7.1 Introduction

The design of a supervised neural network may be pursued in a variety of different ways. The back-propagation algorithm for the design of a multilayer perceptron (under supervision) as described in the previous chapter may be viewed as an application of an optimization method known in statistics as stochastic approximation. In this chapter we take a different approach by viewing the design of a neural network as a curve-fitting (approximation) problem in a high-dimensional space. According to this viewpoint, learning is equivalent to finding a surface in a multidimensional space that provides a best fit to the training data, with the criterion for “best fit” being measured in some statistical sense. Correspondingly, generalization is equivalent to the use of this multidimensional surface to interpolate the test data. Such a viewpoint is indeed the motivation behind the method of radial-basis functions in the sense that it draws upon research work on traditional strict interpolation in a multidimensional space. In the context of a neural network, the hidden units provide a set of “functions” that constitute an arbitrary “basis” for the input patterns (vectors) when they are expanded into the hidden-unit space; these functions are called radial-basis functions. Radial-basis functions were first introduced in the solution of the real multivariate interpolation problem. The early work on this subject is surveyed by Powell (1985). It is now one of the main fields of research in numerical analysis.

Broomhead and Lowe (1988) were the first to exploit the use of radial-basis functions in the design of neural networks. Other major contributions to the theory, design, and application of radial-basis function networks include papers by Moody and Darken (1989), Renals (1989), and Poggio and Girosi (1990a). The paper by Poggio and Girosi emphasizes the use of regularization theory applied to this class of neural networks as a method for improved generalization to new data.

The construction of a radial-basis function (RBF) network in its most basic form involves three entirely different layers. The input layer is made up of source nodes (sensory units). The second layer is a hidden layer of high enough dimension, which serves a different purpose from that in a multilayer perceptron. The output layer supplies 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, whereas the transformation from the hidden-unit space to the output space is linear. A mathematical justification for this rationale may be traced back to an early paper by Cover (1965). In particular, we note from this paper that a pattern-classification problem cast in a high-dimensional space non-linearly is more likely to be linearly separable than in a low-dimensional space—hence the reason for making the dimension of the hidden-unit space in an RBF network high. Through careful design, however, it is possible to reduce the dimension of the hidden-unit space, especially if the centers of the hidden units are made adaptive.
Organization of the Chapter

The main body of the chapter is organized as follows. In Section 7.2 we describe Cover's theorem on the separability of patterns, and revisit the XOR problem in light of Cover's theorem; we thus pave the way for the introduction of RBF networks, which we take up in Section 7.3 by considering the interpolation problem and its relationship to RBF networks. Then, in Section 7.4, we discuss the viewpoint that supervised learning is an ill-posed hypersurface-reconstruction problem. In Section 7.5 we present a detailed treatment of Tikhonov's regularization theory and its application to RBF networks, which leads naturally to the formulation of regularization RBF networks that are described in Section 7.6. This is followed by a detailed discussion of generalized RBF networks in Section 7.7. In Section 7.8 we revisit the XOR problem once more and show how it can be solved using an RBF network. In Section 7.9 we present a comparison between RBF networks and multilayer perceptrons representing two different examples of layered feedforward networks. In Section 7.10 we discuss the relationship between RBF networks and Gaussian mixture models. In Section 7.11 we present different learning strategies for the design of RBF networks. In Section 7.12 we describe a computer experiment on a simple pattern-recognition problem involving the use of RBF networks. In Section 7.13 we discuss the idea of factorizable radial-basis functions, for which the Gaussian description is so well suited. Section 7.14 on discussion and Section 7.15 on applications conclude the chapter.

7.2 Cover's Theorem on the Separability of Patterns

When a standard radial-basis function (RBF) network is used to perform a complex pattern-classification task, the problem is basically solved by transforming it into a high-dimensional space in a nonlinear manner. The underlying justification for so doing is provided by Cover's theorem on the separability of patterns, which states that a complex pattern-classification problem cast in high-dimensional space nonlinearly is more likely to be linearly separable than in a low-dimensional space (Cover, 1965). From the work we did on the perceptron in Chapter 4, we know that once we have linearly separable patterns, then the classification problem is relatively easy to solve. Accordingly, we may develop a great deal of insight into the operation of an RBF network as a pattern classifier by studying the separability of patterns, which we do in this section.

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) \( x_1, x_2, \ldots, x_N \), 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 \( x \in X \), define a vector made up of a set of real-valued functions \( \{ \varphi_i(x) | i = 1, 2, \ldots, M \} \), as shown by

\[
\varphi(x) = [\varphi_1(x), \varphi_2(x), \ldots, \varphi_M(x)]^T
\]

Suppose that the pattern \( x \) is a vector in a \( p \)-dimensional input space. The vector \( \varphi(x) \) then maps points in \( p \)-dimensional input space into corresponding points in a new space of dimension \( M \). We refer to \( \varphi_i(x) \) as a hidden function, because it plays a role similar to that of a hidden unit in a feedforward neural network.

A dichotomy \( \{ X^+, X^- \} \) of \( X \) is said to be \( \varphi \)-separable if there exists an \( m \)-dimensional vector \( w \) such that we may write (Cover, 1965)

\[
w^T \varphi(x) \geq 0, \quad x \in X^+
\]
and

\[ w^T \varphi(x) < 0, \quad x \in X^- \]  \hspace{1cm} (7.2)

The hyperplane defined by the equation

\[ w^T \varphi(x) = 0 \]  \hspace{1cm} (7.3)

describes the separating surface in the \( \varphi \) space. The inverse image of this hyperplane, that is,

\[ \{ x : w^T \varphi(x) = 0 \} \]  \hspace{1cm} (7.4)

defines the separating surface in the input space.

Consider a natural class of mappings obtained by using a linear combination of \( r \)-wise products of the pattern vector coordinates. The separating surfaces corresponding to such mappings are referred to as \( r \)-th order rational Varieties. A \( r \)-th order rational variety in a space of dimension \( p \) is described by an \( r \)-th degree homogeneous equation in the coordinates of the input vector \( x \), as shown by

\[ \sum_{0 \leq i_1, i_2, \ldots, i_r \leq p} a_{i_1, i_2, \ldots, i_r} x_{i_1} x_{i_2} \cdots x_{i_r} = 0 \]  \hspace{1cm} (7.5)

where \( x_i \) is the \( i \)-th component of the input vector \( x \), and \( x_0 \) is set equal to unity in order to express the equation in a homogeneous form. Examples of separating surfaces of this type are hyperplanes (first-order rational varieties), quadrics (second-order rational varieties), and hyperspheres (quadrics with certain linear constraints on the coefficients). These examples are illustrated in Fig. 7.1 for a configuration of five points in a two-dimensional input space. Note that, in general, linear separability implies spherical separability, which in turn implies quadric separability; however, the converses are not necessarily true.

\[ \begin{array}{c}
\text{(a)} \\
\begin{array}{c}
\circ \\
\circ \\
\circ \\
\circ \\
\circ
\end{array}
\end{array} \\
\begin{array}{c}
\text{(b)} \\
\begin{array}{c}
\times \\
\times \\
\times \\
\circ \\
\circ
\end{array}
\end{array} \\
\begin{array}{c}
\text{(c)} \\
\begin{array}{c}
\circ \\
\times \\
\circ \\
\times
\end{array}
\end{array} \]

**FIGURE 7.1** Three examples of \( \varphi \)-separable dichotomies of different sets of five points in two dimensions: (a) linearly separable dichotomy; (b) spherically separable dichotomy; (c) quadrically separable dichotomy.
**Polynomial separability**, as described here, may be viewed as a natural generalization of linear separability. The important point to note here is that, given a set of patterns \( \mathbf{x} \) in an input space of arbitrary dimension \( p \), we can usually find a nonlinear mapping \( \varphi(\mathbf{x}) \) of high enough dimension \( M \) such that we have linear separability in the \( \varphi \) space.

### Separability of Random Patterns

Consider next the separability of patterns that are randomly distributed in the input space. The desired dichotomization may be fixed or random. Under these conditions the separability of the set of pattern vectors becomes a random event that depends on the dichotomy chosen and the distribution of the patterns.

Suppose that the input vectors (patterns) \( \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \) are chosen independently according to a probability measure \( p \) on the input (pattern) space. The set \( X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \) is said to be in \( \varphi \)-**general position** if every \( m \)-element subset of the set of \( M \)-dimensional vectors \( \{\varphi(\mathbf{x}_1), \varphi(\mathbf{x}_2), \ldots, \varphi(\mathbf{x}_N)\} \) is linearly independent for all \( m \leq M \). The necessary and sufficient condition on the probability measure \( \mu \) such that, with probability 1, the vectors \( \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \) are in \( \varphi \)-general position in \( M \)-space is that the probability be zero that any point will fall in any given \((M - 1)\)-dimensional subspace. Equivalently, in terms of the \( \varphi \) surfaces, we may state that a set of vectors \( \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N \) chosen independently according to a probability measure \( \mu \) is in \( \varphi \)-general position with probability 1 if, and only if, every \( \varphi \) surface has \( \mu \) measure zero.

Suppose, next, that a dichotomy of \( X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\} \) is chosen at random with equal probability from the \( 2^N \) equiprobable possible dichotomies of \( X \). Let \( X \) be in \( \varphi \)-general position with probability 1. Let \( P(N, M) \) be the probability that the particular dichotomy picked at random is \( \varphi \)-separable, where the class of \( \varphi \) surfaces has \( M \) degrees of freedom. Then, we may make the following two statements (Cover, 1965):

1. With probability 1 there are \( C(N, M) \) homogeneously \( \varphi \)-separable dichotomies, defined by

\[
C(N, M) = 2 \sum_{m=0}^{M-1} \binom{N-1}{m}
\]  

(7.6)

where the binomial coefficients comprising \( N \) and \( M \) are themselves defined for all real \( l \) and integer \( m \) by

\[
\binom{l}{m} = \frac{l(l-1) \cdots (l-m+1)}{m!}
\]

2. The probability that the random dichotomy is \( \varphi \)-separable is given by

\[
P(N, M) = \left( \frac{1}{2} \right)^{N-1} \sum_{m=0}^{M-1} \binom{N-1}{m}
\]  

(7.7)

which is just the cumulative binomial distribution corresponding to the probability that \( N - 1 \) flips of a fair coin will result in \( M - 1 \) or fewer heads.

Equation (7.6) is Schlafli’s formula for function-counting, and Eq. (7.7) is Cover’s separability theorem for random patterns.

The proof of Cover’s formula follows immediately from (1) Schlafli’s formula for linearly separable dichotomies, and (2) the reflection invariance of the joint probability distribution of \( X \). This invariance implies that the probability (conditional on \( X \)) that a
random dichotomy of $X$ be separable is equal to the unconditional probability that a particular dichotomy of $X$ (all $N$ points in one category) be separable (Cover, 1965).

The important point to note from Eq. (7.7) is that the higher $M$ is, the closer will be the probability $P(N, M)$ to unity.

To sum up, Cover's separability theorem encompasses two basic ingredients:

1. Nonlinear formulation of the hidden function defined by $\varphi_i(x)$, where $x$ is the input vector and $i = 1, 2, \ldots, M$.
2. High dimensionality of the hidden-unit space compared to the input space, which is determined by the value assigned to $M$ (i.e., the number of hidden units).

Thus, in general, a complex pattern-classification problem cast in high-dimensional space nonlinearly is more likely to be linearly separable than in a low-dimensional space. It should be stressed, however, that in some cases the use of nonlinear mapping (i.e., point 1) may be sufficient to produce linear separability without having to increase the dimensionality of the hidden-unit space; see Example 1.

**Separating Capacity of a Surface**

Let $\{x_1, x_2, \ldots\}$ be a sequence of random patterns, and define the random variable $N$ to be the largest integer such that the set $\{x_1, x_2, \ldots, x_N\}$ is $\varphi$-separable, where the function $\varphi$ has $M$ degrees of freedom. Then, using Eq. (7.7), we find that the probability that $N = k$ is given by the negative binomial distribution:

$$\Pr\{N = k\} = P(k, M) - P(k + 1, M)$$

$$= \left(\begin{array}{c} k - 1 \\ M - 1 \end{array}\right) \left(\frac{1}{2}\right)^k, \quad k = 0, 1, 2, \ldots$$

(7.8)

Thus $N$ corresponds to the waiting time for the $M$th failure in a series of tosses of a fair coin, and

$$E[N] = 2M$$

$$\text{median}[N] = 2M$$

(7.9)

The asymptotic probability that $N$ patterns are separable in a space of dimension

$$M \approx \frac{N}{2} + \frac{\alpha}{2} \sqrt{N}$$

is given by

$$P\left(\frac{N}{2} + \frac{\alpha}{2} \sqrt{N} \right) \approx \Phi(\alpha)$$

(7.10)

where $\Phi(\alpha)$ is the cumulative Gaussian distribution; that is,

$$\Phi(\alpha) = \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{\alpha} e^{-x^2/2} dx$$

(7.11)

In addition, for $\varepsilon > 0$ we have

$$\lim_{M \to \infty} P(2M(1 + \varepsilon), M) = 0$$

$$P(2M, M) = \frac{1}{2}$$

(7.12)

$$\lim_{M \to \infty} P(2M(1 - \varepsilon), M) = 1$$
Thus the probability of separability shows a pronounced threshold effect when the number of patterns is equal to twice the number of dimensions. This result suggests that \( 2M \) is a natural definition for the **separating capacity** of a family of decision surfaces having \( M \) degrees of freedom (Cover, 1965).

