Chapter 3. Radial Basis Function Networks

3.1 Introduction

A radial basis function network is a neural network approached by viewing the design as a curve-fitting (approximation) problem in a high dimensional space. Learning is equivalent to finding a multidimensional function that provides a best fit to the training data, with the criterion for "best fit" being measured in some statistical sense [5]. Correspondingly, *regularization* is equivalent to the use of this multidimensional surface to interpolate the test data. This viewpoint is the real motivation behind the RBF method in the sense that it draws upon research work on traditional strict interpolations in a multidimensional space. In a neural network, the hidden units form a set of "functions" that compose a random "basis" for the input patterns (vectors). These functions are called *radial basis functions*[5].

Radial basis functions were first introduced by Powell to solve the real multivariate interpolation problem [6]. This problem is currently one of the principal fields of research in numerical analysis. In the field of neural networks, radial basis functions were first used by Broomhead and Lowe [7]. Other major contributions to the theory, design, and applications of RBFNs can be found in papers by Moody and Darken, Renals, and Poggio and Girosi [8, 9, 10]. The paper by Poggio and Girosi explains the use of *regularization theory* applied to this class of neural networks as a method for improved generalization to new data [10].

The design of a RBFN in its most basic form consists of three separate layers. The input layer is the set of source nodes (sensory units). The second layer is a hidden layer of high dimension. The output layer gives the response of the network to the activation patterns applied to the input layer. The transformation from the input space to the hidden-unit space is *nonlinear*. On the other hand, the transformation from the hidden space to the output space is linear [5]. A mathematical justification of this can be found in the paper by Cover [41]. Cover states that a pattern classification problem cast in a nonlinear high-dimensional space is more likely to be linearly *separable* than in a low-dimensional space. This statement is called *Cover's Theorem* on separability of patterns. It is also the reason for making the dimension of the hidden-unit space high in an RBFN.

This chapter gives a review of separability of patterns, the interpolation problem, supervised learning as an ill-posed hypersurface reconstruction problem, regularization theory, regularization networks, generalized radial basis functions, the XOR problem and learning strategies for radial basis function networks. Sections 3.2-3.6 provide a satisfactory background for regularization networks. Generalized radial basis function networks are introduced in terms of regularization networks in Section 3.7. Section 3.8 introduces an example of nonlinearly separable classification problem (the XOR problem). Finally Section 3.9 ends Chapter 3 by examining the learning strategies for radial basis function networks.

3.2 Cover's Theorem on the Separability of Patterns

Before we talk about the radial basis function networks, we need to introduce the term "separability of patterns". The *Cover's Theorem* gives a detailed description of the separability of patterns. This theorem explains how a radial basis function network can perform a complex pattern classification task. Considering the Cover's Theorem, Haykin declares that a radial basis function network performs a complex pattern-classification task by transforming the problem into a high-dimensional space in a nonlinear manner [5]. He gives a detailed definition of Cover's Theorem as follows:

> Consider a family of surfaces, each of which naturally divides an input space into two regions. Let X denote a set of N patterns (points) $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$, each of which is assigned to one of two classes X^+ and X^-. This *dichotomy* (binary partition) of the points is said to be separable with respect to the family of surfaces if there exists a surface in the family that separates the points in the class X^+ from those in the class X^-. For each pattern $\mathbf{x} \in X$, define a vector made up of a set of real-valued functions $\{ \varphi_i(\mathbf{x}) \mid i = 1, 2,...,M\}$, as shown by
>
> $$\varphi(\mathbf{x}) = [\varphi_1(\mathbf{x}), \varphi_2(\mathbf{x}), \ldots, \varphi_M(\mathbf{x})]^T \quad (3.1)$$
>
> Suppose that the pattern $\mathbf{x}$ is a vector in a p-dimensional input space. The vector $\varphi(\mathbf{x})$ then maps points in p-dimensional input space into corresponding points in a new space of dimension M. We refer to $\varphi_i(\mathbf{x})$ as a *hidden function*, because it plays a role similar to that of a hidden unit in a feed forward neural network.
>
> A dichotomy $\{X^+, X^-\}$ of X is said to be φ-*separable* if there exists an m-dimensional vector $\mathbf{w}$ such that we may write [41]
>
> $$\mathbf{w}^T\varphi(\mathbf{x}) \geq 0, \quad \mathbf{x} \in X^+$$
>
> and
>
> $$\mathbf{w}^T\varphi(\mathbf{x}) < 0, \quad \mathbf{x} \in X^- \quad (3.2)$$
>
> The hyperplane defined by the equation
>
> $$\mathbf{w}^T\varphi(\mathbf{x}) = 0 \quad (3.3)$$
>
> describes the separating surface in the φ space. The inverse image of this hyperplane, that is,
>
> $$\{\mathbf{x}: \mathbf{w}^T\varphi(\mathbf{x}) = 0\} \quad (3.4)$$
>
> defines the separating surface in the input space [5].

After giving the definition of the Cover's theorem on separability of patterns, Haykin gives a mathematical explanation for the class of mapping explained above [5]. The separating surfaces corresponding to such mappings are referred to as *rth-order rational varieties*. A rational variety of order r in a space of dimension p is defined by the rth-degree homogenous equation in the coordinates of the input vector $\mathbf{x}$, as illustrated by

$$\sum_{0 \leq i_1 \leq i_2 \leq \cdots \leq i_r \leq p} a_{i_1 i_2 \cdots i_r} x_{i_1} x_{i_2} \cdots x_{i_r} = 0 \quad (3.5)$$

where x_i is the ith element of the input vector $\mathbf{x}$, and x_0 is set equal to unity in order to express the equation in a homogenous form. Some examples of this type of separating

surfaces are *hyperplanes* (first-order rational varieties), *quadrics* (second-order rational varieties), and *hyperspheres* (quadrics with certain linear constraints on the coefficients). Figure 3.1. illustrates the examples for a configuration of five points in two dimensions.

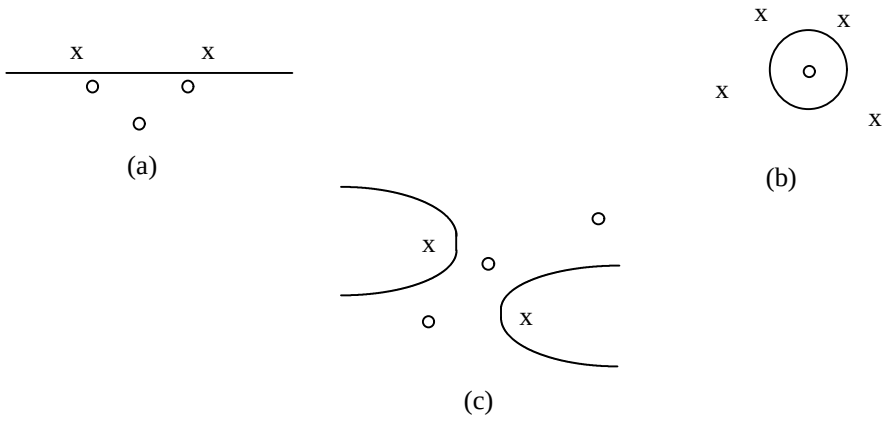

Figure 3.1 Three examples of φ-separable dichotomies of different sets of five points in two dimensions: (a) linearly separable dichotomy; (b) spherically separable dichotomy; (c) quadratically separable dichotomy.

Polynomial separability, as defined here, can be considered as a natural generalization of linear separability. Haykin provides an important point, which states that given a set of patterns **x** in an input space of random dimension *p*, a non-linear mapping (**x**) of high enough dimension *M* can be found so that linear separability in the space is obtained [5]. The next section talks about the interpolation problem which has great importance in solving the nonlinearly separable pattern classification problem.

3.3 Interpolation Problem

This section talks about the interpolation problem that allows us to solve the nonlinearly separable pattern classification problem. The interpolation plays the final role in solving the problem since it finds the linear weight vector of the network, defined in the next paragraph.

In solving a nonlinearly separable pattern classification problem, there is generally a practical benefit in mapping the input space into a new space of sufficiently high dimension. This is an important point that comes forth from Cover's Theorem on the separability of patterns. Let us consider a feed forward network with an input layer, a single hidden layer, and an output layer having a single unit. The network can be designed to perform a nonlinear mapping from the input space to the hidden space, and a linear mapping from the hidden space to the output space.

