vol. 68, no. 1–4, pp. 306–314, 2005.

[151] S. Chen, X. Hong, C. J. Harris, and L. Hanzo, "Fully complex-valued radial basis function networks: orthogonal least squares regression and classification," *Neurocomputing*, vol. 71, no. 16–18, pp. 3421–3433, 2008.

[152] A. R. Webb and D. Lowe, "The optimised internal representation of multilayer classifier networks performs nonlinear discriminant analysis," *Neural Networks*, vol. 3, no. 4, pp. 367–375, 1990.

[153] B. Widrow and M. A. Lehr, "30 years of adaptive neural networks: perceptron, madaline, and backpropagation," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1415–1442, 1990.

[154] K. L. Du, X. Huang, M. Wang, B. Zhang, and J. Hu, "Robot impedance learning of the peg-in-hole dynamic assembly process," *International Journal of Robotics and Automation*, vol. 15, no. 3, pp. 107–118, 2000.

[155] B. Kosko, Ed., *Neural Networks for Signal Processing*, Prentice Hall, Englewood Cliffs, NJ, USA, 1992.

[156] S. Fabri and V. Kadirkamanathan, "Dynamic structure neural networks for stable adaptive control of nonlinear systems," *IEEE Transactions on Neural Networks*, vol. 7, no. 5, pp. 1151–1167, 1996.

[157] M. Lehtokangas and J. Saarinen, "Centroid based multilayer perceptron networks," *Neural Processing Letters*, vol. 7, no. 2, pp. 101–106, 1998.

[158] G. Dorffner, "Unified framework for MLPs and RBFNs: introducing conic section function networks," *Cybernetics and Systemsl*, vol. 25, no. 4, pp. 511–554, 1994.

[159] S. Cohen and N. Intrator, "A hybrid projection-based and radial basis function architecture: initial values and global optimisation," *Pattern Analysis and Applications*, vol. 5, no. 2, pp. 113–120, 2002.

[160] D. Wedge, D. Ingram, D. McLean, C. Mingham, and Z. Bandar, "On global-local artificial neural networks for function approximation," *IEEE Transactions on Neural Networks*, vol. 17, no. 4, pp. 942–952, 2006.

[161] S. B. Holden and P. J. W. Rayner, "Generalization and PAC learning: some new results for the class of generalized single-layer networks," *IEEE Transactions on Neural Networks*, vol. 6, no. 2, pp. 368–380, 1995.

[162] K. M. Adeney and M. J. Korenberg, "Iterative fast orthogonal search algorithm for MDL-based training of generalized single-layer networks," *Neural Networks*, vol. 13, no. 7, pp. 787–799, 2000.

[163] Q. Zhang and A. Benveniste, "Wavelet networks," *IEEE Transactions on Neural Networks*, vol. 3, no. 6, pp. 889–898, 1992.

[164] J. Zhang, G. G. Walter, Y. Miao, and W. N. W. Lee, "Wavelet neural networks for function learning," *IEEE Transactions on Signal Processing*, vol. 43, no. 6, pp. 1485–1497, 1995.

[165] T.-H. Li, "Multiscale representation and analysis of spherical data by spherical wavelets," *SIAM Journal on Scientific Computing*, vol. 21, no. 3, pp. 924–953, 1999.

[166] P. J. W. Rayner and M. R. Lynch, "New connectionist model based on a non-linear adaptive filter," in *Proceedings of the International Conference on Acoustics, Speech, and Signal Processing (ICASSP '89)*, pp. 1191–1194, Glasgow, UK, May 1989.

[167] P. J. Huber, *Robust Statistics*, Wiley Series in Probability and Mathematical Statistic, John Wiley & Sons, New York, NY, USA, 1981.

[168] A. V. D. Sanchez, "Robustization of a learning method for RBF networks," *Neurocomputing*, vol. 9, no. 1, pp. 85–94, 1995.

[169] C. C. Chuang, J. T. Jeng, and P. T. Lin, "Annealing robust radial basis function networks for function approximation with outliers," *Neurocomputing*, vol. 56, no. 1–4, pp. 123–139, 2004.

[170] S. S. Watkins and P. M. Chau, "A radial basis function neuroncomputer implemented with analog VLSI circuits," in *Proceedings of the International Joint Conference on Neural Networks*, vol. 2, pp. 607–612, Baltimore, Md, USA, 1992.

[171] S. Churcher, A. F. Murray, and H. M. Reekie, "Programmable analogue VLSI for radial basis function networks," *Electronics Letters*, vol. 29, no. 18, pp. 1603–1605, 1993.

[172] D. J. Mayes, A. F. Murray, and H. M. Reekie, "Pulsed VLSI for RBF neural networks," in *Proceedings of the 5th IEEE International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pp. 177–184, Lausanne, Switzerland, 1996.

[173] I. C. Cevikbas, A. S. Ogrenci, G. Dundar, and S. Balkir, "VLSI implementation of GRBF (Gaussian Radial Basis Function) networks," in *Proceedings of the IEEE International Symposium on Circuits and Systems (IEEE ISCAS '00)*, vol. 3, pp. 646–649, Geneva, Switzerland, 2000.

[174] P. Maffezzoni and P. Gubian, "VLSI design of radial functions hardware generator for neural computations," in *Proceedings of the 4th IEEE International Conference on Microelectronics for Neural Networks and Fuzzy Systems*, pp. 252–259, Turin, Italy, 1994.

[175] M. Aberbour and H. Mehrez, "Architecture and design methodology of the RBF-DDA neural network," in *Proceedings of the IEEE International Symposium on Circuits and Systems (IEEE ISCAS '98)*, vol. 3, pp. 199–202, June 1998.