**EXAMPLE 1. The XOR Problem**

To illustrate the significance of the idea of \( \varphi \)-separability of patterns, consider the simple and yet important XOR problem. In the XOR problem there are four points (patterns), namely, \((1,1)\), \((0,1)\), \((0,0)\), and \((1,0)\), in a two-dimensional input space, as depicted in Fig. 7.2a. The requirement is to construct a pattern classifier that produces the binary output 0 in response to the input pattern \((1,1)\), or \((0,0)\), and the binary output 1 in response to the input pattern \((0,1)\) or \((1,0)\). Thus points that are closest in the input space, in terms of the Hamming distance, map to regions that are maximally apart in the output space.

Define a pair of Gaussian hidden functions as follows:

\[
\varphi_1(x) = e^{-|x-t_1|^2}, \quad t_1 = [1,1]^T \\
\varphi_2(x) = e^{-|x-t_2|^2}, \quad t_2 = [0,0]^T
\]

We may then construct the results summarized in Table 7.1 for the four different input patterns of interest. Accordingly, the input patterns are mapped onto the \( \varphi_1-\varphi_2 \) plane as shown in Fig. 7.2b. Here we now see that the input patterns \((0,1)\) and \((1,0)\) are indeed linearly separable from the remaining input patterns \((1,1)\) and \((0,0)\). Thereafter, the XOR problem may be readily solved by using the functions \( \varphi_1(x) \) and \( \varphi_2(x) \) as the inputs to a linear classifier such as the perceptron.

In this example there is no increase in the dimensionality of the hidden-unit space compared to the input space. In other words, nonlinearity exemplified by the use of Gaussian hidden functions is sufficient to transform the XOR problem into a linearly separable one.
TABLE 7.1 Specification of the Hidden Functions for the XOR Problem of Example 1

<table>
<thead>
<tr>
<th>Input Pattern, x</th>
<th>First Hidden Function, ( \varphi_1(x) )</th>
<th>Second Hidden Function, ( \varphi_2(x) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>(1,1)</td>
<td>1</td>
<td>0.1355</td>
</tr>
<tr>
<td>(0,1)</td>
<td>0.3678</td>
<td>0.3678</td>
</tr>
<tr>
<td>(0,0)</td>
<td>0.1353</td>
<td>0.3678</td>
</tr>
<tr>
<td>(1,0)</td>
<td>0.3678</td>
<td>0.3678</td>
</tr>
</tbody>
</table>

7.3 Interpolation Problem

The important point that emerges from Cover’s theorem on the separability of patterns is that in solving a nonlinearly separable pattern-classification problem, there is, in general, practical benefit to be gained in mapping the input space into a new space of high enough dimension. Basically, a nonlinear mapping is used to transform a nonlinearly separable classification problem into a linearly separable one. In a similar fashion, we may use a nonlinear mapping to transform a difficult nonlinear filtering problem into an easier one that involves linear filtering.

Consider then a feedforward network with an input layer, a single hidden layer, and an output layer consisting of a single unit. We have purposely chosen a single output unit to simplify the exposition without loss of generality. The network is designed to perform a nonlinear mapping from the input space to the hidden space, followed by a linear mapping from the hidden space to the output space. Let \( p \) denote the dimension of the input space. Then, in an overall fashion, the network represents a map from the \( p \)-dimensional input space to the single-dimensional output space, written as

\[
s: \mathbb{R}^p \rightarrow \mathbb{R}^1
\]  

We may think of the map \( s \) as a hypersurface (graph) \( \Gamma \subset \mathbb{R}^{p+1} \), just as we think of the elementary map \( s: \mathbb{R}^1 \rightarrow \mathbb{R}^1 \), where \( s(x) = x^2 \), as a parabola drawn in \( \mathbb{R}^2 \) space. The surface \( \Gamma \) is a multidimensional plot of the output as a function of the input. In a practical situation, the surface \( \Gamma \) is unknown and the training data are usually contaminated with noise. Accordingly, the training phase and generalization phase of the learning process may be respectively viewed as follows (Broomhead and Lowe, 1988):

- The training phase constitutes the optimization of a fitting procedure for the surface \( \Gamma \), based on known data points presented to the network in the form of input–output examples (patterns).
- The generalization phase is synonymous with interpolation between the data points, with the interpolation being performed along the constrained surface generated by the fitting procedure as the optimum approximation to the true surface \( \Gamma \).

Thus we are led to the theory of multivariable interpolation in high-dimensional space, which has a long history (Davis, 1963). The interpolation problem, in its strict sense, may be stated as follows:

Given a set of \( N \) different points \( \{x_i \in \mathbb{R}^p | i = 1, 2, \ldots, N \} \) and a corresponding set of \( N \) real numbers \( \{d_i \in \mathbb{R} | i = 1, 2, \ldots, N \} \), find a function \( F: \mathbb{R}^N \rightarrow \mathbb{R}^1 \) that satisfies the interpolation condition:

\[
F(x_i) = d_i, \quad i = 1, 2, \ldots, N
\]

(7.14)
Note that for strict interpolation as specified here, the interpolating surface (i.e., function \( F \)) is constrained to pass through all the training data points.

The radial-basis functions (RBF) technique consists of choosing a function \( F \) that has the following form (Powell, 1988):

\[
F(x) = \sum_{i=1}^{N} w_i \varphi(||x - x_i||)
\]  

(7.15)

where \( \{\varphi(||x - x_i||)\}_{i=1}^{N} \) is a set of N arbitrary (generally nonlinear) functions, known as radial-basis functions, and \( ||\cdot|| \) denotes a norm that is usually taken to be Euclidean. The known data points \( x_i \in \mathbb{R}^p, i = 1, 2, \ldots, N \) are taken to be the centers of the radial-basis functions.

Inserting the interpolation conditions of Eq. (7.14) in (7.15), we obtain the following set of simultaneous linear equations for the unknown coefficients (weights) of the expansion \( \{w_i\} \):

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

(7.16)

where

\[
\varphi_{ji} = \varphi(||x_j - x_i||), \quad j, i = 1, 2, \ldots, N
\]  

(7.17)

Let

\[
d = [d_1, d_2, \ldots, d_N]^T
\]

(7.18)

\[
w = [w_1, w_2, \ldots, w_N]^T
\]

(7.19)

The N-by-1 vectors \( d \) and \( w \) represent the desired response vector and linear weight vector, respectively. Let \( \Phi \) denote an N-by-N matrix with elements \( \varphi_{ji} \):

\[
\Phi = \{\varphi_{ji} | j, i = 1, 2, \ldots, N\}
\]

(7.20)

We call this matrix the interpolation matrix. We may then rewrite Eq. (7.16) in the compact form

\[
\Phi w = x
\]

(7.21)

There is a class of radial-basis functions that enjoys the following remarkable property (Light, 1992):

*Let \( x_1, x_2, \ldots, x_N \) be distinct points in \( \mathbb{R}^p \). Then the N-by-N interpolation matrix \( \Phi \) whose jth element is \( \varphi_{ji} = \varphi(||x_j - x_i||) \) is positive definite.*

This theorem is more powerful than a previous result due to Micchelli (1986) stating that the interpolation matrix \( \Phi \) is nonsingular. Light’s theorem applies to the following cases (among others):

1. **Inverse multiquadrics** (Hardy, 1971)

\[
\varphi(r) = \frac{1}{(r^2 + c^2)^{1/2}} \quad \text{for some } c > 0, \text{ and } r \geq 0
\]

(7.22)
2. Gaussian functions

\[ \varphi(r) = \exp \left( -\frac{r^2}{2\sigma^2} \right) \quad \text{for } \sigma > 0, \text{ and } r \geq 0 \]  \hspace{1cm} (7.23)

Theoretical investigations and practical results, however, seem to show that the type of nonlinearity \( \varphi(\cdot) \) is not crucial to the performance of RBF networks (Powell, 1988).

Returning to the implication of Light’s theorem, we note that, provided that the data points are all distinct, the interpolation matrix \( \Phi \) is positive definite, and so we may solve Eq. (7.21) for the weight vector \( w \), obtaining

\[ w = \Phi^{-1}d \]  \hspace{1cm} (7.24)

where \( \Phi^{-1} \) is the inverse of the interpolation matrix \( \Phi \). Although in theory we are always assured a solution to the strict interpolation problem, in practice we cannot solve Eq. (7.21) when the matrix \( \Phi \) is arbitrarily close to singular. At this point, regularization theory can help by perturbing the matrix \( \Phi \) to \( \Phi + \Xi \), as we shall see in Section 7.5.

7.4 Supervised Learning as an Ill-Posed Hypersurface Reconstruction Problem

The strict interpolation procedure as described here may not be a good strategy 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 we are constrained to have as many radial-basis functions as data points, the problem is overdetermined. Consequently, the network may end up fitting misleading variations due to idiosyncrasies or noise in the input data, thereby resulting in a degraded generalization performance (Broomhead and Lowe, 1988).

To develop a deep understanding of the overfitting problem described here and how to cure it, we first go back to the viewpoint that the design of a neural network (more precisely, an associative memory) trained to retrieve 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. In other words, \textit{learning is viewed as a problem of hypersurface reconstruction, given a set of data points that may be sparse.} According to this viewpoint, the hypersurface reconstruction or approximation problem belongs to a generic class of problems called \textit{inverse problems}.

An inverse problem may 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. To explain what we mean by this terminology, 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 \textit{well posed} if three conditions are satisfied (Morozov, 1993; Tikhonov and Arsenin, 1977):

1. \textit{Existence}. For every input vector \( x \in X \), there does exist an output \( y = F(x) \), where \( y \in Y \).
2. \textit{Uniqueness}. For any pair of input vectors \( x, t \in X \), we have \( F(x) = F(t) \) if, and only if, \( x = t \).
3. \textit{Continuity}. The mapping is continuous, that is, for any \( \varepsilon > 0 \) there exists \( \delta = \delta(\varepsilon) \) such that the condition \( \rho_0(x,t) < \delta \) implies that \( \rho_0(F(x),F(t)) < \varepsilon \), where \( \rho(\cdot;\cdot) \) is the symbol for distance between the two arguments in their respective spaces. This criterion is illustrated in Fig. 7.3.

If these conditions are not satisfied, the inverse problem is said to be ill-posed.
Learning, viewed as a hypersurface reconstruction problem, is an ill-posed inverse problem for the following reasons. First, there is not as much information in the training data as we really need to reconstruct the input–output mapping uniquely, hence the uniqueness criterion is violated. Second, the presence of noise or imprecision in the input data adds uncertainty to the reconstructed input–output mapping. In particular, if the noise level in the input is too high, it is possible for the neural network to produce an output outside of the range $Y$ for a specified input $x$ in the domain $X$; in other words, the continuity criterion is violated. To make the learning problem well posed so that generalization to new data is feasible, some form of prior information about the input-output mapping is needed (Poggio and Girosi, 1990a). This, in turn, means that the process responsible for the generation of input–output examples used to train a neural network must exhibit redundancy in an information-theoretic sense. This requirement is indeed satisfied by the physical processes with which we have to deal in practice (e.g., speech, pictures, radar signals, sonar signals, seismic data), which are all redundant by their very nature. Furthermore, for a physical process, the generator of the data is usually smooth. Indeed, so long as the data generation is smooth, then small changes in the input can give rise to large changes in the output and still be approximated adequately (Lowe, 1992); this point is well illustrated by a chaotic map. The important point to note here is that the smoothness of data generation is a basic form of functional redundancy.

7.5 Regularization Theory

In 1963, Tikhonov proposed a new method called regularization for solving ill-posed problems. In the context of approximation problems, the basic idea of regularization is to stabilize the solution by means of some auxiliary nonnegative functional that embeds prior information, e.g., smoothness constraints on the input–output mapping (i.e., solution to the approximation problem), and thereby make an ill-posed problem into a well-posed one (Poggio and Girosi, 1990a).

To be specific, let the set of input–output data available for approximation be described by

$$\text{Input signal: } x_i \in \mathbb{R}^p, \quad i = 1, 2, \ldots, N \quad (7.25)$$

$$\text{Desired response: } d_i \in \mathbb{R}^l, \quad i = 1, 2, \ldots, N \quad (7.26)$$

Note that the output is assumed to be one-dimensional; this assumption does not in any way limit the general applicability of the regularization theory being developed here. Let the approximating function be denoted by $F(x)$, where (for convenience of presentation)

---

1 Regularization theory is discussed in book form by Tikhonov and Arsenin (1977), and Morozov (1993).
we have omitted the weight vector $\mathbf{w}$ of the network from the argument of the function $F$. According to Tikhonov’s regularization theory, the function $F$ is determined by minimizing a costfunctional $\mathcal{C}(F)$, so-called because it maps functions (in some suitable function space) to the real line. It involves two terms.