The network represents a map from *p*-dimensional input space to the single dimensional output space, expressed as

$$s: \mathbb{R}^p \rightarrow \mathbb{R}^1 \tag{3.6}$$

The theory of *multivariable interpolation* in high-dimensional space has a long history starting with Davis [59]. The interpolation problem, in its *strict* sense can be stated as follows:

> Given a set of N different points $\{\mathbf{x}_i \in \mathbb{R}^2 \mid i = 1, 2,...., N\}$ and a corresponding set of N real numbers $\{d_i \in \mathbb{R}^1 \mid i = 1,2,...., N\}$, find a function F: $\mathbb{R}^N \rightarrow \mathbb{R}^1$ that satisfies the interpolation condition [5]:
>
> $$F(\mathbf{x}_i) = d_i \qquad i = 1,2,....,N \tag{3.7}$$

The interpolating surface (i.e. function F) has to pass through *all* the training data points [5]. The radial basis function technique consists of choosing a function that has the following form given by Powell [6].

$$F(\mathbf{x}) = \sum_{i=1}^{N} w_i \varphi(\|\mathbf{x} - \mathbf{x}_i\|) \tag{3.8}$$

where $\{\varphi(\|\mathbf{x} - \mathbf{x}_i\|) \mid i = 1,2,\ldots N)\}$ is a set of N random (usually nonlinear) functions, known as *radial basis functions*, and $\|\ .\ \|$ represents a *norm* that is generally Euclidean. The known data points $\mathbf{x}_i \in \mathbb{R}^p$, $i = 1,2,\ldots N$ are the *centers* of radial basis functions [6].

If the interpolation conditions equation (3.7) is inserted in (3.8), the following set of simultaneous linear equations can be obtained for the unknown coefficients (weights) of the expansion $\{w_i\}$:

$$\begin{bmatrix} \varphi_{11} & \varphi_{12} & \cdots\cdots & \varphi_{1N} \\ \varphi_{21} & \varphi_{22} & \cdots\cdots & \varphi_{2N} \\ \vdots & \vdots & & \vdots \\ \varphi_{N1} & \varphi_{N2} & \cdots\cdots & \varphi_{NN} \end{bmatrix} \begin{bmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{bmatrix} = \begin{bmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{bmatrix} \tag{3.9}$$

where

$$\varphi_{ji} = \varphi(\|\mathbf{x}_j - \mathbf{x}_i\|) \qquad j,i = 1,2,\ldots, N \tag{3.10}$$

Let

$$\mathbf{d} = [d_1, d_2,....., d_N] \tag{3.11}$$

$$\mathbf{w} = [w_q, w_2,....,w_N] \tag{3.12}$$

The vectors $\mathbf{d}$ and $\mathbf{w}$ represent the *desired response vector* and *linear weight vector*, respectively. Let $\mathbf{\Phi}$ denote an N-by-N matrix with elements φ_{ji}:

$$\mathbf{\Phi} = \{\varphi_{ji} \mid j,i = 1,\ 2,....,N\} \tag{3.13}$$

The matrix $\mathbf{\Phi}$ is called the interpolation matrix. Equation (3.9) can be written in the compact form:

$$\mathbf{\Phi w = x} \tag{3.14}$$

Light gives a remarkable property for a class of radial basis functions which obtains a positive definite interpolation matrix $\mathbf{\Phi}$ [60]. Because a positive definite matrix has

always an inverse this specific class of radial basis functions will always solve the interpolation problem. This remarkable property can be expressed as follows:

> Let $\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N$ be distinct in $\mathbb{R}^p$. Then the N-by-N interpolation matrix Φ whose jith element is $\varphi_{ji} = \varphi(\|\mathbf{x}_j - \mathbf{x}_i\|)$ is positive definite.

The common examples of this specific class of radial basis functions are given as follows:

1. *Inverse Multiquadrics*

$$\varphi(r) = \frac{1}{(r^2 + c^2)^{1/2}} \text{ for some } c > 0, \text{ and } r \geq 0 \tag{3.15}$$

2. *Gaussion Functions*

$$\varphi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right) \text{ for } \sigma > 0, \text{ and } r \geq 0 \tag{3.16}$$

Powell declares that theoretical investigation and practical results show that the type of nonlinearity $\varphi(\cdot)$ is not vital to the performance of RBFNs [6]. Considering Light's Theorem, it is noted that if the data points are all distinct, the interpolation matrix Φ is positive definite, and the weight vector $\mathbf{w}$ can be formed as follows:

$$\mathbf{w} = \Phi^{-1}\mathbf{d} \tag{3.17}$$

Where Φ^{-1} is the inverse of the interpolation matrix Φ. Even though in theory a solution to the strict interpolation problem exists, in practice equation (3.14) cannot be solved when the matrix Φ is arbitrarily close to singular. Regularization theory can solve this problem by perturbating the matrix Φ to $\Phi + \lambda\mathbf{I}$, as described in Section 3.5. This problem leads us to examine the solving of an ill-posed hypersurface reconstruction problem because this inverse problem can become an ill-posed problem. The next section talks about how an ill-posed problem can be solved.

3.4 Supervised Learning as an Ill-Posed Hypersurface Reconstruction Problem

The strict interpolation procedure as defined above may not be a good method for the training of RBF networks for certain classes of tasks because of poor generalization to new data for the following reason. When the number of data points in the training set is much larger than the number of degrees of freedom of the underlying physical process, and the network cannot have radial basis functions greater than the number of data points, the problem is over determined. Therefore, the network may end up fitting misleading variations because of "idiosyncrasies" or noise in the input data, thereby resulting in a degraded generalization performance [7].

The design of a neural network trained to have an output pattern when presented with an input pattern is equivalent to learning a hypersurface (i.e. multidimensional mapping) that defines the output in terms of the input. That is, learning is considered as a hypersurface reconstruction problem, given a set of data that may be sparse. Therefore,

the hypersurface reconstruction or approximation problem belongs to a "generic" class of problems called *inverse* problems [5].

An inverse problem can be well-posed or ill-posed. The term "well-posed" has been used in applied mathematics since the time of Hadamard in the early 1900s. Haykin gives an explanation about an inverse problem as follows:

> Assume that we have a domain X and a range Y taken to be metric spaces, and that are related by a fixed but unknown mapping F. The problem of reconstructing the mapping F is said to be *well posed* if three conditions are satisfied
>
> 1. *Existence.* For every input vector $\mathbf{x} \in X$, there exist an output $y = F(\mathbf{x})$, where $y \in Y$,
> 2. *Uniqueness.* For any pair of input vectors $\mathbf{x}, \mathbf{t} \in X$. We have $F(\mathbf{x}) = F(\mathbf{t})$ if, and only if, $\mathbf{x} = \mathbf{t}$.
> 3. *Continuity.* The mapping is continuous, that is, for any $\varepsilon > 0$, there exists $\delta = \delta(\varepsilon)$ such that the condition $\rho_X(\mathbf{x}, \mathbf{t}) < \delta$ implies that $\rho_Y(F(\mathbf{x}), F(\mathbf{t})) < \varepsilon$, where $\rho(\cdot,\cdot)$ is the symbol for distance between the two arguments in their respective spaces.
>
> If these conditions are not satisfied, the inverse problem is said to be ill-posed [5].

Haykin supports reasons why learning is an ill-posed inverse problem when learning is viewed as a hypersurface reconstruction problem. First, there is insufficient information in the training data in order to reconstruct the input-output mapping uniquely, thus the uniqueness criterion is not satisfied. Second, the existence of noise and imprecision in the input data adds uncertainty to the reconstructed input-output mapping. Particularly, if the noise level in the input is too high, the neural network may produce an output outside of the range Y for a specified input $\mathbf{x}$ in the domain X; hence the continuity criterion is not satisfied [5].

Poggio and Girosi state that some form of prior information about the input-output mapping is necessary to make the learning problem well-posed so that generalization to the new data can be accomplished [10]. In other words, the process responsible for generation of input-output examples used to train a neural network must show *redundancy* in an information-theoretic sense to establish the prior information about the input-output mapping. This necessity is, indeed, satisfied by the physical processes such as speech, pictures, radar, and sonar signals in practice. These processes are all redundant by their nature. Moreover, the *generator* of the data is generally *smooth*. As long as the data generation is smooth, small changes in the input can cause large changes in the output and can still be approximated successfully. Haykin notes that the smoothness of data generation is a fundamental form of functional redundancy [5].

As was said in Section 3.3, the interpolation problem can not be solved when the matrix Φ is arbitrarily close to singular or ill-posed. The next section gives a solution to this problem using regularization theory.

3.5 Regularization Theory

Regularization Theory was first introduced by Tikhonov in 1963 [63]. The fundamental idea of regularization is to *stabilize* the solution in terms of some auxiliary nonnegative functional that embeds prior information, e.g., smoothness constraints on the input-output mapping, and make an ill-posed problem into a well-posed one [10]. Let the set of input-output data available for approximation be described by

$$\text{Input signal :} \quad \mathbf{x}_i \in \mathbb{R}^p, \quad i = 1, 2, ..., N \tag{3.18}$$

$$\text{Desired signal:} \quad d_i \in \mathbb{R}^1, \quad i = 1, 2, ..., N \tag{3.19}$$

The dimensionality of the output is chosen as one. This choice does not limit the general applicability of the regularization theory in any way. The approximation function is denoted by $F(\mathbf{x})$. The weight factor $\mathbf{w}$ of the network is omitted from the argument of the function F for convenience of presentation. According to Tikhonov's regularization theory [63], the function F is obtained by minimizing a *cost functional* $\xi(F)$ that maps functions to the real line. Haykin expresses the cost functional using two terms of regularization as follows:

$$\xi(F) = \xi_s(F) + \lambda\xi_c(F) \tag{3.23}$$

where $\xi_s(F)$ is *standard error term* that measures the standard error (distance) between the desired response d_i and the actual response y_i for training samples $i = 1, 2, .., N$. The term $\xi_c(F)$ is *regularization term* that depends on the geometric properties of the approximation function $F(\mathbf{x})$. The symbol λ is a positive real number called *regularization parameter*. The main aim of the regularization is to minimize the cost functional $\xi(F)$. The cost functional can be written interms of the desired response d_i, the actual response y_i, and the regularization parameter λ as follows:

$$\xi(F) = \frac{1}{2}\sum_{i=1}^{N}[d_i - F(\mathbf{x}_i)]^2 + \frac{1}{2}\lambda\,\|\mathbf{P}F\|^2 \tag{3.24}$$

where $\mathbf{P}$ is a linear (pseudo) differantial operator that contains the prior information about the form of the solution. Haykin refers to $\mathbf{P}$ as a *stabilizer* in the sense that stabilizes the solution F making it smooth and therefore continuous. A detailed explanation of the operator $\mathbf{P}$ is given in [5].