1. **Standard Error Term.** This first term, denoted by $\mathcal{E}_s(F)$, measures the standard error (distance) between the desired (target) response $d_i$ and the actual response $y_i$ for training example $i = 1, 2, \ldots, N$. Specifically, we define

$$\mathcal{E}_s(F) = \frac{1}{2} \sum_{i=1}^{N} (d_i - y_i)^2$$

$$= \frac{1}{2} \sum_{i=1}^{N} [d_i - F(x_i)]^2$$

(7.27)

where we have introduced the scaling factor $\frac{1}{2}$ for the sake of consistency with material presented in previous chapters.

2. **Regularizing Term.** This second term, denoted by $\mathcal{E}_r(F)$, depends on the geometric properties of the approximating function $F(x)$. Specifically, we write

$$\mathcal{E}_r(F) = \frac{1}{2} \| PF \|^2$$

(7.28)

where $P$ is a linear (pseudo) differential operator. Prior information about the form of the solution (i.e., the function $F(x)$) is embedded in the operator $P$, which naturally makes the selection of $P$ problem-dependent. We refer to $P$ as a stabilizer in the sense that it stabilizes the solution $F$, making it smooth and therefore continuous. Note, however, that smoothness implies continuity, but the reverse is not necessarily true.

The analytic approach used for the situation described here draws a strong analogy between linear differential operators and matrices, thereby placing both types of models in the same conceptual framework. Thus, the symbol $\| \|$ in Eq. (7.28) denotes a norm imposed on the function space to which $PF$ belongs. By *function space* we mean a normed vector space of functions. Ordinarily, the function space used here is the $L^2$ space that consists of all real-valued functions $f(x)$, $x \in \mathbb{R}^p$, for which $|f(x)|^2$ is Lebesgue integrable. The function $f(x)$ denotes the actual function that defines the underlying physical process responsible for the generation of the input–output pairs $(x_1, d_1), (x_2, d_2), \ldots, (x_N, d_N)$. Strictly speaking, however, we require the function $f(x)$ to be a member of a *reproducing kernel Hilbert space (RKHS)* with a reproducing kernel in the form of the Dirac delta distribution $\delta$ (Tapia and Thompson, 1978). We need to do this because, further on in the derivations, we require that the Dirac delta distribution $\delta$ to be in the dual of the function space. The simplest RKHS satisfying our needs is the space of rapidly decreasing, infinitely continuously differentiable functions, that is, the classical space $S$ of rapidly decreasing test functions for the *Schwarz theory of distributions*, with finite $P$-induced norm, as shown by

$$H_p = \{ f \in S; \| Pf \| < \infty \}$$

(7.29)

Generally speaking, engineers tend to think of only the $L^2$ space whenever Hilbert space is mentioned, perhaps on the grounds that $L^2$ space is isomorphic to any Hilbert space. But the norm is the most important feature of a Hilbert space, and *isometries* (i.e., norm-preserving isomorphism) are more important than simply additive isomorphism (Kailath, 1974). The theory of RKHS shows that there are many other different and quite useful Hilbert spaces besides the $L^2$ space. For a tutorial review of RKHS, see Kailath (1971).
where the norm of \( Pf \) is taken with respect to the range of \( P \), assumed to be another Hilbert space. By a “rapidly decreasing” function \( \varphi \) we mean one that satisfies the condition

\[
\lim_{|x| \to \infty} |x^\alpha \varphi(x)| = 0
\]

for all pairs of multi-indices \( \alpha \) and \( \beta \). In what follows we shall refer to the \( H_p \) of Eq. (7.29) simply as \( H \) when the operator \( P \) is clear from context.

The total cost functional to be minimized is

\[
\mathcal{E}(F) = \mathcal{E}_s(F) + \lambda \mathcal{E}_c(F)
\]

\[
= \frac{1}{2} \sum_{i=1}^{N} [d_i - F(x_i)]^2 + \frac{1}{2} \lambda \| PF \|^2
\]

(7.30)

where \( \lambda \) is a positive real number called the regularizing parameter.

In a sense, we may view the regularization parameter \( \lambda \) as an indicator of the sufficiency of the given data set as examples that specify the solution \( F(x) \). In particular, the limiting case \( \lambda \to 0 \) implies that the problem is unconstrained, with the solution \( F(x) \) being completely determined from the examples. The other limiting case, \( \lambda \to \infty \), on the other hand, implies that the a priori smoothness constraint is by itself sufficient to specify the solution \( F(x) \), which is another way of saying that the examples are unreliable. In practical applications, the regularization parameter \( \lambda \) is assigned a value somewhere between these two limiting conditions, so that both the sample data and the a priori information contribute to the solution \( F(x) \). Thus, the regularizing term \( \mathcal{E}_c(F) \) represents a model complexity-penalty function, the influence of which on the final solution is controlled by the regularization parameter \( \lambda \).

As we remarked previously in Section 6.17, regularization theory is closely linked with the model order-selection problem in statistics, which is exemplified by the minimum description length (MDL) criterion (Rissanen, 1978) and an information-theoretic criterion (AIC) (Akaike, 1973).

**Solution to the Regularization Problem**

The principle of regularization may now be stated as follows:

*Find the function \( F(x) \) that minimizes the cost functional \( \mathcal{E}(F) \), defined by

\[
\mathcal{E}(F) = \mathcal{E}_s(F) + \lambda \mathcal{E}_c(F)
\]

where \( \mathcal{E}_s(F) \) is the standard error term, \( \mathcal{E}_c(F) \) is the regularizing term, and \( \lambda \) is the regularization parameter.*

To proceed with the minimization of the cost functional \( \mathcal{E}(F) \), we need a rule for evaluating the differential of \( \mathcal{E}(F) \). We can take care of this matter by using the Fréchet differential, which for this functional is defined by (Dorny, 1975; de Figueiredo and Chen, 1993):

\[
d\mathcal{E}(F,h) = \left[ \frac{d}{d \beta} \mathcal{E}(F + \beta h) \right]_{\beta=0}
\]

(7.31)

\(^3\) A multi-index \( \alpha = (\alpha_0, \alpha_1, \ldots, \alpha_n) \) of order \( |\alpha| = \sum_{i=1}^{n} \alpha_i \) is a set of whole numbers used to abbreviate the following notations (Al-Gwaiz, 1992):

1. \( x^\alpha = x_0^{\alpha_0} x_1^{\alpha_1} \cdots x_n^{\alpha_n} \quad \text{for } x \in \mathbb{R}^n \)
2. \( \frac{\partial f}{\partial x_i} x_j^{\alpha_i} \cdots \partial x_{\alpha_n} \quad \text{for } f: \mathbb{R}^n \to \mathbb{R} \)
where \( h(x) \) is a fixed function of the vector \( x \). In Eq. (7.31), the ordinary rules of differentiation are used. A necessary condition for the function \( F(x) \) to be a relative extremum of the functional \( \mathcal{E}(F) \) is that the Fréchet differential \( d\mathcal{E}(F,h) \) be zero at \( F(x) \) for all \( h \in H \), as shown by

\[
d\mathcal{E}(F,h) = d\mathcal{E}_s(F,h) + A d\mathcal{E}_s(F,h) = 0 \quad (7.32)
\]

where \( d\mathcal{E}_s(F,h) \) and \( d\mathcal{E}_s(F,h) \) are the Fréchet differentials of the functionals \( \mathcal{E}_s(F) \) and \( \mathcal{E}_s(F) \), respectively.

Evaluating the Fréchet differential of the standard error term \( \mathcal{E}_s(F,h) \) of Eq. (7.27), we have

\[
d\mathcal{E}_s(F,h) = \left[ \frac{d}{d\beta} \mathcal{E}_s(F + \beta h) \right]_{\beta=0}
\]

\[
= \left[ \frac{1}{2} \frac{d}{d\beta} \sum_{i=1}^{N} \left[ d_i - F(x_i) - \beta h(x_i) \right]^2 \right]_{\beta=0}
\]

\[
= -\sum_{i=1}^{N} \left[ d_i - F(x_i) - \beta h(x_i) \right] h(x_i)_{\beta=0}
\]

\[
= -\sum_{i=1}^{N} [d_i - F(x_i)] h(x_i) \quad (7.33)
\]

At this point in the discussion, we find it instructive to invoke the Riesz representation theorem, which may be stated as follows (Debnath and Mikusiński, 1990):

Let \( f \) be a bounded linear functional in a general Hilbert space denoted by \( H \). There exists one \( h_0 \in H \) such that

\[
f = (h, h_0)_H \quad \text{for all } h \in H
\]

Moreover, we have

\[
\|f\|_{H^*} = \|h_0\|_H
\]

where \( H^* \) is the dual or conjugate of the Hilbert space \( H \).

The symbol \( (\cdot, \cdot)_H \) used here refers to the inner product in \( H \) space. Hence, in light of the Riesz representation theorem, we may rewrite the Fréchet differential \( d\mathcal{E}_s(F,h) \) of Eq. (7.33) in the equivalent form

\[
d\mathcal{E}_s(F,h) = -\left( h, \sum_{i=1}^{N} (d_i - F) \delta_{x_i} \right)_H \quad (7.34)
\]

where \( \delta_{x_i} \) denotes the Dirac delta distribution centered at \( x_i \); that is,

\[
\delta_{x_i}(x) = \delta(x - x_i) \quad (7.35)
\]

Consider next the evaluation of the Fréchet differential of the regularizing term \( \mathcal{E}_c(F) \) of Eq. (7.28), where the norm is defined in accordance with Eq. (7.29). Thus, proceeding in a manner similar to that described above, we have
Using the definition of an adjoint differential operator, we may equivalently write
\[
d^\varepsilon_c(F, h) = (h, P^* P F)_H
\] (7.37)
where \( P^* \) is the adjoint of the differential operator \( P \). In a loose sense, taking adjoints is similar to conjugation of complex numbers.

Returning to the extremum condition described in Eq. (7.32) and substituting the Fréchet differentials of Eqs. (7.34) and (7.37) in that equation, we may now make the following statements:

- The Fréchet differential \( d^\varepsilon_c(F, h) \) is
\[
d^\varepsilon_c(F, h) = 2 \left[ h_i P^* P F - \frac{1}{\lambda} \sum_{i=1}^{N} (d_i - F) \delta_i \right] (7.38)
\]

- Since the regularization parameter \( \lambda \) is ordinarily assigned a value somewhere in the open interval \((0, \infty)\), the Fréchet differential \( d^\varepsilon_c(F, h) \) is zero for every \( h(x) \) in \( H \) space if and only if the following condition is satisfied in the distributional sense:
\[
P^* P F - \frac{1}{\lambda} \sum_{i=1}^{N} (d_i - F(x)) \delta_i = 0
\] (7.39)
or, equivalently,
\[
P^* P F(x) = \frac{1}{\lambda} \sum_{i=1}^{N} [d_i - F(x_i)] \delta(x - x_i)
\] (7.40)

Equation (7.40) is referred to as the Euler–Lagrange equation for the cost functional \( \varepsilon_c(F) \) defined in Eq. (7.30) (Poggio and Girosi, 1990a).

EXAMPLE 2. Spline Functions

Consider the simple example of one-dimensional data, for which the differential operator \( P \) is defined by
\[
\|PF\|^2 = \int_{\mathbb{R}} \left[ \frac{d^2 F(x)}{dx^2} \right]^2 dx
\]
In this case, the function \( F(x) \) that minimizes the cost functional of Eq. (7.30) is a cubic spline. Spline functions are examples of piecewise polynomial approximators (Schumaker, 1981). The basic idea behind the method of splines is as follows. An approximation region of interest is broken up into a finite number of subregions via the use of knots;
the knots can be fixed, in which case the approximators are \textit{linearly} parameterized, or they can be variable, in which case the approximators are \textit{nonlinearly} parameterized. In both cases, in each region of the approximation a polynomial of degree at most \( n \) is used, with the additional requirement that the overall function be \( n - 1 \) times differentiable. Among the spline functions used in practice, \textit{cubic splines} (for which \( n = 3 \)) are the most popular; here, the overall function must be continuous with second-order derivatives at the knots.

\section*{Green's Functions}

Equation (7.40) represents a partial pseudodifferential equation in \( F \). The solution of this equation is known to consist of the integral transformation of the right-hand side of the equation with a kernel given by the \textit{influence function} or \textit{Green’s function} for the self-adjoint differential operator \( P^*P \) (Poggio and Girosi, 1990a; Courant and Hilbert, 1970; Dorny, 1975). The Green’s function plays the same role for a linear differential equation as does the inverse matrix for a matrix equation.

Let \( G(x; x_i) \) denote the Green’s function centered at \( x_i \). By definition, the Green’s function \( G(x; x_i) \) satisfies the partial differential equation (taken in the sense of distributions):

\[ P^*P G(x; x_i) = \delta(x - x_i) \]

where, as defined previously, \( \delta(x - x_i) \) is a delta function located at \( x = x_i \).

The solution \( F(x) \) for the partial differential equation (7.40) may now be expressed in the form of a multiple integral transformation as follows (Courant and Hilbert, 1970):

\[ F(x) = \int_{\mathbb{R}^d} G(x; \xi) \varphi(\xi) d\xi \]

(7.42)

where the function \( \varphi(\xi) \) denotes the right-hand side of Eq. (7.40) with \( x \) replaced by \( \xi \), that is,

\[ \varphi(\xi) = \frac{1}{\lambda} \sum_{i=1}^{N} [d_i - F(x_i)] \delta(\xi - x_i) \]

(7.43)

Substituting Eq. (7.43) in (7.42), interchanging the order of summation and integration, and then using the \textit{sifting property} of a delta function, we get the desired result (Girosi and Poggio, 1990a):

\[ F(x) = \frac{1}{\lambda} \sum_{i=1}^{N} [d_i - F(x_i)] G(x; x_i) \]

(7.44)

Equation (7.44) states that the minimizing solution \( F(x) \) to the regularization problem is a linear superposition of \( N \) Green’s functions. The \( x_i \) represent the \textit{centers of the expansion}, and the weights \( [d_i - F(x_i)]/\lambda \) represent the \textit{coefficients of the expansion}. In other words, the solution to the regularization problem lies in an \( N \)-dimensional subspace of the space of smooth functions, and the set of Green’s functions \( \{G(x; x_i)\} \) centered at \( x_i, i = 1, 2, \ldots, N \), constitutes a basis for this subspace.
The next issue to be resolved is the determination of the unknown coefficients in the expansion of Eq. (7.44). Let

\[ w_i = \frac{1}{\lambda} \left[ d_i - F(x_i) \right], \quad i = 1, 2, \ldots, N \]  

(7.45)

Accordingly, we may recast the minimizing solution of Eq. (7.44) simply as follows:

\[ F(x) = \sum_{i=1}^{N} w_i G(x; x_i) \]  

(7.46)

Evaluating Eq. (7.46) at \( x_j, j = 1, 2, \ldots, N \), we get

\[ F(x_j) = \sum_{i=1}^{N} w_i G(x_j; x_i), \quad j = 1, 2, \ldots, N \]  

(7.47)

We now introduce the following definitions:

\[ F = [F(x_1), F(x_2), \ldots, F(x_N)]^T \]  

(7.48)

\[ d = [d_1, d_2, \ldots, d_N]^T \]  

(7.49)

\[ G = \begin{bmatrix} G(x_1; x_1) & G(x_1; x_2) & \cdots & G(x_1; x_N) \\ G(x_2; x_1) & G(x_2; x_2) & \cdots & G(x_2; x_N) \\ \vdots & \vdots & \ddots & \vdots \\ G(x_N; x_1) & G(x_N; x_2) & \cdots & G(x_N; x_N) \end{bmatrix} \]  

(7.50)

\[ w = [w_1, w_2, \ldots, w_N]^T \]  

(7.51)

Then we may rewrite Eqs. (7.45) and (7.47) in matrix form as follows, respectively:

\[ w = \frac{1}{\lambda} (d - F) \]  

(7.52)

and

\[ F = Gw \]  

(7.53)

Eliminating \( F \) between Eqs. (7.52) and (7.53) and rearranging terms, we get

\[ (G + \lambda I)w = d \]  

(7.54)

where \( I \) is the N-by-N identity matrix. We call the matrix \( G \) the Green’s matrix.

Since the combined operator \( P^*P \) in Eq. (7.40) is self-adjoint, it follows that the associated Green’s function \( G(x; x_i) \) is a symmetric function, as shown by (Courant and Hilbert, 1970)

\[ G(x_i; x_j) = G(x_j; x_i) \quad \text{for all } i \text{ and } j \]  

(7.55)

Equivalently, the Green’s matrix \( G \) defined in Eq. (7.50) is a symmetric matrix; that is,

\[ G^T = G \]  

(7.56)

where the superscript \( T \) denotes matrix transposition. We now invoke Light’s theorem, which was described in Section 7.3 in the context of the interpolation matrix \( \Phi \). We first note that Green’s matrix \( G \) plays a role in regularization theory similar to that of \( \Phi \) in RBF interpolation theory. Both \( G \) and \( \Phi \) are N-by-N symmetric matrices. Accordingly, we may state that the matrix \( G \), for certain classes of Green’s functions, is positive definite.
provided that the data points \( x_1, x_2, \ldots, x_N \) are distinct. The classes of Green’s functions covered by Light’s theorem include multiquadrics and Gaussian functions. In practice, we may always choose \( \lambda \) sufficiently large to ensure that \( G + \lambda I \) is positive definite and, therefore, invertible. This, in turn, means that the linear system of equations (7.54) will have a unique solution given by (Poggio and Girosi, 1990a)

\[
w = (G + \lambda I)^{-1}d
\]  

(7.57)

Thus, having selected the pseudodifferential operator \( P \) and therefore identified the associated Green’s function \( G(x_j; x_i) \), where \( i = 1, 2, \ldots, N \), we may use Eq. (7.57) to obtain the weight vector \( w \) for a specified desired response vector \( d \) and an appropriate value of regularization parameter \( \lambda \).

In conclusion, we may state that the solution to the regularization problem is given by the expansion

\[
F(x) = \sum_{i=1}^{N} w_i G(x; x_i)
\]  

(7.58)

where \( G(x; x_j) \) is the Green’s function for the self-adjoint differential operator \( P^*P \), and \( w_i \) is the \( i \)th element of the weight vector \( w \); these two quantities are themselves defined by Eq. (7.41) and (7.57), respectively. Equation (7.58) states the following (Poggio and Girosi, 1990a):

- The regularization approach is equivalent to the expansion of the solution in terms of a set of Green’s functions, whose characterization depends only on the form adopted for the stabilizer \( P \) and the associated boundary conditions.

- The number of Green’s functions used in the expansion is equal to the number of examples used in the training process.

It should be noted, however, that the solution of the regularization problem given in Eq. (7.58) is incomplete, as it represents a solution modulo a term \( g(x) \) that lies in the null space of the operator \( P \) (Poggio and Girosi, 1990a). We say this because all the functions that lie in the null space of \( P \) are indeed “invisible” to the smoothing term \( \|PF\|^2 \) in the cost functional \( E(F) \) of Eq. (7.30); by the null space of \( P \), we mean the set of all functions \( g(x) \) for which \( P^*P g \) is zero. The exact form of the additional term \( g(x) \) is problem-dependent in the sense that it depends on the stabilizer chosen and the boundary conditions of the problem at hand. For example, it is not needed in the case of a stabilizer \( P \) corresponding to a bell-shaped Green’s function such as a Gaussian or inverse multiquadric. For this reason, and since its inclusion does not modify the main conclusions, we will disregard it in the sequel.

The characterization of the Green’s function \( G(x; x_j) \), for a specified center \( x_j \), depends only on the form of the stabilizer \( P \), that is, on the a priori assumption made concerning the input–output mapping. If the stabilizer \( P \) is translationally invariant, then the Green’s function \( G(x; x_j) \) centered at \( x_j \) will depend only on the Euclidean norm of the difference vector \( x - x_j \), as shown by

\[
G(x; x_j) = G(\|x - x_j\|)
\]  

(7.60)
Under these conditions, the Green’s function must be a **radial-basis function**. In such a case, the regularized solution of Eq. (7.58) takes on the following special form (Poggio and Girosi, 1990a):

\[
F(x) = \sum_{i=1}^{N} w_i G(||x - x_i||)
\]

(7.61)

The solution described in Eq. (7.61) constructs a linear function space that depends on the known data points according to the Euclidean distance measure.

The solution described by Eq. (7.61) is termed **strict interpolation**, since all the N data points available for training are used to generate the interpolating function \( F(x) \). It is important, however, to realize that this solution differs from that of Eq. (7.15) in a fundamental respect: The solution of Eq. (7.61) is **regularized** by virtue of the definition given in Eq. (7.57) for the weight vector \( w \). It is only when we set the regularization parameter \( \lambda \) equal to zero that the two solutions may become one and the same.

**Multivariate Gaussian Functions**

A special class of differential operators \( \mathbf{P} \) that are invariant under both rotations and translations is defined by (Poggio and Girosi, 1990a):

\[
||\mathbf{P}F||^2 = \sum_{i=0}^{K} a_i ||D^i F(x)||^2
\]

(7.62)

where, using standard multi-index notation (see footnote 3), the norm of the differential operator \( D^i \) is defined as

\[
||D^i F||^2 = \sum_{|\alpha|=i} \int_{\mathbb{R}^p} |\partial^\alpha F(x)|^2 \, dx
\]

(7.63)

The integral in Eq. (7.63) is a multiple integral defined over the \( p \) coordinates of the input vector \( x \). The Green’s function \( G(x; x_i) \) associated with this differential operator satisfies the following differential equation (in the sense of distributions)

\[
\sum_{k=0}^{K} (-1)^k a_k \nabla^{2k} G(x; x_i) = \delta(x - x_i)
\]

(7.64)

where \( \nabla^{2k} \) is the k-iterated Laplacian operator in \( p \) dimensions, and

\[
\nabla^2 = \frac{\partial^2}{\partial x_1^2} + \frac{\partial^2}{\partial x_2^2} + \cdots + \frac{\partial^2}{\partial x_p^2}
\]

(7.65)

Specializing to the case of **Gaussian radial basis functions**, we permit the upper limit \( K \) in the summation of Eq. (7.64) to approach infinity, and define the coefficient \( a_k \) as

\[
a_k = \frac{\sigma_i^{2k}}{k!2^k}
\]

(7.66)

where \( \sigma_i \) is some constant associated with the data point \( x_i \). In such a case the Green’s function \( G(x; x_i) \) satisfies the pseudodifferential equation

\[
\sum_{k=0}^{\infty} (-1)^k \frac{\sigma_i^{2k}}{k!2^k} \nabla^{2k} G(x; x_i) = \delta(x - x_i)
\]

(7.67)

To solve Eq. (7.67) for the Green’s function \( G(x; x_i) \), we use the multidimensional Fourier transform. To simplify matters, we put \( \mathbf{x}' = 0 \) and \( \sigma_i^2 = 1 \); their effects will be accounted
for later. We may then reduce Eq. (7.67) to its basic form:

\[(P^*P)G(x) = \delta(x)\]  

(7.68)

where

\[P^*P = \sum_{k=0}^{\infty} \frac{(-1)^k}{k!2^k} \nabla^2_k\]  

(7.69)

The multidimensional Fourier transform of \(G(x)\) is defined by (Debnath and Mikusiński, 1990)

\[\hat{G}(s) = \int_{\mathbb{R}^p} G(x) \exp(-is^T x) \, dx\]  

(7.70)

where \(i = \sqrt{-1}\) and \(s\) is the \(p\)-dimensional transform variable; the notation \(i = \sqrt{-1}\) only applies to Eqs. (7.70) through (7.74). Then, recognizing that the multidimensional Fourier transform of the Dirac delta distribution \(\delta(x)\) is unity, and using the differentiation property of the multidimensional Fourier transform, we find that Eq. (7.68) is transformed as follows:

\[\sum_{k=0}^{\infty} \frac{1}{k!2^k} \left( \sum_{n=1}^{p} s_n^2 \right)^k \hat{G}(s) = 1\]  

(7.71)

where \(s_n\) is the \(n\)th element of the vector \(s\), and differentiation of \(G(x)\) with respect to the \(n\)th element of the vector \(x\) has the effect of multiplying the Fourier transform by \(is_n\). We next note that the infinite series in the transform variable \(s\) in Eq. (7.71) converges for all \(s \in \mathbb{R}^p\) to the following:

\[\sum_{k=0}^{\infty} \frac{1}{k!} \left( \frac{1}{2} s^2 s \right) = \exp \left( \frac{1}{2} s^2 s \right)\]  

(7.72)

Hence, substituting Eq. (7.72) in (7.71), and solving for the multidimensional Fourier transform \(\hat{G}(s)\), we get

\[\hat{G}(s) = \exp \left( -\frac{1}{2} s^2 s \right)\]  

(7.73)

Next, we use the inverse multidimensional Fourier transform to obtain

\[G(x) = \left( \frac{1}{2\pi} \right)^p \int_{\mathbb{R}^p} \hat{G}(s) \exp(is^T x) \, ds\]

\[= \left( \frac{1}{2\pi} \right)^p \int_{\mathbb{R}^p} \exp \left(-\frac{1}{2} s^2 s \right) \exp(is^T x) \, ds\]

\[= \prod_{n=1}^{p} \mathcal{F}^{-1} \left[ \exp \left(-\frac{1}{2} s_n^2 \right) \right]\]

\[= \prod_{n=1}^{p} \exp \left(-\frac{1}{2} s_n^2 \right)\]

\[= \exp \left(-\frac{1}{2} ||x||^2 \right)\]  

(7.74)

where the symbol \(\mathcal{F}^{-1}\), in the third line, denotes inverse Fourier transformation. Finally, we replace \(G(x)\) with \(G(x; \sigma) = G(||x - x||)\), and reintroduce the scaling factor \(\sigma^2\), the combined effort of which is to yield the following solution for Eq. (7.67):
Since the coefficient $a_0 = 1 > 0$ in Eq. (7.67), it follows that $G(x; x_i)$ is positive definite\footnote{A continuous function $f(t)$, defined on the interval $(0, \infty)$, is said to be \textit{positive definite} if, for any distinct points $x_1, x_2, \ldots, x_n \in \mathbb{R}^p$ and certain scalars $c_1, c_2, \ldots, c_n$, the quadratic form $\sum_{i=1}^{n} \sum_{j=1}^{n} c_i c_j f(||x_i - x_j||)$ is positive definite; the selection of the scalars is constrained by the order of positive definiteness of the function $f(t)$ (Poggio and Girosi, 1990a).} for all $i$. Hence the terms in the null space of the stabilizer $P$ are not necessary for the function $F(x)$ to minimize the functional $\mathcal{E}(F)$.