The regularization parameter, λ, is considered as indicator of the sufficiency of the given data set as examples that specify the solution $F(\mathbf{x})$. If $\lambda \to 0$, the problem is unconstrained and the solution $F(\mathbf{x})$ can be completely determined from the examples. On the other hand, if, $\lambda \to \infty$, the *a priori* smoothness constraint is sufficient to specify the solution $F(\mathbf{x})$; that is, the examples are unreliable. In practice, λ is assigned a value somewhere between 0 and ∞, so that both the sample data and the *a priori* information contribute to the solution $F(\mathbf{x})$. Therefore, the regularizing term $\xi_c(F)$ represents a *model complexity penalty function*, the influence of which on the final solution is controlled by the regularization parameter λ [5]. This section gave broad explanation of the cost functional $\xi(F)$ and its parameters. The next section gives detailed mathematical review of the solution of the regularization problem minimizing the cost functional $\xi(F)$.

3.5.1 Solution to the Regularization Problem

The *principle of regularization* is to find the function $F(\mathbf{x})$ that minimizes the cost functional $\xi(F)$, defined by equation (3.23). To manage the minimization of the cost functional $\xi(F)$, an evaluation of the differential of $\xi(F)$ is necessary. The *Frechet differential* [65] can be employed to do the minimization. The *Frechet differential* has the following form:

$$d\ (F,h) = \left[\frac{d}{d\beta}\xi(F+\beta h)\right]_{\beta=0} \tag{3.25}$$

where $h(\mathbf{x})$ is a fixed function of the vector $\mathbf{x}$, and β is a muti index. A *multi-index* $\beta=(\beta_1, \beta_2, ..., \beta_n)$ of *order* $|\beta| = \sum_{i=1}^{n}\beta_i$ is a set of whole numbers used to abrreviate the following notations [61]:

1. $\mathbf{x} = \mathbf{x}_1^{\beta_1}\mathbf{x}_2^{\beta_2}\cdots\mathbf{x}_n^{\beta_n}$ for $\mathbf{x} \in \Re^n$

2. $\mathbf{x} = \dfrac{\partial^{|\beta|} f}{\partial\mathbf{x}_1^{\beta_1}\partial\mathbf{x}_2^{\beta_2}\cdots\partial\mathbf{x}_n^{\beta_n}}$ for f: $\Re^n \rightarrow \Re^1$

A necessary condition for the function $F(\mathbf{x})$ to be a relative extremum of the functional $\xi(F)$ is that the Frechet differential $d\xi(F,h)$ be zero at $F(\mathbf{x})$ for all $h \in H$, as expressed by

$$d\xi(F,h) = d\xi_s(F,h) + \lambda d\xi_c(F,h) = 0 \tag{3.26}$$

where $d\xi_s(F,h)$ and $d\xi_c(F,h)$ are the Frechet differentials of the functionals $\xi_s(F)$ and $\xi_c(F)$, respectively. After the evaluation of the Frechet differential, the standard error term $\xi_s(F,h)$ is expressed as follows [5]:

$$\begin{aligned} d\ _s(F,h) &= \left[\frac{d}{d\beta}\xi_s(F+\beta h)\right]_{\beta=0} \\ &= -\sum_{i=1}^{N}[d_i - F(\mathbf{x}_i)]h(\mathbf{x}_i) \end{aligned} \tag{3.27}$$

Similarly, the regularizing term $\xi_c(F)$ can be expressed by the following equation:

$$\begin{aligned} d\xi_c(F,h) &= \frac{d}{d\beta}\xi_c(F+\beta h)|_{\beta=0} \\ &= (\mathbf{P}h, \mathbf{P}F)_H \end{aligned} \tag{3.28}$$

where H represents *Hilbert* space and the symbol $(\cdot,\cdot)_H$ represents the inner product in H space. Hilbert space is a normed vector space of functions. These functions are rapidly decreasing, infinitely continuously differentiable functions. Further details about Hilbert space can be found in [5]. Considering the definition of an *adjoint* differential operator, equation (3.28) can be written as:

$$d\xi_c(F,h) = (h, \mathbf{P}^*\mathbf{P}F)_H \tag{3.29}$$

where $\mathbf{P}^*$ is the adjoint of the differential operator $\mathbf{P}$.

Substituting the Frechet differentials of equation (3.27), and (3.29), in the equation (3.26), Haykin states that the Frechet differential $d\xi(F,h)$ is zero for every $h(\mathbf{x})$ in H space if and only if the following condition is satisfied:

$$\mathbf{P^*P}F - \frac{1}{\lambda}\sum_{i=1}^{N}(d_i - F)\delta_{\mathbf{x}_i} = 0$$

or, equivalently,

$$\mathbf{P^*P}F(\mathbf{x}) = \frac{1}{\lambda}\sum_{i=1}^{N}[d_i - F(\mathbf{x}_i)]\delta(\mathbf{x} - \mathbf{x}_i) \tag{3.30}$$

where $\delta(\mathbf{x} - \mathbf{x}_i)$ or $(\delta_{\mathbf{x}_i})$ is a delta function located at $\mathbf{x} = \mathbf{x}_i$.

Poggio and Grosi refer to equation (3.30) as the *Euler-Lagrange Equation* for the cost functional $\xi(F)$ expressed in equation (3.24) [10]. They also declare that the equation (3.30) symbolizes a partial pseudodifferential equation in F. The solution of this equation can be obtained by applying the integral transformation of the right hand side of the equation with a kernel given by the *influence function* or *Green's Function* for the self-adjoint differential operator $\mathbf{P}^*\mathbf{P}$. The role of the Green's function in a linear differential equation is the same as the role that an inverse matrix plays in a matrix equation.

Let $G(\mathbf{x};\mathbf{x}_i)$ be a Green's function centered at $\mathbf{x}_i$. A Green's function $G(\mathbf{x};\mathbf{x}_i)$ is any function that satisfies the partial differential equation

$$\mathbf{P^*P}G(\mathbf{x};\mathbf{x}_i) = 0$$

everywhere other than at the point $\mathbf{x} = \mathbf{x}_i$, where the Green's function has a singularity. Haykin gives the solution $F(\mathbf{x})$ for the differential equation (3.30) after some mathematical steps resulting with the following equation:

$$F(\mathbf{x}) = \frac{1}{\lambda}\sum_{i=1}^{N}[d_i - F(\mathbf{x}_i)]G(\mathbf{x};\mathbf{x}_i) \tag{3.31}$$

Equation (3.31) shows that the minimizing solution $F(\mathbf{x})$ to the regularization problem is a linear superposition of N Green's functions. In this equation, the $\mathbf{x}_i$ symbolize the *centers of the expansion*, and the weights $\left[d_i - F(\mathbf{x}_i)\right]/\lambda\right]$ symbolize the *coefficients of the expansion*. Haykin declares that the solution of the regularization problem lies in an N-dimensional subspace of the space of smooth functions, and the set of Green's functions $G(\mathbf{x};\mathbf{x}_i)$ centered at $\mathbf{x}_i$, i =1, 2,…, N, builds a basis for this subspace [5].

Since we have the solution $F(\mathbf{x})$ to the regularization problem, our next step is to determine the unknown coefficients in the equation (3.31). Let us denote:

$$w_i = \frac{1}{\lambda}[d_i - F(\mathbf{x}_i)], \qquad i = 1, 2, ..., N \tag{3.32}$$

The minimizing solution can be symbolized by the following equation:

$$F(\mathbf{x}) = \sum_{i=1}^{\mathbf{N}}\mathbf{w}_i G(\mathbf{x};\mathbf{x}_i) \tag{3.33}$$

After evaluation of the equation (3.33) at $\mathbf{x}_j$, j =1, 2,.., N, the equation can be expanded to

$$F(\mathbf{x}_j) = \sum_{i=1}^{\mathbf{N}}\mathbf{w}_i G(\mathbf{x}_j;\mathbf{x}_i), \qquad j = 1, 2,...., N \tag{3.34}$$

The definitions needed to be introduced can be given as follows;

$$\mathbf{F} = [F(\mathbf{x}_1),\ F(\mathbf{x}_2),\cdots, F(\mathbf{x}_N)]^T \tag{3.35}$$

$$\mathbf{d} = [d_1,\ d_2,\ .....,\ d_N] \tag{3.36}$$

and,

$$\mathbf{G} = \begin{bmatrix} G(\mathbf{x}_1;\mathbf{x}_1) & G(\mathbf{x}_1;\mathbf{x}_2)\cdots & G(\mathbf{x}_1;\mathbf{x}_N) \\ G(\mathbf{x}_2;\mathbf{x}_1) & G(\mathbf{x}_2;\mathbf{x}_2)\cdots & G(\mathbf{x}_2;\mathbf{x}_N) \\ \vdots & \vdots & \vdots \\ G(\mathbf{x}_N;\mathbf{x}_1) & G(\mathbf{x}_N;\mathbf{x}_2)\cdots & G(\mathbf{x}_N;\mathbf{x}_N) \end{bmatrix} \tag{3.37}$$

$$\mathbf{w} = [w_1,\ w_2,\ ....,\ w_N] \tag{3.38}$$

Now, the equation (3.32) and equation (3.34) can be rewritten in matrix form as follows, respectively:

$$\mathbf{w} = \frac{1}{\lambda}(\mathbf{d} - \mathbf{F}) \tag{3.39}$$

$$\mathbf{F} = \mathbf{G}\mathbf{w} \tag{3.40}$$

When **F** is eliminated between equations (3.39) and (3.40), the following equation can be obtained:

$$(\mathbf{G} + \lambda\mathbf{I})\mathbf{w} = \mathbf{d} \tag{3.41}$$

where **I** is the *N*-by-*N* identity matrix. The matrix **G** is called the *Green's Matrix*. Since the combined operator **P*P** in equation (3.30) is self-adjoint, the associated Green's function $\mathbf{G}(\mathbf{x};\mathbf{x}_i)$ is a symmetric function, as shown by Courant and Hilbert [66].

$$G(\mathbf{x}_i;\mathbf{x}_j) = G(\mathbf{x}_j;\mathbf{x}_i) \qquad \text{for all } i \text{ and } j \tag{3.42}$$

Similarly, the Green's matrix**G** defined in equation (3.37) is a symmetric matrix;

$$\mathbf{G}^T = \mathbf{G} \tag{3.43}$$

where the subscript *T* stands for matrix transposition. After revisiting Light's theorem, defined in Section 3.3 in the context of the interpolation matrix **Φ**, Haykin notes that Green's matrix, **G** has a role in regularization theory similar to the role that of **Φ** in RBF interpolation theory [5]. Both **G** and **Φ** are *N*-by-*N* symmetric matrices. Haykin also states that the matrix **G** is positive definite for certain classes of Green's functions if the data points $\mathbf{x}_1, \mathbf{x}_2,\ldots,\mathbf{x}_N$ are distinct [5]. Multiquadrics and Gaussian functions are the classes of Green's functions covered by Light's theorem. In practice, λ can be chosen sufficiently large enough to suffice so that (**G** + λ**I**) is positive definite, and thus, invertible. Poggio and Girosi give a unique solution of the linear system of equations (3.41) as follows:

$$\mathbf{w} = (\mathbf{G} + \lambda\mathbf{I})^{-1}\mathbf{d} \tag{3.44}$$

Haykin concludes that the solution to the regularization problem is expressed by the equation

$$F(\mathbf{x}) = \sum_{i=1}^{N} w_i G(\mathbf{x};\mathbf{x}_i) \tag{3.45}$$

where $G(\mathbf{x}_i;\mathbf{x}_j)$ is the Green's function for the self adjoint differential operator **P*P**, and w_i is the *i*th element weight vector **w** [5]. He also states that if the Green's functions in equation (3.45) are radial basis functions, the solution can be rewritten as follows:

$$F(\mathbf{x}) = \sum_{i=1}^{N} w_i G(\|\mathbf{x} - \mathbf{x}_i\|) \tag{3.46}$$

There is one more step left to reach the final solution of the regularization problem. We need to define the radial basis function in equation (3.46). The following functions can be employed in equation (3.46):

$$G(\mathbf{x};\mathbf{x}_i) = \exp\left(-\frac{1}{2\sigma_i^2}\|\mathbf{x} - \mathbf{x}_i\|^2\right) \tag{3.47}$$

where $G(\mathbf{x}; \mathbf{x}_i)$ is a *multivariate Gaussian function* characterized by a *mean vector* $\mathbf{x}_i$ and common *variance* σ_i^2, except for a scaling factor that can be put in the weight w_i. Now we can present our final solution to the regularization problem as follows:

$$F(\mathbf{x}) = \sum_{i=1}^{N} w_i \exp\left(-\frac{1}{2\sigma_i^2}\|\mathbf{x} - \mathbf{x}_i\|^2\right) \tag{3.48}$$

which contains a linear superposition of multivariate Gaussian basis functions with centers $\mathbf{x}_i$ (located at the data points) and widths σ_i [5]. In section 3.5, we discussed regularization theory and we declared the solution of the regularization problem in terms of radial basis functions. Now, we need to define our network structure that supports the solution we found in this section. The next section talks about regularization networks that are built by the radial basis functions defined above.

3.6 Regularization Networks

Figure 3.2 shows the network structure established considering the expansion of the approximation function $F(\mathbf{x})$ given in equation (3.45) in terms of the Green's function $G(\mathbf{x}; \mathbf{x}_i)$ centered at $\mathbf{x}_i$. This network is called a *regularization network*, introduced by Poggio and Girosi because it uses the solution to the regularization problem expressed in Section 3.5. The network has three layers. The first layer of the network consists of input (source) nodes whose number is equal to the dimension p of the input vector $\mathbf{x}$ (i.e., the number of independent variables of the problem). The second layer is a hidden layer, made up of nonlinear units that are connected *directly* to all of the nodes in the input layer. There is one hidden unit for each data vector $\mathbf{x}_i$, i = 1, 2,...., N, where N is the number of training samples. The activation function of the individual hidden units are described by the Green's functions. Correspondingly, $G(\mathbf{x}; \mathbf{x}_i)$ represents the output of the ith hidden unit. The output layer has one single linear unit which is fully connected to the hidden layer. The term "linearity" is introduced because the output of the network is a linearly weighted sum of the outputs of the hidden units. The weights of the output layer are the unknown coefficients of the expression described in equation (3.44) in terms of the Green's functions $G(\mathbf{x}; \mathbf{x}_i)$ and the regularization parameter λ. Obviously, such a network structure can be readily extended to have any number of outputs desired [5].

The Green's function $G(\mathbf{x}; \mathbf{x}_i)$ is assumed to be *positive definite* for all i in the regularization network represented in Figure 3.2. Haykin states that if this condition is satisfied, which is true in the case where the Green's functions $G(\mathbf{x}; \mathbf{x}_i)$ have the form of Gaussian functions, then this network will produce an "optimal" interpolant solution in the sense that it minimizes the functional $\xi(F)$. Furthermore, Poggio and Girosi give three

properties of the regularization network from the viewpoint of approximation theory as follows:

1. The regularization network is a *universal approximator* in that it can approximate arbitrarily well any multivariate continuous function on a compact subset of $\mathbb{R}^p$, given a sufficiently large number of hidden units.
2. Since the approximation scheme derived from regularization theory is linear in the unknown coefficients, it follows that the regularization network has the *best-approximation property*. This means that given an unkown nonlinear function *f*, there always exists a choice of coefficients that approximates f better than all other possible choices.
3. The solution computed by the regularization network is *optimal.* Optimality here means that the regularization network minimizes a functional that measures how much the solution deviates from its true value as represented by training data [10].

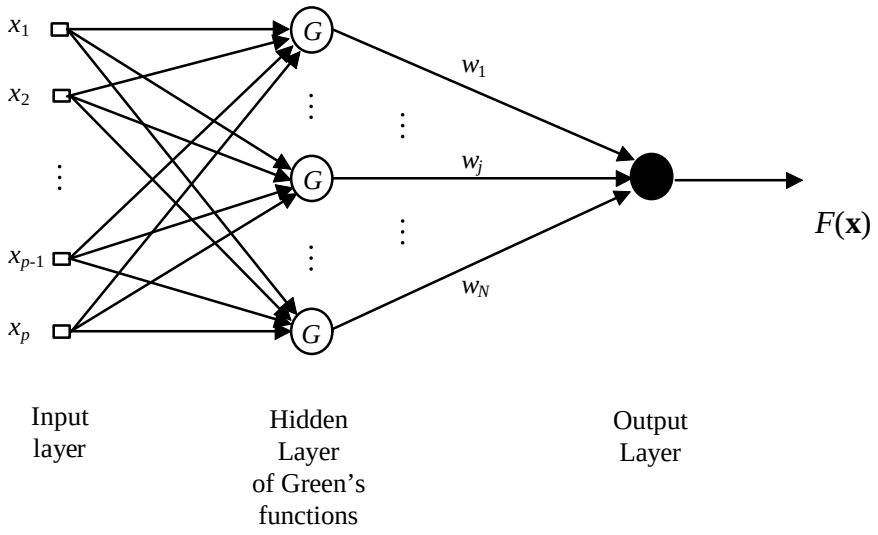

Figure 3.2. Regularization network

A regularization network is prohibitively expensive to build in computational terms for large N because there is a one-to-one correspondence between the training input data $\mathbf{x}_i$ and Green's function $G(\mathbf{x}; \mathbf{x}_i)$ for $i = 1, 2,...., N$. In particular, the inversion of N-by-N matrix is necessary to calculate the linear weights of the network (i.e., the coefficients of the expression in equation (3.45)). When N gets to large, the computational complexity will be very high. Moreover, the probability of *ill conditioning* is higher for larger matrices. Because of these reasons, we need a generalization of the solution in some sense. The next section introduces a generalization of the solution defining a new type of network structure which is called generalized radial basis function network.