The Green's function $G(x; x_i)$ defined in Eq. (7.75) is recognized to be a \textit{multivariate Gaussian function} characterized by a mean vector $x_i$ and common variance $\sigma_i^2$, except for a scaling factor that may be absorbed in the weight $w_i$. Correspondingly, the regularized solution defined by Eq. (7.58) takes on the following special form:

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

(7.76)

which consists of a linear superposition of multivariate Gaussian basis functions with centers $x_i$ (located at the data points) and widths $\sigma_i$.

### 7.6 Regularization Networks

The expansion of the approximating function $F(x)$ given in Eq. (7.58) in terms of the Green's function $G(x; x_i)$ centered at $x_i$ suggests the network structure shown in Fig. 7.4 as a method for its implementation. For obvious reasons, this network is called a \textit{regularization network} (Poggio and Girosi, 1990a). It consists of three layers. The first layer is composed of input (source) nodes whose number is equal to the dimension $p$ of the input vector $x$ (i.e., the number of independent variables of the problem). The second layer is a hidden layer, composed of nonlinear units that are connected \textit{directly} to all of the nodes in the input layer. There is one hidden unit for each data point $x_i$, $i = 1, 2, \ldots, N$, where $N$ is the number of training examples. The activation functions of the individual hidden units are defined by the Green's functions. Accordingly, the output of the $i$th hidden unit is $G(x; x_i)$. The output layer consists of a single linear unit, being fully connected to the hidden layer. By “linearity” we mean that 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 expansion, defined in terms of the Green’s functions $G(x; x_i)$ and the regularization parameter $\lambda$ by Eq. (7.57). Figure 7.4 depicts the architecture of the regularization network for a single output. Clearly, such an architecture can be readily extended to accommodate any number of network outputs desired.

The regularization network shown in Fig. 7.4 assumes that the Green's function $G(x; x_i)$ is \textit{positive definite} for all $i$. Provided that this condition is satisfied, which it is in the case of the $G(x; x_i)$ having the Gaussian form given in Eq. (7.76), for example, then the solution produced by this network will be an “optimal” interpolant in the sense that it minimizes the functional $\mathcal{E}(F)$. Moreover, from the viewpoint of approximation theory, the regularization network has three desirable properties (Poggio and Girosi, 1990a):

1. The regularization network is a \textit{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 unknown 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 the training data.

7.7 Generalized Radial-Basis Function Networks

The one-to-one correspondence between the training input data \( x_i \) and the Green's function
\( G(x; x_i) \) for \( i = 1, 2, \ldots, N \) produces a regularization network that is prohibitively
expensive to implement in computational terms for large \( N \). Specifically, the computation
of the linear weights of the network [i.e., the coefficients of the expansion in Eq. (7.58)]
requires the inversion of an \( N \)-by-\( N \) matrix, which therefore grows polynomially with \( N \)
(roughly as \( N^3 \)). Furthermore, the likelihood of ill conditioning is higher for larger matrices;
the condition number of a matrix is defined as the ratio of the largest eigenvalue to the
smallest eigenvalue of the matrix. To overcome these computational difficulties, the
complexity of the network would have to be reduced, which requires an approximation
to the regularized solution, as discussed next.

The approach taken involves searching for a suboptimal solution in a lower-dimensional
space that approximates the regularized solution of Eq. (7.58). This is done by using a
standard technique known in variational problems as Galerkin's method. According to
this technique, the approximated solution \( F^*(x) \) is expanded on a finite basis, as shown
by (Poggio and Girosi, 1990a),

\[
F^*(x) = \sum_{i=1}^{M} w_i \varphi_i(x) \quad (7.77)
\]

where \( \{\varphi_i(x)\}_{i=1}^{M} \) is a new set of basis functions that we assume to be
linearly independent without loss of generality. Typically, the number of basis functions
is less than the number of data points (i.e., \( M \leq N \)), and the \( w_i \) constitute a new set of
weights. With radial-basis functions in mind, we set
where the set of centers \( \{ t_i | i = 1, 2, \ldots, M \} \) is to be determined. This particular choice of basis functions is the only one that guarantees that in the case of \( M = N \), and

\[
t_i = x_i, \quad i = 1, 2, \ldots, N
\]

the correct solution of Eq. (7.58) is consistently recovered. Thus, using Eq. (7.78) in (7.77), we may redefine \( F^*(x) \) as

\[
F^*(x) = \sum_{i=1}^{M} w_i G(x; t_i)
\]

\[
= \sum_{i=1}^{M} w_i G(\|x - t_i\|) \tag{7.79}
\]

Given the expansion of Eq. (7.79) for the approximating function \( F^*(x) \), the problem we have to address is the determination of the new set of weights \( \{ w_i | i = 1, 2, \ldots, M \} \) so as to minimize the new cost functional \( \mathcal{E}(F^*) \) defined by

\[
\mathcal{E}(F^*) = \sum_{i=1}^{N} \left( d_i - \sum_{j=1}^{M} w_j G(\|x_i - t_j\|) \right)^2 + \lambda \|PF^*\|^2 \tag{7.80}
\]

The first term on the right-hand side of Eq. (7.80) may be expressed as the squared Euclidean norm \( \|d - Gw\|^2 \), where

\[
d = [d_1, d_2, \ldots, d_N]^T
\]

\[
G = \begin{bmatrix}
G(x_1; t_1) & G(x_1; t_2) & \ldots & G(x_1; t_M) \\
G(x_2; t_1) & G(x_2; t_2) & \ldots & G(x_2; t_M) \\
\vdots & \vdots & \ddots & \vdots \\
G(x_N; t_1) & G(x_N; t_2) & \ldots & G(x_N; t_M)
\end{bmatrix}
\]

\[
w = [w_1, w_2, \ldots, w_M]^T \tag{7.83}
\]

The desired response vector \( d \) is \( N \)-dimensional as before. However, the matrix \( G \) of Green’s functions and the weight vector \( w \) have different dimensions; the matrix \( G \) is now \( N \)-by-\( M \) and therefore no longer symmetric, and the vector \( w \) is \( M \)-by-1. From Eq. (7.79) we note that the approximating function \( F^* \) is a linear combination of the Green’s functions for the stabilizer \( P \). Accordingly, we may express the second term on the right-hand side of Eq. (7.80) as

\[
\|PF^*\|^2 = (PF^*, PF^*)_H
\]

\[
= \left[ \sum_{i=1}^{M} w_i G(x; t_i), P^* P \sum_{j=1}^{M} w_j G(x; t_j) \right)_H
\]

\[
= \left[ \sum_{i=1}^{M} w_i G(x; t_i), \sum_{j=1}^{M} w_j \delta_i \right)_H
\]

\[
= \sum_{i=1}^{M} \sum_{j=1}^{M} w_j w_i G(t_j; t_i)
\]

\[
= w^T G_0 w \tag{7.84}
\]
where, in the second and third lines, we made use of the definition of an adjoint operator and Eq. (7.41), respectively. The matrix $G_0$ is a symmetric $M$-by-$M$ matrix, defined by

$$G_0 = \begin{bmatrix} G(t_1; t_1) & G(t_1; t_2) & \ldots & G(t_1; t_M) \\ G(t_2; t_1) & G(t_2; t_2) & \ldots & G(t_2; t_M) \\ \vdots & \vdots & \ddots & \vdots \\ G(t_M; t_1) & G(t_M; t_2) & \ldots & G(t_M; t_M) \end{bmatrix}$$

(7.85)

Thus the minimization of Eq. (7.80) with respect to the weight vector $w$ yields the result (see Problem 7.10)

$$(G^T G + \lambda G_0)w = G^T d$$

(7.86)

Note that as the regularization parameter $\lambda$ approaches zero, the weight vector $w$ converges to the pseudoinverse (minimum-norm) solution to the overdetermined least-squares data-fitting problem, as shown by (Broomhead and Lowe, 1988)

$$w = G^+ d, \quad \lambda = 0$$

(7.87)

where $G^+$ is the pseudoinverse of matrix $G$; that is,

$$G^+ = (G^T G)^{-1} G^T$$

(7.88)

### Weighted Norm

The norm in the approximate solution of Eq. (7.80) is ordinarily intended to be a Euclidean norm. When, however, the individual elements of the input vector $x$ belong to different classes, it is more appropriate to consider a general weighted norm, the squared form of which is defined as (Poggio and Girosi, 1990a)

$$\|x\|^2_C = (C x)^T (C x)$$

$$= x^T C^T C x$$

(7.89)

where $C$ is a $p$-by-$p$ norm weighting matrix, and $p$ is the dimension of the input vector $x$. Depending on how the weighting matrix $C$ is defined, we may identify three specific cases of interest.

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

$$\|x\|^2_C = \|x\|^2, \quad C = I$$

(7.90)

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

$$\|x\|^2_C = \sum_{k=1}^{p} c_k^2 x_k^2$$

(7.91)

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

3. The matrix $C$ is a nondiagonal matrix, in which case the weighted norm takes on a quadratic form as shown by
\[
\|x\|_C^2 = \sum_{l=1}^{p} \sum_{i=1}^{p} a_{kl} x_i x_l
\]

where \(a_{kl}\) is the \(kl\)th element of the matrix product \(C^T C\).

Using the definition of weighted norm, we may now rewrite the approximation to the regularized solution given in Eq. (7.80) in the more generalized form (Poggio and Girosi, 1990a; Lowe, 1989)

\[
F^*(x) = \sum_{i=1}^{M} w_i G(\|x - t_i\|_C)
\]

The use of a weighted norm may be interpreted in two ways. We may simply view it as applying an affine transformation to the original input space. In principle, allowing for such a transformation cannot degrade results from the default case, since it actually corresponds to an identity norm-weighting matrix. On the other hand, the weighted norm follows directly from a slight generalization of the \(p\)-dimensional Laplacian in the definition of the pseudodifferential operator \(P\) in Eq. (7.69); see Problem 7.11. In any event, the use of a weighted norm may also be justified in the context of Gaussian radial-basis functions on the following grounds. A Gaussian radial-basis function \(G(\|x - t_i\|_C)\) centered at \(t_i\) and with norm weighting matrix \(C_i\) may be expressed as

\[
G(\|x - t_i\|_C) = \exp[-(x - t_i)^T C_i^{-1} C_i (x - t_i)]
\]

where the inverse matrix \(C_i^{-1}\) is defined by

\[
\frac{1}{2} C_i^{-1} = C_i^T C_i
\]

Equation (7.94) represents a multivariate Gaussian distribution with mean vector \(t_i\) and covariance matrix \(C_i\). As such, it represents a generalization of the distribution described in Eq. (7.76).

The solution to the approximation problem given in Eq. (7.80) provides the framework for the generalized radial-basis function (RBF) network having the structure shown in Fig. 7.5. In this network, provision is made for a bias (i.e., data-independent variable) applied to the output unit. This is done simply by setting one of the linear weights in the output layer of the network equal to the bias and treating the associated radial-basis function as a constant equal to \(+1\). (The bias is the negative of a threshold.)

In structural terms, the generalized RBF network of Fig. 7.5 is similar to the regularization RBF network of Fig. 7.4. However, these two networks differ from each other in two important respects:

1. The number of nodes in the hidden layer of the generalized RBF network of Fig. 7.5 is \(M\), where \(M\) is ordinarily smaller than the number \(N\) of examples available for training. On the other hand, the number of hidden nodes in the regularization RBF network of Fig. 7.4 is exactly \(N\).

2. In the generalized RBF network of Fig. 7.5, 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 Fig. 7.4 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.

7.8 The XOR Problem (Revisited)

Consider again the XOR (Exclusive OR) problem, which we solved in Chapter 6 using a multilayer perceptron with a single hidden layer. Here we are going to present a solution of this same problem using an RBF network.

The RBF network to be investigated consists of a pair of Gaussian functions, defined as follows:

\[ G(||x - t_i||) = \exp(-||x - t_i||^2), \quad i = 1, 2 \]  

(7.96)

where the centers \( t_1 \) and \( t_2 \) are

\[ t_1 = [1,1]^T \]
\[ t_2 = [0,0]^T \]

For the characterization of the output unit, we assume the following:

1. The output unit uses weight sharing, which is justified by virtue of the symmetry of the problem; this is a form of prior information being built into the design of the network. With only two hidden units, we therefore only have a single weight \( w \) to be determined.