3.7 Generalized Radial Basis Function Networks

This section explains the structure of a generalized radial basis function network applying some approximation approaches to the regularized solution. The *condition number* of a matrix is defined as the ratio of the largest eigenvalue to the smallest eigenvalue of the matrix. This number should not be too high not to have a ill-conditioned matrix in the solution. Haykin states that the reduction of the complexity of the network is necessary to overcome the computational difficulties, which demands an approximation to the regularized solution [5].

The approximation approach includes searching for a suboptimal solution in a lower-dimensional space that approximates the regularized solution of equation (3.45). Poggio and Girosi perform this approximation by using a standard technique known as *Galerkin's method* in variational problems [10]. According to this technique, they expand the approximation solution $F^*(\mathbf{x})$ on a finite basis as follows:

$$F^*(\mathbf{x}) = \sum_{i=1}^{M} w_i \varphi_i(\mathbf{x}) \tag{3.49}$$

where $\{\varphi_i(\mathbf{x}) | i = 1, 2, \cdots, M\}$ is a new set of basis functions that are assumed to be linearly independent without loss of generality. In general, the number of basis functions is less than the number of data points (i.e., $M \leq N$), and the w_i represents a new set of weights. Considering radial basis functions, Haykin gives the following equation:

$$\varphi(\mathbf{x}) = G(\|\mathbf{x} - \mathbf{t}_i\|) \qquad i = 1, 2, ..., M \tag{3.50}$$

where the set of centers $\{\mathbf{t}_i \mid i = 1, 2, ..., M\}$ is to be concluded [5]. This special choice of basis functions is the only one that guarantees that the correct solution of equation (3.45) completely recovered in the case of $M = N$ and $\mathbf{t}_i = \mathbf{x}_i$, $i = 1, 2, ..., N$. Substituting equation (3.50) in equation (3.49), $F^*(\mathbf{x})$ can be rewritten as

$$\begin{aligned} F^*(\mathbf{x}) &= \sum_{i=1}^{N} w_i G(\mathbf{x}; \mathbf{t}_i) \\ &= \sum_{i=1}^{N} w_i G(\|\mathbf{x} - \mathbf{t}_i\|) \end{aligned} \tag{3.51}$$

Using the expression in equation (3.51), in order to minimize the new cost functional $\xi(F^*(\mathbf{x}))$, the determination of the new weights $\{w_i \mid i = 1,2,.., M\}$ can be defined by

$$\xi(F^*) = \sum_{i=1}^{N} \left(d_i - \sum_{j=1}^{M} w_i G(\left\|\mathbf{x}_i - \mathbf{t}_j\right\|) \right)^2 + \lambda \left\|\mathbf{P}F^*\right\|^2 \tag{3.52}$$

Haykins insists that the first term on the right-hand side of the equation (3.65) can be phrased as the squared Euclidean norm $\|\mathbf{d} - \mathbf{G}\mathbf{w}\|^2$, where

$$\mathbf{d} = [d_1, d_2, ..., d_N] \tag{3.53}$$

$$\mathbf{G} = \begin{bmatrix} G(\mathbf{x}_1;\mathbf{t}_1) & G(\mathbf{x}_1;\mathbf{t}_2) & \cdots & G(\mathbf{x}_1;\mathbf{t}_M) \\ G(\mathbf{x}_2;\mathbf{t}_1) & G(\mathbf{x}_2;\mathbf{t}_2) & \cdots & G(\mathbf{x}_2;\mathbf{t}_M) \\ \vdots & \vdots & & \vdots \\ G(\mathbf{x}_N;\mathbf{t}_1) & G(\mathbf{x}_N;\mathbf{t}_2) & \cdots & G(\mathbf{x}_N;\mathbf{t}_M) \end{bmatrix} \tag{3.54}$$

$$\mathbf{w} = [w_1, w_2, ..., w_N] \tag{3.55}$$

Here, the desired response has the same dimension, N, as before. On the other hand, the matrix $\mathbf{G}$ of Green's functions and the weight vector $\mathbf{w}$ have different dimensions. The matrix $\mathbf{G}$ has N rows and M columns, and thus it is not symmetric. The vector $\mathbf{w}$ has M rows and one column. Haykins notes that the approximation function F^* is a linear combination of the Green's functions for the stabilizer $\mathbf{P}$ [5]. Correspondingly, he also expresses the second term on the right-hand side of the equation (3.52) as

$$\begin{aligned} \left\|\mathbf{P}F^*\right\|^2 &= \left(\mathbf{P}F^*, \mathbf{P}F^*\right)_H \\ \left\|\mathbf{P}F^*\right\|^2 &= \left[\sum_{i=1}^{M} w_i G(\mathbf{x};\mathbf{t}_i), \mathbf{P} * \mathbf{P}\sum_{i=1}^{M} w_i G(\mathbf{x};\mathbf{t}_i)\right]_H \\ &= \left[\sum_{i=1}^{M} w_i G(\mathbf{x};\mathbf{t}_i), \sum_{i=1}^{M} w_i \delta_{\mathbf{t}_i}\right]_H \\ &= \sum_{j=1}^{M}\sum_{i=1}^{M} w_j w_i G(\mathbf{t}_j;\mathbf{t}_i) \\ &= \mathbf{w}^T \mathbf{G}_0 \mathbf{w} \end{aligned} \tag{3.56}$$

where the matrix $\mathbf{G}_0$ is a symmetric M-by-M matrix, given by

$$\mathbf{G}_0 = \begin{bmatrix} G(\mathbf{t}_1;\mathbf{t}_1) & G(\mathbf{t}_1;\mathbf{t}_2) & \cdots & G(\mathbf{t}_1;\mathbf{t}_M) \\ G(\mathbf{t}_2;\mathbf{t}_1) & G(\mathbf{t}_2;\mathbf{t}_2) & \cdots & G(\mathbf{t}_2;\mathbf{t}_M) \\ \vdots & \vdots & & \vdots \\ G(\mathbf{t}_M;\mathbf{t}_1) & G(\mathbf{t}_M;\mathbf{t}_2) & \cdots & G(\mathbf{t}_M;\mathbf{t}_M) \end{bmatrix} \tag{3.57}$$

Haykin concludes that the minimization of equation (3.52) in relation to the weight vector $\mathbf{w}$ produces the result [5]

$$(\mathbf{G}^T\mathbf{G} + \lambda\mathbf{G}_0)\mathbf{w} = \mathbf{G}^T\mathbf{d} \tag{3.58}$$

Broomhead and Lowe show that as the regularization parameter converges to zero, the weight vector $\mathbf{w}$ converges to the pseudoinverse (minimum-norm) solution to the overdetermined least-square data fitting problem, given the equation [7]

$$\mathbf{w} = \mathbf{G}^+\mathbf{d}, \qquad \lambda = 0 \tag{3.59}$$

where $\mathbf{G}^+$ is the pseudoinverse of the matrix $\mathbf{G}$; that is,

$$\mathbf{G}^+ = (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T \tag{3.60}$$

The framework for the *generalized radial-basis function (RBF) network* shown in Figure 3.3 is provided by the solution to the approximation problem defined in the equation (3.52). In this network, a bias (i.e., data-independent variable) is applied to the output unit. In order to do that, one of the linear weights in the output layer of the network is set equal to a bias and the associated radial basis function is treated as a constant equal to +1.

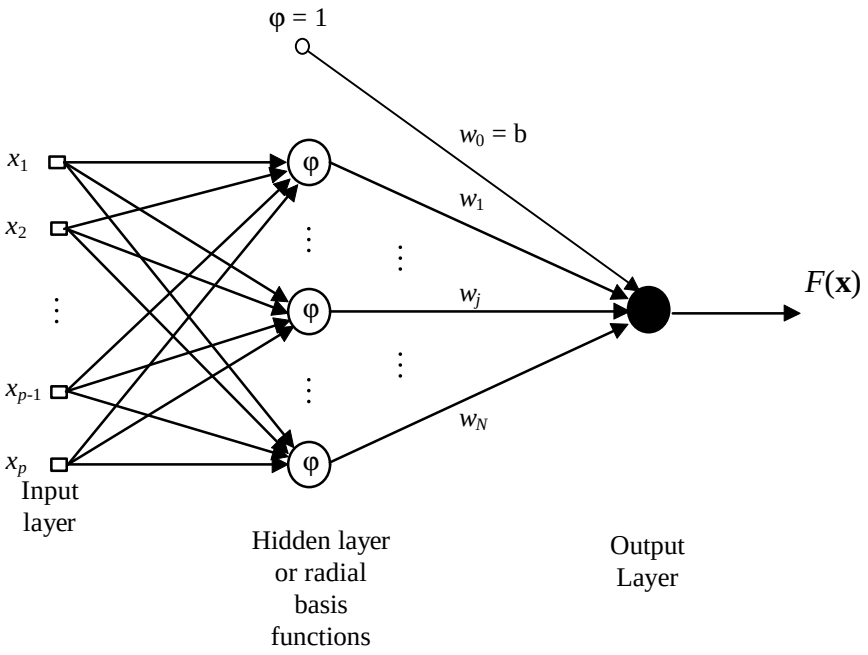

Figure 3.3. Generalized radial basis function network

Although the generalized RBF network is structurally akin to the regularization RBF network, Haykin gives two important differences between these two networks in the following senses:

> 1. The number of nodes in the hidden layer of the generalized RBF network of Figure 3.3 is *M*, where *M* is ordinarily smaller that the number *N* of examples available for training. On the other hand, the number of hidden nodes in the regularization RBF network of Figure 3.2 is exactly *N*.
> 2. In the generalized RBF network of Figure 3.3, the linear weights associated with the output layer, and the positions of the centers of the radial basis functions and the norm weighting matrix associated with the hidden layer, are all unknown parameters that have to be learned. On the other hand, the activation functions of the hidden layer in the regularization RBF network of Figure 3.2 are known, being defined by a set of Green's functions centered at the training data points; the linear weights of the output layer are the only unknown parameters of the network [5].

3.7.1 Weighted Norm

The norm in equation (3.52) is generally intended to be a Euclidean norm. On the other hand, it is more appropriate to use a general *weighted norm* when the individual components of the input vector **x** belong to different classes. Poggio and Girosi give the squared form of a weighted norm as follows [10]:

$$\|\mathbf{x}\|^2 = (\mathbf{Cx})^T(\mathbf{Cx}) = \mathbf{x}^T\mathbf{C}^T\mathbf{Cx} \tag{3.61}$$

where **C** is a *p*-by-*p norm weighting matrix,* and *p* represesnts the dimension of the input vector **x**. Haykin gives three specific cases of interest depending on how the weighting matrix **C** is described.

1. The matrix **C** is equal to the identity matrix **I**, for which the standard Euclidean norm is obtained; that is,

$$\|\mathbf{x}\|_C^2 = \|\mathbf{x}\|^2, \quad \mathbf{C} = \mathbf{I} \tag{3.62}$$

2. The matrix **C** is a diagonal matrix, in which case the diagonal elements assign a specific weight to each input coordinate, as shown by

$$\|\mathbf{x}\|_c^2 = \sum_{k=1}^{p} c_k^2 x_k^2 \tag{3.63}$$

where x_k is th *k*th element of the input vector **x** and c_k is the *k*th diagonal element of matrix **C**.

3. The matrix **C** is a nondiagonal matrix, in which case the weighted norm takes on a quadratic form as shown by

$$\|\mathbf{x}\|_C^2 = \sum_{k=1}^{p}\sum_{l=1}^{p} a_{kl} x_k x_l \tag{3.64}$$

where a_{kl} is the *kl*th element of the matrix product $\mathbf{C}^T\mathbf{C}$ [5].

Considering the definition of the weighted norm, the approximation to the regularized solution expressed in equation (3.52) can be rewritten in the more generalized form [10].

$$F^*(\mathbf{x}) = \sum_{i=1}^{M} w_i G(\|\mathbf{x} - \mathbf{t}_i\|_{\mathbf{C}_i}) \tag{3.65}$$

Haykin introduces a Gaussian radial basis function $G(\|\mathbf{x} - \mathbf{t}_i\|_{\mathbf{C}_i})$ centered at $\mathbf{t}_i$ and with norm weighing matrix $\mathbf{C}_i$ as follows:

$$G(\|\mathbf{x} - \mathbf{t}_i\|_{\mathbf{C}_i}) = \exp[-(\mathbf{x} - \mathbf{t}_i)^T \mathbf{C}_i^T \mathbf{C}_i (\mathbf{x} - \mathbf{t}_i)] = \exp\left[\frac{1}{2}(\mathbf{x} - \mathbf{t}_i)\sum_i^{-1}(\mathbf{x} - \mathbf{t}_i)\right] \tag{3.66}$$

where the inverse matrix $\sum_i^{-1}$ is given by

$$\frac{1}{2}\sum_i^{-1} = \mathbf{C}_i^T \mathbf{C}_i \tag{3.67}$$

Equation (3.66) is a multivariable Gaussian distribution with mean vector $\mathbf{t}_i$ and covariance matrix $\sum_i$. It is a generalization of the distribution defined in equation (3.48). In this work the covariance matrix is determined by calculating covariance matrix of the input vectors of sample images. This allows us to find the optimum $\mathbf{C}_i$. The calculation of covariance matrix $\sum_i$ is managed by statistical methods. This statistical method is given in Chapter 4.

The generalized radial basis function network is defined and explained in detail in this section. This network structure is employed in this work to classify images of finished kitchen cabinets and doors, introduced in Chapter 1. The generalized radial basis function networks are capable of solving nonlinear classification problem with generalization. The next section introduces an example of nonlinear classification problem solved by generalized radial basis function network. The XOR problem is introduced in Section 3.8 as such a nonlinear classification problem.

3.8 XOR Problem

The XOR problem is one of the popular examples of a nonlinear classification problem. In this problem there are four points (patterns), namely, (1,1), (0,1), (0,0) and (1,0), in a two dimensional input space, as shown in Figure 3.4a. The requirement is to construct a pattern classifier that gives the binary input 0 in response to the input patterns (1,1) or (0,0), and the binary output 1 in response to the input pattern (0,1) or (1,0). As discussed before, RBFs transform a nonlinearly separable problem into a linearly separable problem basically converting the input space into higher dimensional space. Haykin uses a pair of Gaussian functions to make this transformation possible [5]. The pair of Gaussian hidden functions is given as follows:

$$\varphi_1 = e^{-\|\mathbf{x}-\mathbf{t}_1\|}, \qquad \mathbf{t}_1 = [1, 1]^T$$

$$\varphi_2 = e^{-\|\mathbf{x}-\mathbf{t}_2\|}, \qquad \mathbf{t}_2 = [0, 0]^T$$

where $\mathbf{t}_1$ and $\mathbf{t}_2$ are the centers of the gaussian functions. The results are summarized in Table 3.1 for the four different input patterns of interest. Correspondingly, the input patterns are mapped onto the φ_1-φ_2 plane as shown in Figure 3.4b. As can been seen in Figure 3.4b, there is no increase in the dimensionality of the hidden-unit space compared to the input space because the use of Gaussian hidden functions is satisfactory to transform the XOR problem into linearly separable one.

Table 3.1. Specification of the Hidden Functions for XOR problem

Input Pattern **x**	First Hidden Function, $\varphi_1(\mathbf{x})$	Second Hidden Function $\varphi_2(\mathbf{x})$
(1, 1)	1	0.1353
(0, 1)	0.3678	0.3678
(0, 0)	0.1353	1
(1, 0)	0.3678	0.3678

T

When we construct our network we can obtain our **G** matrix as follows:

$$G(\|\mathbf{x} - \mathbf{t}_i\|) = \exp(-\|\mathbf{x} - \mathbf{t}_i\|^2), \qquad i = 1, 2 \qquad (3.68)$$

For the characterization of the output unit, Haykin gives the following two assumptions [5]. First, the output unit uses *weight sharing* because of the symmetry of the problem. In particular, the values in the Table 3.1 are symmetric. For example, the output of the hidden units are equal when the input is either (0, 1) or (1, 0). Using only two hidden units, therefore, is enough to have a single weight w to be determined. Second, the output unit includes a bias b. The significance of this bias that the desired output values of the XOR function have nonzero mean.