2. The output unit includes a bias \( b \) (i.e., data-independent variable). The significance of this bias is that the desired output values of the XOR function have nonzero mean.
Thus the structure of the RBF network proposed for solving the XOR problem is as shown in Fig. 7.6. The input–output relation of the network is defined by

\[
y(x) = \sum_{i=1}^{4} w_i g_i(\|x - t_i\|) + b
\]  

(7.97)

To fit the training data of Table 7.2, we require that

\[
y(x_j) = d_j, \quad j = 1, 2, 3, 4
\]  

(7.98)

where \(x_j\) is an input vector and \(d_j\) is the corresponding value of the desired output. Let

\[
g_{ji} = G(\|x_j - t_i\|), \quad j = 1, 2, 3, 4; i = 1, 2
\]  

(7.99)

Then, using the values of Table 7.2 in Eq. (7.99), we get the following set of equations written in matrix form:

\[
Gw = d
\]  

(7.100)

where

\[
G = \begin{bmatrix}
g_{11} & g_{12} & 1 
g_{21} & g_{22} & 1 
g_{31} & g_{32} & 1 
g_{41} & g_{42} & 1 
\end{bmatrix}
\]

\[
= \begin{bmatrix}
1 & 0.1353 & 1 
0.3678 & 0.3678 & 1 
0.1353 & 1 & 1 
0.3678 & 0.3678 & 1 
\end{bmatrix}
\]  

(7.101)

\[
d = [1 \quad 0 \quad 1 \quad 0]^T
\]  

(7.102)

\[
w = [w \quad w \quad b]^T
\]  

(7.103)

The problem described here is over determined in the sense that we have more data points than free parameters. This explains why the matrix \(G\) is not square. Consequently, no
unique inverse exists for the matrix $G$. To overcome this difficulty, we use the minimum-norm solution of Eq. (7.88), and so write

$$w = G'd$$

$$= (G^TG)^{-1}G't$$

Note that $G^TG$ is a square matrix with a unique inverse of its own. Substituting Eq. (7.101) in (7.104), we get

$$G^+ = \begin{bmatrix} 1.656 & -1.158 & 0.628 & -1.158 \\ 0.628 & -1.158 & 1.656 & -1.158 \\ -0.846 & 1.301 & -0.846 & 1.301 \end{bmatrix}$$

Finally, substituting Eqs. (7.102) and (7.105) in (7.104), we get

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

which completes the specification of the RBF network.

The last column of Table 7.2 includes the actual values of the RBF network output produced in response to the four input patterns. These results show that the RBF network of Fig. 7.6 successfully discriminates states of opposite polarity, and the XOR problem is therefore solved.

### 7.9 Comparison of RBF Networks and Multilayer Perceptrons

Radial-basis function (RBF) networks and multilayer perceptrons are examples of non-linear layered feedforward networks. They are both universal approximators. The universality of multilayer perceptrons was considered in Chapter 6. There also exists a formal proof for the universality of radial-basis function networks, as described by Park and Sandberg (1991). It is therefore not surprising to find that there always exists an RBF network capable of accurately mimicking a specified MLP, or vice versa. However, these two networks differ from each other in several important respects, as outlined here.

1. An RBF network (in its most basic form) has a single hidden layer, whereas an MLP may have one or more hidden layers.

---

5 An RBF network has been proposed by He and Lapedes (1991) that involves two hidden layers. The second hidden layer and output layer are both linear, performing successive approximations on the activation functions of the first hidden layer. See Problem 7.4 for more details.
2. Typically, the computation nodes of an MLP, be they located in a hidden or output layer, share a common neuron model. On the other hand, the computation nodes in the hidden layer of an RBF network are quite different and serve a different purpose from those in the output layer of the network.

3. The hidden layer of an RBF network is nonlinear, whereas the output layer is linear. On the other hand, the hidden and output layers of an MLP used as a classifier are usually all nonlinear; however, when the MLP is used to solve nonlinear regression problems, a linear layer for the output is usually the preferred choice.

4. The argument of the activation function of each hidden unit in an RBF network computes the Euclidean norm (distance) between the input vector and the center of that unit. On the other hand, the activation function of each hidden unit in an MLP computes the inner product of the input vector and the synaptic weight vector of that unit.

5. MLPs construct global approximations to nonlinear input–output mapping. Consequently, they are capable of generalization in regions of the input space where little or no training data are available. On the other hand, RBF networks using exponentially decaying localized nonlinearities (e.g., Gaussian functions) construct local approximations to nonlinear input–output mapping, with the result that these networks are capable of fast learning and reduced sensitivity to the order of presentation of training data. In many cases, however, we find that in order to represent a mapping to some desired degree of smoothness, the number of radial-basis functions required to span the input space adequately may have to be very large.

The linear characteristic of the output layer of the RBF network means that such a network is more closely related to Rosenblatt’s perceptron than the multilayer perceptron. However, the RBF network differs from the perceptron in that it is capable of implementing arbitrary nonlinear transformations of the input space. This is well illustrated by the XOR problem, which cannot be solved by any linear perceptron but can be solved by an RBF network.

### 7.10 Mixture Models

The expansion described in Eq. (7.93) is closely related to mixture models, that is, mixtures of Gaussian distributions. Mixtures of probability distributions, in particular Gaussian distributions, have been used extensively as models in a wide variety of applications where the data of interest arise from two or more populations mixed together in some varying properties (McLachlan and Basford, 1988). According to the mixture model, we have a superpopulation \( S \) that is a mixture of finite number—say, \( n \)—of populations \( S_1, S_2, \ldots, S_n \) in some proportions \( \pi_1, \pi_2, \ldots, \pi_n \), respectively, where

\[
\sum_{i=1}^{n} \pi_i = 1 \quad \text{and} \quad \pi_i \geq 0 \quad \text{for all } i \quad (7.106)
\]

The probability density function (pdf) of a random vector \( x \) in \( S \) is represented by

\[
f(x; \varphi) = \sum_{i=1}^{n} \pi_i f_i(x; \theta) \quad (7.107)
\]

---

6. Simard et al. (1992, 1993) have proposed a new distance measure for radial-basis functions, which can be made locally invariant to any set of transformations of the input and which can be computed efficiently. The new distance measure is called target distance; see Problem 1.16.

7. Lane et al. (1991) discuss a general neural network formulation that combines the generalization capability of global multilayer feedforward networks with the computational efficiency and learning speed of local networks.
where \( f_\theta(x; \mathbf{0}) \) is the pdf corresponding to \( S_i \) and \( \mathbf{0} \) represents the vector of all unknown associated parameters. The vector \( \mathbf{c} \) is made up of \( \mathbf{0} \) and the mixing proportions \( \pi_1, \pi_2, \ldots, \pi_n \). We are given a set of observation vectors \( \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_n \), say, and the problem is to estimate the unknown parameter vector \( \mathbf{c} \). We thus see that the mixture model problem is indeed similar to the problem of finding the \( \mathbf{w}, \mathbf{t}, \) and the \( S_i \) in the expansion of Eq. (7.93).

This analogy suggests that we may develop insight into the characterization of RBF networks from the literature on probability density function estimation. A particular paper that we have in mind here is that of Parzen (1962), who considered the estimation of a probability density function \( f(x) \), given a sequence of independent and identically distributed random variables with common probability density function \( f(x) \). Lowe (1991a) uses Parzen's basic exposition of the problem to derive the radial-basis function network. Lowe also points out that the traditional method of expectation-maximization (EM) (Dempster et al., 1977), based on maximizing the likelihood of the data given the model, may be used to optimize the positions and spreads of the centers in the hidden layer of an RBF network in an unsupervised manner; the EM algorithm is explained in Chapter 12.

### 7.11 Learning Strategies

The learning process undertaken by a radial-basis function (RBF) network may be visualized as follows. The linear weights associated with the output unit(s) of the network tend to evolve on a different “time scale” compared to the nonlinear activation functions of the hidden units. Thus, as the hidden layer’s activation functions evolve slowly in accordance with some nonlinear optimization strategy, the output layer’s weights adjust themselves rapidly through a linear optimization strategy. The important point to note here is that the different layers of an RBF network perform different tasks, and so it is reasonable to separate the optimization of the hidden and output layers of the network by using different techniques, and perhaps operating on different time scales (Lowe, 1991a).

There are different learning strategies that we can follow in the design of an RBF network, depending on how the centers of the radial basis functions of the network are specified. Essentially, we may identify three approaches, as discussed below.

#### 1. Fixed Centers Selected at Random

The simplest approach is to assume fixed radial-basis functions defining the activation functions of the hidden units. Specifically, the locations of the centers may be chosen randomly from the training data set. This is considered to be a “sensible” approach, provided that the training data are distributed in a representative manner for the problem at hand (Lowe, 1989). For the radial-basis functions themselves, we may employ an isotropic Gaussian function whose standard deviation is fixed according to the spread of the centers. Specifically, a (normalized) radial-basis function centered at \( t_i \) is defined as

\[
G(\|x - t_i\|) = \exp \left( -\frac{M}{d^2} \|x - t_i\|^2 \right), \quad i = 1, 2, \ldots, M
\]  

(7.108)

where \( M \) is the number of centers and \( d \) is the maximum distance between the chosen centers. In effect, the standard deviation (i.e., width) of all the Gaussian radial-basis functions is fixed at

\[
\sigma = \frac{d}{\sqrt{2M}}
\]  

(7.109)
Such a choice for the standard deviation \( \sigma \) merely ensures that the Gaussian functions are not too peaked or too flat; both of these extremes are to be avoided.

The only parameters that would need to be learned in this approach are the linear weights in the output layer of the network. A straightforward procedure for doing this is to use the pseudoinverse method (Broomhead and Lowe, 1988). Specifically, we have [see also Eqs. (7.87) and (7.88)]

\[
    w = G^+ d
\]

(7.110)

where \( d \) is the desired response vector in the training set. The matrix \( G^+ \) is the pseudoinverse of the matrix \( G \), which is itself defined as

\[
    G = \{g_{ij}\}
\]

(7.111)

where

\[
    g_{ij} = \exp \left( -\frac{M}{\sigma^2 \|x_j - t_i\|^2} \right), \quad j = 1, 2, \ldots, N; i = 1, 2, \ldots, M
\]

(7.112)

where \( x_j \) is the jth input vector of the training set.

Basic to all algorithms for the computation of a pseudoinverse of a matrix is the singular-value decomposition (Golub and Van Loan, 1989; Haykin, 1991):

**If \( G \) is a real \( N \)-by-\( M \) matrix, then there exist orthogonal matrices**

\[
    U = \{u_1, u_2, \ldots, u_N\}
\]

and

\[
    V = \{v_1, v_2, \ldots, v_M\}
\]

such that

\[
    U^T GV = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_K), \quad K = \min(M, N)
\]

(7.113)

where

\[
    \sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_K > 0
\]

The column vectors of the matrix \( U \) are called the left singular vectors of \( G \), and the column vectors of the matrix \( V \) are called its right singular vectors. The \( \sigma_1, \sigma_2, \ldots, \sigma_K \) are called the singular values of the matrix \( G \). According to the singular value decomposition theorem, the \( M \)-by-\( N \) pseudoinverse of matrix \( G \) is defined by

\[
    G^+ = V \Sigma^+ U^T
\]

(7.114)

where \( \Sigma^+ \) is itself an \( N \)-by-\( N \) matrix defined in terms of the singular values of \( G \) by

\[
    \Sigma^+ = \text{diag}\left( \frac{1}{\sigma_1}, \frac{1}{\sigma_2}, \ldots, \frac{1}{\sigma_K}, 0, \ldots, 0 \right)
\]

(7.115)

Efficient algorithms for the computation of a pseudoinverse matrix are discussed in the books by Golub and Van Loan (1989) and Haykin (1991).

2. **Self-Organized Selection of Centers**

In the second approach, the radial-basis functions are permitted to move the locations of their centers in a self-organized fashion, whereas the linear weights of the output layer are computed using a supervised learning rule. In other words, the network undergoes a
Radial-Basis Function Networks

hybrid learning process (Moody and Darken, 1989; Lippmann, 1989b). The self-organized component of the learning process serves to allocate network resources in a meaningful way by placing the centers of the radial-basis functions in only those regions of the input space where significant data are present.

For the self-organized selection of the hidden units’ centers, we may use the standard k-nearest-neighbor rule (Moody and Darken, 1989). This rule classifies an input vector \( \mathbf{x} \) by assigning it the label most frequently represented among the \( k \) nearest samples; in other words, a decision is made by examining the labels on the \( k \) nearest neighbors and taking a vote (Duda and Hart, 1973).

For the supervised learning operation to compute the linear weights of the output layer, we may use an error-correction learning rule such as the simple and yet highly effective least-mean-square (LMS) algorithm; this algorithm was discussed in Chapter 5. The outputs of the hidden units in the RBF network serve as the inputs of the LMS algorithm.

3. Supervised Selection of Centers

In the third approach, the centers of the radial-basis functions and all other free parameters of the network undergo a supervised learning process; in other words, the RBF network takes on its most generalized form. A natural candidate for such a process is error-correction learning, which is most conveniently implemented using a gradient-descent procedure that represents a generalization of the LMS algorithm.