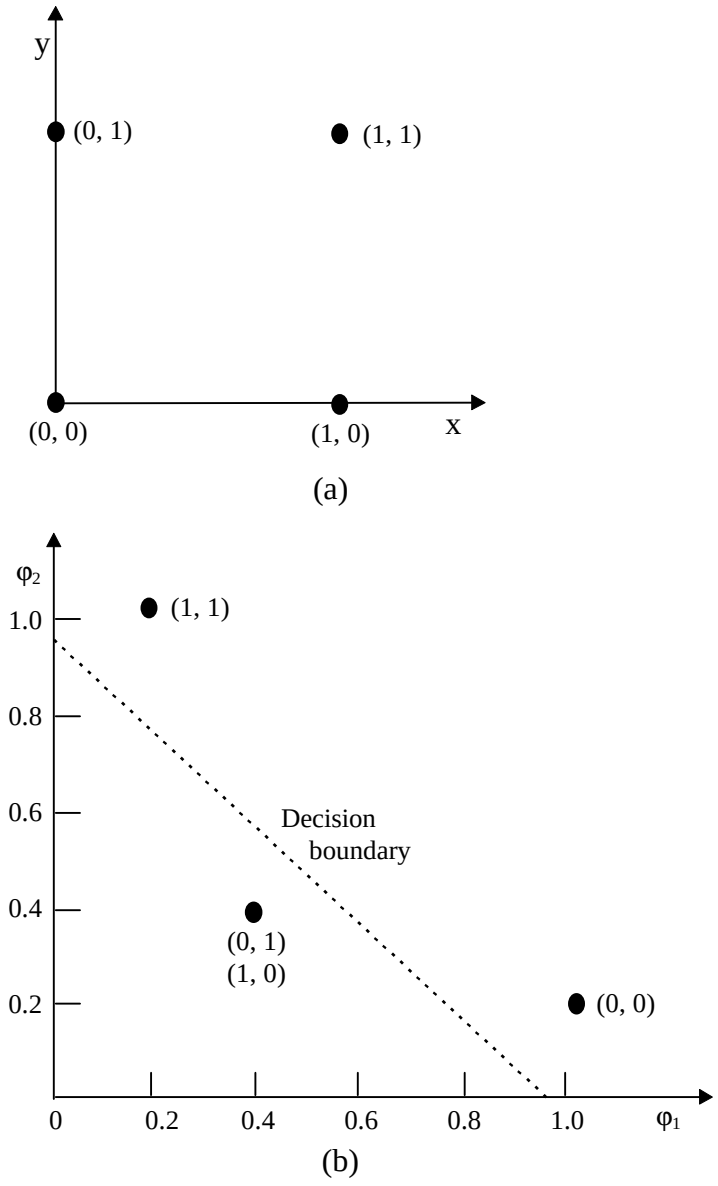

Figure 3.4 (a) The four patterns of the XOR problem; (b) Decision-making diagram.

The structure of the RBF network proposed for solving the XOR problem is shown in Figure 3.5. There is no increase in the dimensionality of the hidden-unit space compared to the input space because the two Gaussian hidden functions are enough to transform the XOR problem into a linearly separable one.

The relationship between the input and the output of the network can be given by

$$y(\mathbf{x}) = \sum_{i=1}^{2} wG(\|\mathbf{x} - \mathbf{t}_i\|) + b \tag{3.69}$$

$$y(\mathbf{x}_j) = d_{j,} \qquad j = 1, 2, 3, 4 \tag{3.70}$$

where $\mathbf{x}_j$ is an input vector and d_j is the associated value of the desired output. Let

$$g_{ji} = G(\|\mathbf{x}_j - \mathbf{t}_i\|), \qquad j = 1, 2, 3, 4;\ i = 1, 2. \tag{3.71}$$

Then substituting the values of Table 3.2 in equation (3.84), Haykin gives the following set of equations written in matrix form:

$$\mathbf{Gw} = \mathbf{d} \tag{3.72}$$

where

$$\mathbf{G} = \begin{bmatrix} g_{11} & g_{12} & 1 \\ g_{21} & g_{22} & 1 \\ g_{31} & g_{32} & 1 \\ g_{41} & g_{42} & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0.1353 & 1 \\ 0.3678 & 0.3678 & 1 \\ 0.1353 & 1 & 1 \\ 0.3678 & 0.3678 & 1 \end{bmatrix} \tag{3.73}$$

$$\mathbf{d} = [\,1 \quad 0 \quad 1 \quad 0\,]^T \tag{3.74}$$

$$\mathbf{w} = [\,w \quad w \quad b\,]^T \tag{3.75}$$

As can be seen, the elements of the matrix **G** are the outputs of the hidden units corresponding the four input patterns. These elements are calculated by equation (3.71).

Table 3.2. Input-Output Transformation Computed for XOR Problem [5].

Data Point, j	Input Pattern, $\mathbf{x}_j$	Desired Output, d_j	Actual Output, y_i
1	(1, 1)	1	+0.901
2	(0, 1)	0	-0.01
3	(0, 0)	1	+0.901
4	(1, 0)	0	-0.01

The problem represented here is *overdetermined* in the sense that there are more data points than free parameters [5]. That is why the **G** matrix is not square. Therefore, matrix **G** does not have a unique inverse. The *minumum norm* solution of equation (3.60) can be employed to defeat this difficulty in the following form:

$$\begin{aligned} \mathbf{w} &= \mathbf{G}^{+}\mathbf{d} \\ &= (\mathbf{G}^T\mathbf{G})^{-1}\mathbf{G}^T\mathbf{d} \end{aligned} \tag{3.76}$$

where $\mathbf{G}^T\mathbf{G}$ is a square matrix with a unique inverse of its own. Combining equation (3.73) and equation (3.76), the following matrix is obtained.

$$\mathbf{G} = \begin{bmatrix} 1.656 & -1.158 & -0.628 & -1.158 \\ 0.628 & -1.158 & -1.656 & -1.158 \\ 0.846 & -1.301 & -0.846 & -1.301 \end{bmatrix} \tag{3.77}$$

Finally, using the matrices in equation (3.74) and (3.77), the weights can be calculated as

$$\mathbf{w} = \begin{bmatrix} 2.284 \\ 2.284 \\ -1.692 \end{bmatrix} \tag{3.78}$$

which completes the specification of the RBF network.

The last column of Table 3.2 displays the real values of the RBF network output generated in response to the four input patterns [5]. As can be seen in Table 3.2, the results are very near to desired values. The output of the network is -0.01 when the desired is 0. Similarly, the output of the network is 0.901 when the desired output is 1. If we take the absolute values of the output and put a hard limiter centered at 0.45, we can have desired outputs 0 and 1. These results indicate that the RBF network can successfully separate states of the opposite polarity, and solve the problem.

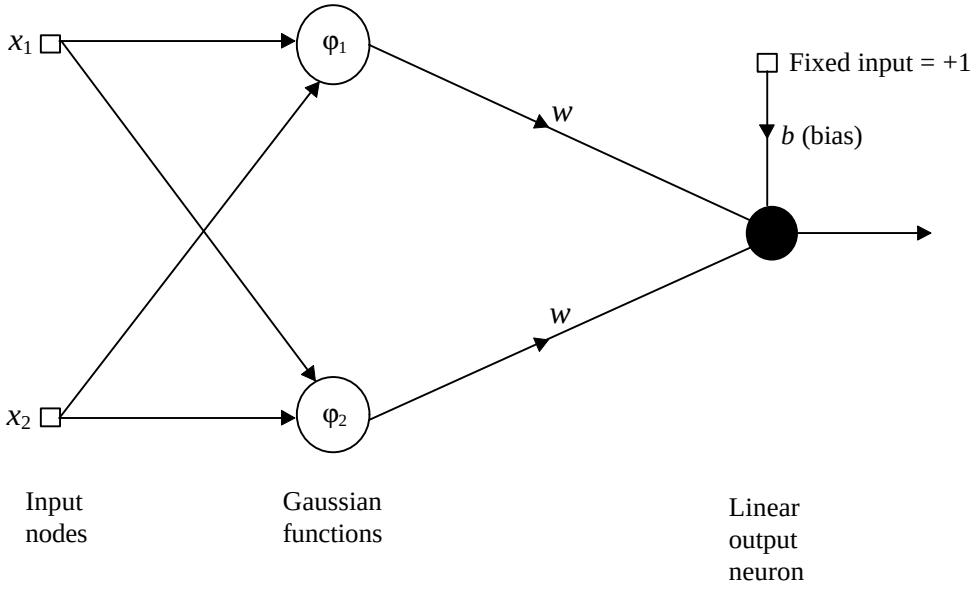

Figure 3.5. RBF network solving the XOR problem.

3.9 Learning Strategies

In a RBF network, different layers perform different tasks. Therefore, it is useful to separate the optimization of the hidden unit and output layers of the network by using different techniques. There are different learning strategies in the design of an RBF network depending on how the centers of RBFs of the network are determined. There are three major approaches to determine the centers.

3.9.1 Fixed Centers Selected at Random

In this approach, the locations of the centers may be chosen randomly from the training data. The activation functions of the hidden units are *fixed* radial basis functions. Lowe insists that this is a "sensible" approach, if the training data are distributed in a representative manner for the problem at hand [67]. Haykin states that the *isotropic* Gaussian function, whose standard deviation is fixed according to the spread of the centers, can be used as the radial basis function. In particular, he gives a (normalized) radial basis function centered at $\mathbf{t}_i$ as

$$G(\|\mathbf{x} - \mathbf{t}_i\|) = \exp(-\frac{M}{d^2}\|\mathbf{x} - \mathbf{t}_i\|^2), \qquad i = 1, 2, ..., M \tag{3.79}$$

where M is the number of centers and d is the maximum distance between the chosen centers [5]. All the Gaussian radial basis functions have the same standard deviation (i.e., witdh) at

$$\sigma = \frac{d}{\sqrt{2M}} \tag{3.80}$$

Such a choice for the standard deviation σ does not guarantee that the Gaussian functions are not too peaked or too flat. These extremes have to be avoided.