The first step in the development of such a learning procedure is to define the instantaneous value of the cost function

\[
\mathcal{E} = \frac{1}{2} \sum_{j=1}^{N} e_j^2
\]

where \( N \) is the number of training examples used to undertake the learning process, and \( e_j \) is the error signal, defined by

\[
e_j = d_j - F^*(\mathbf{x}_j)
\]

\[
= d_j - \sum_{i=1}^{M} w_i G(||\mathbf{x}_j - \mathbf{t}_i||)
\]

The requirement is to find the free parameters \( \mathbf{w}, \mathbf{t}, \) and \( \Sigma_i^{-1} \) (the latter being related to the norm-weighting matrix \( \mathbf{C}_i \)) so as to minimize \( \mathcal{E} \). The results of this minimization are summarized in Table 7.3; the derivations of these results are presented as an exercise to the reader as Problem 7.12. The following points are noteworthy in Table 7.3.

- The cost function \( \mathcal{E} \) is convex with respect to the linear parameters \( \mathbf{w} \), but nonconvex with respect to the centers \( \mathbf{t} \) and matrix \( \Sigma_i^{-1} \); in the latter case, the search for the optimum values of \( \mathbf{t} \) and \( \Sigma_i^{-1} \) may get stuck at a local minimum in parameter space.
- The update equations for \( w_i, t_i, \) and \( \Sigma_i^{-1} \) are (in general) assigned different learning-rate parameters \( \eta_1, \eta_2, \) and \( \eta_3, \) respectively.
- Unlike the back-propagation algorithm, the gradient-descent procedure described in Table 7.3 for an RBF network does not involve error back-propagation.
- The gradient vector \( \partial\mathcal{E}/\partial \mathbf{t} \) has an effect similar to a clustering effect that is task-dependent (Poggio and Girosi, 1990a).

For the initialization of the gradient-descent procedure, it is often desirable to begin the search in parameter space from a structured initial condition that limits the region of
Parameter space to be searched to an already known useful area, which may be achieved by implementing a standard pattern-classification method as an RBF network (Lowe, 1991a). In so doing, the likelihood of converging to an undesirable local minimum in weight space is reduced. For example, we may begin with a standard Gaussian classifier, which assumes that each pattern in each class is drawn from a full Gaussian distribution.

An obvious question that arises at this stage of the discussion is: What can be gained by adapting the positions of the centers of the radial-basis functions? The answer to this question naturally depends on the application of interest. Nevertheless, on the basis of some results reported in the literature, there is practical merit to the idea of allowing the centers to move. Work done by Lowe (1989) on speech recognition using RBF networks indicates that nonlinear optimization of the parameters that define the activation functions of the hidden layer is beneficial when a minimal network configuration is required. However, 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 (1992) have compared the performance of (Gaussian) radial-basis function networks with fixed centers to that of generalized radial-basis function networks with adjustable centers; in the latter case, the positions of the centers are determined by supervised learning. The performance comparison was made for the NETtalk task. The original NETtalk experiment was carried out by Sejnowski and Rosenberg (1987) using a multilayer perceptron trained with the back-propagation algorithm; the purpose of the experiment was to understand how a neural network could learn to map English spelling into its phonetic pronunciation. The experimental study by Wettschereck and Dietterich in the NETtalk domain may be summarized as follows:

### TABLE 7.3 Adaptation Formulas for the Linear Weights and the Positions and Spreads of Centers for RBF Network

1. **Linear weights (output layer)**
   \[
   \frac{\partial \tilde{e}(n)}{\partial w_i(n)} = \sum_{j=1}^{N} e_j(n)G(||x_j - t_i(n)||_c)
   \]
   \[
   w_i(n + 1) = w_i(n) - \eta_i \frac{\partial \tilde{e}(n)}{\partial w_i(n)}, \quad i = 1, 2, \ldots, M
   \]

2. **Positions of centers (hidden layer)**
   \[
   \frac{\partial \tilde{e}(n)}{\partial t_i(n)} = 2w_i(n) \sum_{j=1}^{N} e_j(n)G'(||x_j - t_i(n)||_c) \Sigma_i^{-1}[x_j - t_i(n)]
   \]
   \[
   t_i(n + 1) = t_i(n) - \eta_i \frac{\partial \tilde{e}(n)}{\partial t_i(n)}, \quad i = 1, 2, \ldots, M
   \]

3. **Spreads of centers (hidden layer)**
   \[
   \frac{\partial \tilde{e}(n)}{\partial \Sigma_i^{-1}(n)} = -w_i(n) \sum_{j=1}^{N} e_j(n)G'(||x_j - t_i(n)||_c)Q_{0i}(n)
   \]
   \[
   Q_{0i}(n) = [x_j - t_i(n)][x_j - t_i(n)]^T
   \]
   \[
   \Sigma_i^{-1}(n + 1) = \Sigma_i^{-1}(n) - \eta_i \frac{\partial \tilde{e}(n)}{\partial \Sigma_i^{-1}(n)}
   \]

*a The term \(e_j(n)\) is the error signal of output unit \(j\) at time \(n\). The term \(G'(\cdot)\) is the first derivative of the Green’s function \(G(\cdot)\) with respect to its argument.
RBF networks (with unsupervised learning of the centers’ locations and supervised learning of the output-layer weights) did not generalize nearly as well as multilayer perceptrons trained with the back-propagation algorithm.

Generalized RBF networks (with supervised learning of the centers’ locations as well as the output-layer weights) were able to exceed substantially the generalization performance of multilayer perceptrons.

7.12 Computer Experiment

In this section we use a computer experiment to illustrate the design of a regularization RBF network based on the use of fixed centers selected randomly from the training data. The computer experiment involves a binary classification problem based on data drawn from two equiprobable overlapping two-dimensional Gaussian distributions corresponding to classes \( \mathcal{C}_1 \) and \( \mathcal{C}_2 \). Details of the Gaussian distributions are the same as those described in Section 6.9. Class \( \mathcal{C}_1 \) is characterized by mean vector \([0,0]^T\) and common variance \(1\), whereas class \( \mathcal{C}_2 \) is characterized by mean vector \([0,2]^T\) and common variance \(4\). The experiment described in this section may thus be viewed as the regularization RBF counterpart to the back-propagation learning experiment of Section 6.9.

With two classes \( \mathcal{C}_1 \) and \( \mathcal{C}_2 \), the regularization RBF network is constructed to have two output functions, one for each class. Also, binary-valued class indicator outputs are used as the desired output values, as shown by

\[
d_k^\theta = \begin{cases} 
1 & \text{if pattern } p \text{ belongs to class } \mathcal{C}_k \\
0 & \text{otherwise}
\end{cases}
\]

where \( k = 1, 2 \).

Before we proceed with the experiment, however, we have to resolve the issue of an output decision rule for performing the pattern classification. In Yee and Haykin (1993) it is shown that the outputs of a regularization RBF network classifier provide estimates of the posterior class probabilities. This is true only under the condition that the network is trained with the binary-valued class indicator vector type of desired outputs. Accordingly, we may proceed to apply the decision rule of Eq. (6.47) for this class of networks:

**Select the class corresponding to the maximum output function.**

The method of random selection of fixed centers is tested with different values of regularization parameter \( h \). For a prescribed \( \lambda \), Eq. (7.86) is used to compute the weight vector of the output layer in the RBF network, as shown by

\[
w = (G^TG + \lambda G_0)^{-1}G^Td
\]

where \( G \) is an \( N \)-by-\( M \) matrix whose jith element is equal to the radially symmetric function \( G(x_j; t_i) \), and \( G_0 \) is a symmetric \( M \)-by-\( M \) matrix whose jith element is equal to \( G(t_i; t_i) \).

For each \( \lambda \), the ensemble comprises 50 independent networks, each of which is tested against the same reference set of 1000 patterns. The norm weighting matrix \( C \) for a trial is assumed to be a diagonal matrix, defined by the diagonal elements of the sample covariance matrix that is computed from the particular trial’s training input data.

The results of the computer experiment described here are based on Yee (1992).
Table 7.4 presents the ensemble statistics for the average probability of correct classification \( P_c \), computed for the case of \( M = 20 \) centers that are randomly selected from \( N = 200 \) sized training input sets for each network trial. The ensemble statistics are computed for different values of the regularization parameter \( \lambda \).

Table 7.5 presents the corresponding results computed for the case of a smaller regularization RBF network with \( M = 10 \) centers that are selected randomly for \( N = 200 \) training input sets.

Figure 7.7 displays the decision boundaries formed by the network outputs for a regularization parameter \( \lambda = 1 \), for which we have the best statistics. The two parts of Fig. 7.7 correspond to the best- and worst-performing network within the ensemble under test; both parts of the figure are for the case of \( M = 20 \) units.

Comparing the results of Table 7.4 with those of Table 7.5, we see that the classification performance of the regularization RBF network is only marginally improved as a result of doubling the number of centers. Moreover, we note from both Tables 7.4 and 7.5 that an RBF network trained in accordance with the method of fixed centers selected at random is relatively insensitive to the regularization parameter \( \lambda \). This observation suggests that the use of randomly selected centers from a fixed training set is actually a regularization method in its own right!

### 7.13 Factorizable Radial-Basis Functions

From an implementation point of view, it would be highly attractive if the radial-basis functions used in the construction of a neural network are factorizable. Given such a property, we may then synthesize a multidimensional radial-basis function as the product of lower-dimensional (e.g., one-dimensional and two-dimensional) radial-basis functions. Indeed, the only radial-basis function that is factorizable is the Gaussian function (Poggio...

**TABLE 7.4** Size of Hidden Layer \( M = 20 \) Centers: Average Probability of Correct Classification, \( P_c \) (%)

<table>
<thead>
<tr>
<th>Ensemble Statistic</th>
<th>Regularization Parameter, ( \lambda )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Mean</td>
<td>80.01</td>
</tr>
<tr>
<td>Std. dev.*</td>
<td>1.20</td>
</tr>
<tr>
<td>Minimum</td>
<td>76.70</td>
</tr>
<tr>
<td>Maximum</td>
<td>81.80</td>
</tr>
</tbody>
</table>

* Std. dev.: Standard deviation.

**TABLE 7.5** Size of Hidden Layer \( M = 10 \) Centers: Average Probability of Correct Classification, \( P_c \) (%)

<table>
<thead>
<tr>
<th>Ensemble Statistic</th>
<th>Regularization Parameter, ( \lambda )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0</td>
</tr>
<tr>
<td>Mean</td>
<td>78.53</td>
</tr>
<tr>
<td>Std. dev.*</td>
<td>1.53</td>
</tr>
<tr>
<td>Minimum</td>
<td>75.00</td>
</tr>
<tr>
<td>Maximum</td>
<td>80.90</td>
</tr>
</tbody>
</table>

* Std. dev.: Standard deviation.
and Girosi, 1990a, b). To demonstrate this important property of Gaussian radial-basis functions, consider the simple case of a Gaussian function in two dimensions:

\[
G(\mathbf{x}; \mathbf{t}) = G(\| \mathbf{x} - \mathbf{t} \|^2) \\
= e^{-\| \mathbf{x} - \mathbf{t} \|^2} \\
= e^{-(x_1 - t_1)^2 - (x_2 - t_2)^2} \\
= e^{-(x_1 - t_1)^2} e^{-(x_2 - t_2)^2} \\
= G(x_1; t_1) G(x_2; t_2)
\]

\((7.118)\)
Equation (7.118) shows that a two-dimensional Gaussian activation function \( G(x; t) \) with a center \( t = [t_1, t_2]^T \) is equivalent to the product of a pair of one-dimensional Gaussian activation functions \( G(x_1; t_1) \) and \( G(x_2; t_2) \), where \( x_1 \) and \( x_2 \) are the elements of the input vector \( x \), and \( t_1 \) and \( t_2 \) are the elements of the center \( t \).