In this learning strategy, the only parameters that would need to be learned are the linear weights in the output layer of the network. A straightforward procedure is given by Broomhead and Lowe, called the *pseudoinverse method*. Particularly, the linear weights can be calculated using the following formulation:

$$\mathbf{w} = \mathbf{G}^{+}\mathbf{d} \tag{3.81}$$

where $\mathbf{d}$ is the desired response vector in the training set. The matrix $\mathbf{G}^{+}$ is the pseusoinverse of the matrix $\mathbf{G}$, which is itself described as

$$g_{ji} = \exp(-\frac{M}{d^2}\|\mathbf{x} - \mathbf{t}_i\|^2), \quad j = 1, 2, ..., N; \; i = 1, 2, ..., M \tag{3.82}$$

where $\mathbf{x}_j$ is the jth input vector of the training set.

This method can be useful if the training data are distributed in a representative manner for the specified problem. In our case, we are not really sure that our samples are distributed in a representative manner for our problem. Since there is an uncertainity in this matter, we intend not to use this learning strategy. There should be some kind of learning or a clustering algorithm to define the centers of RBFs appropriately. The next approach describes a self-organized selection of the centers to ease this difficulty.

3.9.2 Self-Organized Selection of Centers

In the second approach, the radial basis functions can move the locations of their centers in a self-organized fashion. On the other hand, the linear weights of the output layer are computed employing a *supervised learning* rule. In particular, the network experiences a *hybrid learning process* [8] because it has both superwised and unsuperwised learning processes. The self-organized element of the learning procedure allocates network resources in a meaningful way by placing the centers of radial basis functions in only those regions of the input space where important data exist [5].

Moody and Darken use the standard *k-nearest-neighbor rule* for the self-organized selection of the hidden units' centers [8]. This rule classifies an input vector **x** by assigning the label of most frequently represented class among the *k* nearest samples. In fact, a decision is made by examining the labels on*k*-nearest neighbors and taking a vote.

The simple and yet highly effective *least-mean-square* (*LMS*) algorithm can be employed for the supervised learning procedure to calculate the linear weights of the output layer. The LMS algorithm is an error-correction learning rule. The inputs of the LMS algorithm are the outputs of the hidden units in the RBF network [5].

3.9.3 Supervised Selection of Centers

In the third approach, a supervised learning process is employed to obtain the centers of the radial basis functions and all other free parameters of the network. Haykin claims that the RBF network goes into its most generalized form in this approach. He also states that an error-correction learning is a natural candidate for such a process, which is most conveniently implemented using a gradient-decent procedure that indicates a generalization of the LMS algorithm [5].

Defining the instantaneous value of the cost function can be the first step in the generation of such a learning process. The cost function can be given by

$$\xi = \frac{1}{2}\sum_{j=1}^{N} e_j^2 \tag{3.83}$$

where N is the number of training samples used to undertake the learning procedure, and e_j is the errror signal, given by

$$\begin{aligned} e_j &= d_j - F^*(\mathbf{x}_j) \\ &= d_j - \sum w_i G(\left\|\mathbf{x}_j - \mathbf{t}_i\right\|_{\mathbf{C}_i}) \end{aligned} \tag{3.84}$$

In this approach, the calculation of the free parameters w_i, $\mathbf{t}_i$, and $\sum_i^{-1}$ is necessary in order to minimize ξ. The covariance matrix $\sum_i$ is related to the norm-weighting matrix **C** as explained in Section 3.7.1. The selection method of the matrix **C** is also explained in Section 3.7.1. The detailed equations of the results of this minimization are given in [5].

The adaptation of the positions of the centers of the radial basis functions does not necessarily provide the best results. The success of the moving centers depends on the application of interest. There are some applications for which adaptation of the centers of the RBFs has practical advantages. For example, it allows us to define the classes (input patterns) better and more accurately. The following chapter gives some applications of RBFNs in various fields.

Work done by Lowe on speech recognition using RBF networks shows that nonlinear optimization of the parameters that define the activation functions of the hidden layer is advantageous [67]. On the other hand, according to Lowe, the same performance on generalization may be achieved by using a larger RBF network, that is, a network with a larger number of fixed centers in the hidden layer, and only adapting the output layer of the network by linear optimization.

Wettschereck and Dietterich give a comparison between the performance of (Gaussian) radial basis function networks with fixed centers and that of generalized radial basis function with adjustable centers [68]. The adjustment in the latter case is made by supervised learning. The performance comparison was made for the NETtalk task to map English spelling into its phonetic pronounciation. The experimental study by Wettscherech and Dictterich indicates that generalized RBF networks (with supervised learning of the centers' locations) are able to exceed the generalization performance of muti-layer perceptrons trained with the back-propagation algorithm while RBF networks (with unsupervised learning of the centers' location and supervised learning of the output layer weights) cannot generalize as well as multi-layer perceptrons. Since the success of the selection of the center's location in a RBF network is problem-dependent, several training methods were tried in this work. Chapter 4 explains what kind of training algorithms are used for our classification problem.

3.10 Applications

Radial basis function networks have been employed in many different problems even though they have not been used as often as multi-layer perceptrons. However, in the literature, the number of applications covered by RBFNs is quite high. Some of the major application areas for RBFNs can be listed as follows:

- Image processing [3, 69, 73]
- Speech recognition [70, 71, 72, 79, 9]
- Time-series analysis [74, 75, 8, 7, 52]
- Adaptive equalization [76, 77, 78, 80]
- Radar point source location [81]
- Medical diagnosis [82]
- Process faults detection [83]
- Pattern recognition [84, 85, 86, 87]

As can be seen, the RBF networks are employed mostly in classification problems. The main classification problem involving RBF networks is the pattern

recognition problem. Time-series analysis is the second most common application area for RBF networks.

Many pattern recognition experiments show that the RBFNs are superior over other neural network approaches in the following senses. First, the RBFNs are capable of approximating nonlinear mappings effectively [78]. Second, the training time of the RBFNs is quite low compared to that of other neural network approaches such as the multi-layer perceptron, because training of the two layers of the network is decoupled [89, 83]. Third, the RBFNs produce classification accuracies from 5% to 10% higher than accuracies produced by the back propagation algorithm as described in the paper by Chapman et al. [90]. Fourth, the RBFNs are quite successful for identifying regions of sample data not in any known class because it uses a nonmonotonic transfer function based on the Gaussian density function [84]. Finally, the work done by Zhong et al. indicates that the RBFNs perform better than the conventional kernel classifiers [88].

A multi-layer perceptron can separate the classes by using hidden units which form hyperplanes in the input space, as indicated in Figure 3.6a. Bishop insists that an alternative approach is to model the separate class distributions by local kernel functions, as indicated in Figure 3.6b [58]. He then states that this latter type of representation is related to the radial basis functions. Mathematical justification of this type of representation can be examined in [58].

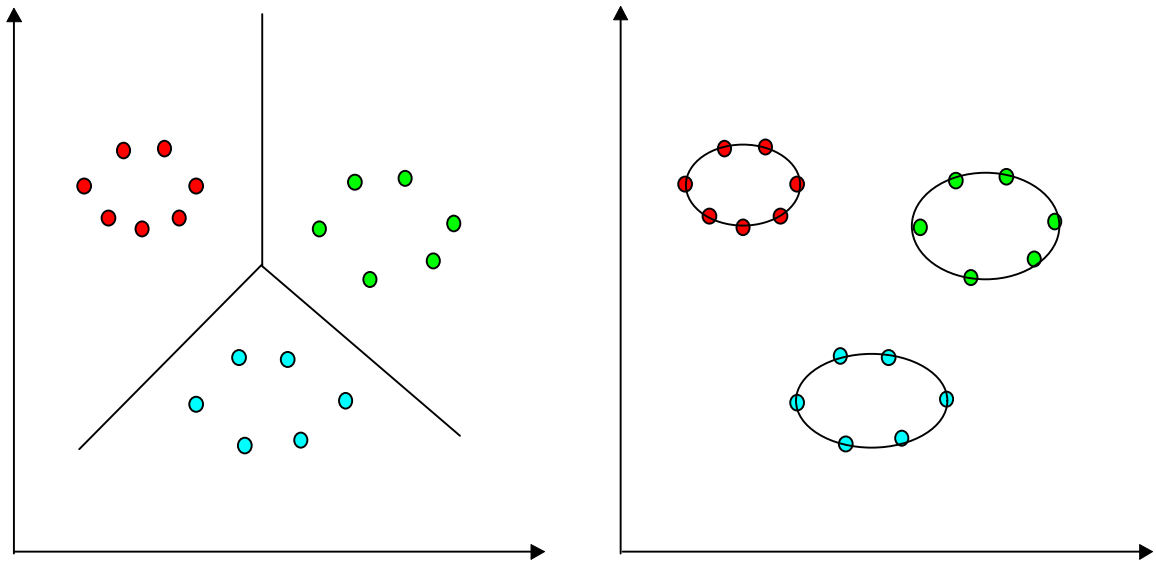

Figure 3.6. Decision surfaces in two dimensional space: (a) Multi-layer perceptron; (b) Radial basis function network

In light of all these reasons, the RBFN has been selected to solve the problem of the classification of color and species of finished wooden kitchen cabinets in this work. The following chapter explains the application of the RBFNs to the specified problem above.