We may readily generalize the result of Eq. (7.118) for the case of a multivariate Gaussian activation function \( G(x; t) \) that computes the weighted norm, assuming that the weighting matrix \( C \) is a diagonal matrix. To be specific, suppose that we have

\[
C = \text{diag}[c_1, c_2, \ldots, c_p] \quad (7.119)
\]

We may then factorize \( G(x; t) \) as follows:

\[
G(x; t) = G(\|x - t\|^2_C)
= \prod_{k=1}^p \exp\left(-\frac{(x_k - t_k)^2}{2\sigma_k^2}\right) \quad (7.120)
\]

where the \( k \)th element \( t_k \) of the center \( t \) plays the role of a “mean.” The variance \( \sigma_k^2 \) is defined in terms of the \( k \)th diagonal element of the matrix \( C \) by

\[
\sigma_k^2 = \frac{1}{2c_k^2} \quad (7.121)
\]

For the more general case of a norm weighting matrix \( C \) that is nondiagonal, we may proceed as follows. Referring to Eq. (7.89), we first express the weighted norm \( \|x - t\|^2_C \) as

\[
\|x - t\|^2_C = (x - t)^T C^T C (x - t)
= (x - t)^T \Sigma^{-1} (x - t) \quad (7.122)
\]

where

\[
\Sigma^{-1} = C^T C
\]

In the context of a multivariate Gaussian function, the matrix \( \Sigma \) plays the role of a covariance matrix. In any event, we use the similarity transformation to factorize the inverse matrix \( \Sigma^{-1} \) in terms of its eigenvalues \( \lambda_1, \lambda_2, \ldots, \lambda_p \) and associated eigenvectors \( q_1, q_2, \ldots, Q \) as follows (Strang, 1980):

\[
\Sigma^{-1} = QAQ' \quad (7.123)
\]

where

\[
Q = [q_1, q_2, \ldots, q_p] \quad (7.124)
\]

\[
A = \text{diag} [\lambda_1, \lambda_2, \ldots, \lambda_p] \quad (7.125)
\]

Accordingly, we may rewrite the weighted norm of Eq. (7.122) in the new form

\[
\|x - t\|^2_C = (x' - t')^T A (x' - t') \quad (7.126)
\]

where

\[
x' = Qx \quad (7.127)
\]

and

\[
t' = Qt \quad (7.128)
\]

We now recognize that the matrix \( A \) is a diagonal matrix. We may therefore use our previous result to factorize the multivariate Gaussian activation function \( G(x, t) \).
as follows:

$$G(x; t) = \prod_{k=1}^{p} \exp \left( -\frac{(x_k' - t_k')^2}{2\sigma_k^2} \right)$$  \hspace{1cm} (7.129)

where

$$x_k' = q_k'x$$  \hspace{1cm} (7.130)

$$t_k' = q_k't$$  \hspace{1cm} (7.131)

$$\sigma_k^2 = \frac{1}{2\lambda_k}$$  \hspace{1cm} (7.132)

Knowing the norm weighting matrix $C$, we may precompute the transformed centers and the spreads. Then, the multivariate Gaussian function $G(x; t)$ may be computed using a set of one-dimensional Gaussian functions as shown in the block diagram of Fig. 7.8.

### 7.14 Discussion

The structure of an RBF network is unusual in that the constitution of its hidden units is entirely different from that of its output units. With radial-basis functions providing the foundation for the design of the hidden units, the theory of RBF networks is linked intimately with that of radial-basis functions, which is nowadays one of the main fields of study in numerical analysis (Singh, 1992). Another interesting point to note is that with linear weights of the output layer providing a set of adjustable parameters, much can be gained by ploughing through the extensive literature on linear adaptive filtering (Haykin, 1991).

### Curse of Dimensionality

Kernel-type approximations, such as standard RBF networks, suffer from the so-called *curse of dimensionality*, referring to the exponential increase in the number of hidden units with the dimension of the input space. The curse of dimensionality becomes particu-
larly acute in trying to solve large-scale problems such as image recognition and speech recognition. Standard RBF networks, using fixed Gaussian radial-basis functions with their exponents based on the Euclidean distance matrix, operate by constructing hyperspheres around the centers of the basis functions, and therefore suffer from the curse of dimensionality. We may alleviate this problem by replacing the hyperspheres with hyperellipses that have adjustable orientations. This is achieved, for example, by using generalized Gaussian radial-basis functions based on the notion of a weighted norm (Poggio and Girosi, 1990a; Lowe, 1989). Thus, in the case of hypersurface estimation (reconstruction), such a scheme permits the use of a smaller number of hidden units, but more parameters to adapt. Saha et al. (1991) describe another scheme referred to as oriented nonradial-basis function (ONRBF) networks, in which case the exponent of the Gaussian radial-basis function is replaced by a polynomial. Specifically, the activation function of the ith hidden unit is defined by

\[ \phi(x) = \exp(-\|T_i(x_1, x_2, \ldots, x_p, 1)^T\|^p) \] (7.133)

where \( x_1, x_2, \ldots, x_p \) are the elements of the input vector \( x \), and \( T_i \) is a \( p \)-by-(\( p + 1 \)) matrix.

The matrices \( T_i, i = 1, 2, \ldots, M \), transform the input vectors, and these transformations correspond to translation, scaling, and rotation of the input vectors.

**Orthogonal Least-Squares Method**

Another important issue that occupied much of our attention in this chapter was the use of regularization to discourage the RBF network from overfitting the training data. An alternative way in which this objective may be achieved is to use the orthogonal least squares (OLS) learning procedure (Chen et al., 1991). This method is rooted in linear regression models, according to which a desired response \( d(n) \) is defined by (Haykin, 1991):

\[ d(n) = \sum_{i=1}^{M} x_i(n)a_i + e(n), \quad n = 1, 2, \ldots, N \] (7.134)

where the \( a_i \) are the model parameters, the \( x_i(n) \) are the regressors, and \( e(n) \) is the residue (error). Using matrix notation, we may rewrite Eq. (7.134) as

\[ d = Xa + e \] (7.135)

where

\[ d = [d(1), d(2), \ldots, d(N)]^T \]

\[ a = [a_1, a_2, \ldots, a_M]^T \]

\[ X = [x_1, x_2, \ldots, x_M] \]

\[ x_i = [x_i(1), x_i(2), \ldots, x_i(N)]^T, \quad 1 \leq i \leq M \]

\[ e = [e(1), e(2), \ldots, e(N)]^T \]

The regressor vectors \( x_i \) form a set of basis vectors, and the least-squares solution of Eq. (7.135) satisfies the condition that the matrix product \( Xa \) be the projection of the desired response vector \( d \) onto the space spanned by the basis vectors—hence the name of the method. The OLS method involves the transformation of the regressor vectors \( x_1, x_2, \ldots, x_M \) into a corresponding set of orthogonal basis vectors denoted by \( u_1, u_2, \ldots, u_M \). For example, the standard Gram–Schmidt orthogonalization procedure may be used to perform
Radial-Basis Function Networks

this transformation, as shown by

\[ u_1 = x_1 \]

\[ \alpha_k = \frac{u_i^T x_i}{u_i^T u_k}, \quad 1 \leq i < k \]

\[ u_k = x_k - \sum_{i=1}^{k-1} \alpha_i x_i \]

where \( k = 2, \ldots, M \).

In the context of a neural network, the OLS learning procedure chooses the radial-basis function centers \( t_1, t_2, \ldots, t_M \) as a subset of the training data vectors \( x_1, x_2, \ldots, x_M \), where \( M < N \). The centers are determined one-by-one in a well-defined manner (e.g., following the Gram–Schmidt orthogonalization procedure), until a network of adequate performance is constructed. At each step of the procedure, the increment to the explained variance of the desired response is maximized. In so doing, the OLS learning procedure will generally produce an RBF network whose hidden layer is smaller than that of an RBF network with randomly selected centers, for a specified level of unexplained variance of the desired response. Furthermore, the problem of numerical ill-conditioning encountered in the random selection of centers method is avoided. Thus, the OLS learning procedure provides another useful approach for the construction of a parsimonious RBF network with good numerical properties.

7.15 Applications

Radial-basis function (RBF) networks have been applied to a wide variety of problems, though perhaps not as many as those involving multilayer perceptrons. Nevertheless, the range of applications of RBF networks covered in the literature is quite broad, as illustrated by the following representative list:

- Image processing (Saha et al., 1991; Poggio and Edelman, 1990)
- Speech recognition (Ng and Lippmann, 1991; Niranjan and Fallside, 1990)
- Time-series analysis (He and Lapedes, 1991; Kadirkamanathan et al., 1991; Moody and Darken, 1989; Broomhead and Lowe, 1988)
- Adaptive equalization (Chen et al., 1992a, b; Cid-Sueiro and Figueiras-Vidal, 1993; Kassam and Cha, 1993)
- Radar point-source location (Webb, 1993)
- Medical diagnosis (Lowe and Webb, 1990)

In this section we focus on two specific applications of RBF networks, one relating to the prediction of a chaotic time series, and the other relating to adaptive equalization of communication channels.

Prediction of Chaotic Time Series

Broomhead and Lowe (1988) and Kadirkamanathan et al. (1991), have used RBF networks successfully for the prediction of a chaotic time series. The chaotic time series considered therein is the logistic map whose dynamics is governed by the difference equation:

\[ x_{n+1} = \mu x_n (1 - x_n) \]

\[ \mu = 3.57 \]

For a discussion of chaos, see Chapter 14.
This is a first-order nonlinear process where only the previous sample \( x_{n-1} \) determines the value of the present sample. The logistic map is known to be chaotic on the interval \([0,1]\). The RBF network used by Broomhead and Lowe was constructed as follows. The centers of the radial-basis function were chosen to be uniformly spaced on \((0,1)\); the number of centers was an adjustable parameter. The linear weights of the output layer were computed using the pseudoinverse of the matrix \( G \) in accordance with Eq. (7.88); the pseudoinverse matrix \( G^+ \) was itself computed using the singular value decomposition of matrix \( G \). Figure 7.9 shows the actual and predicted outputs of the network over the interval \([0,1]\) for one iterate.
Kadirkamanathan et al. (1991) use a sequential adaptation algorithm for Gaussian RBF networks, which is based on the principle of successive $F$-projections. This is a general method of choosing a posterior function estimate of an unknown function $f$ when there exists a prior estimate and new information about $f$ in the form of constraints. The principle of successive $F$ projections states that, of all the functions that satisfy the constraints, we should choose the posterior $F_n$ that has the least $L^2$ norm, $\|F_n - F_{n-1}\|$, where $F_{n-1}$ is the prior estimate off. That is,

$$F_n = \arg \min_F \|F - F_{n-1}\| \text{ such that } F_n \in H_I$$

(7.137)

where $H_I$ is the set of functions that satisfy the new constraints, and

$$\|F - F_{n-1}\|^2 = \int_{\mathbb{R}^x} |F(x) - F_{n-1}(x)|^2 \, dx$$

(7.138)

where $x$ is the input vector, and $dx$ is an infinitesimal volume in the input space $X$. For RBF networks, the application of the principle of successive $F$ projections yields an algorithm that has two steps:

- Initialize parameters of the network with random values or values based on prior knowledge.
- For each training example $(x_j, d_j), j = 1, 2, \ldots, N$, compute the posterior parameter estimate

$$\hat{\theta}_j = \arg \min_{\theta} \int_{\mathbb{R}^x} |F(x, \theta) - F(x_j, \theta_{j-1})|^2 \, dx$$

(7.139)

where $(x_j, d_j), j = 1, 2, \ldots, N$, constitutes the observation (training) set, $\theta$ is the parameter set of the network, and $F(x, \theta)$ is the function constructed by the network.

There are two good reasons for applying the sequential adaptation algorithm to an RBF network with Gaussian hidden units (Kadirkamanathan et al., 1991). First, the method of successive $F$ projections minimizes the volume change in the hypersurface as new patterns are learned. Since an RBF network with Gaussian hidden units provides a local approximation to the functional relationship between the desired response and input patterns, it follows that only a few hidden units undergo adaptation. The algorithm is therefore quite stable. Second, for an RBF network with Gaussian hidden units, the $L^2$-norm measure of volume change in the hypersurface lends itself to an analytic solution.

Kadirkamanathan et al. (1991) used an RBF network with the successive $F$-projections algorithm to predict the chaotic time series defined by the logistic map of Eq. (7.136). The network had a single input node and 8 Gaussian hidden units. Figure 7.10 shows the result on the map constructed by the network for 0, 20, and 60 samples; the samples used to do the training are also shown in Fig. 7.10. Each sample was presented to the network only once for training. From Fig. 7.10 we see that the algorithm provides a close fit to the logistic map after training on 20 samples. We also see that the algorithm is stable in that the map is maintained up to training on 60 samples.

Adaptive Equalization

Adaptive equalization is a well-established technique used in digital communication systems to combat the effects of imperfections in communication channels (Qureshi, 1985). The equalizer is positioned at the front end of the receiver, as depicted in the baseband discrete-time model of Fig. 7.11. A data sequence $\{a(n)\}$ is transmitted through a dispersive channel whose transfer function is modeled by
The data sequence \( \{a(n)\} \) is assumed to consist of a string of equiprobable and statistically independent binary symbols 0 and 1, represented by the levels \(-1\) and \(+1\), respectively. The channel output \( s(n) \) is corrupted by front-end receiver noise \( v(n) \) that is commonly modeled as additive white Gaussian noise. The received signal is

\[
x(n) = s(n) + v(n)
\]

whose two components are uncorrelated with each other. The signal \( x(n) \) is applied to an equalizer, positioned at the front end of the receiver. The equalizer is designed to produce an estimate \( \hat{a}(n) \) of the original symbol \( a(n) \).

There are three factors to be considered in the design of the equalizer:

1. **Intersymbol interference (ISI)**, which is a signal-dependent form of interference that is caused by the dispersive effects of the channel; it is because of ISI that the channel output \( s(n) \) is markedly different from the transmitted symbol \( a(n) \).

2. **Receiver noise**, which is always present.
3. **Type of channel**, that is, whether the channel is *minimum phase* or *nonminimum phase*. The channel is minimum phase if the zeros of its transfer function \( H(z) \) are all confined to the interior of the unit circle in the \( z \) plane; under this condition, the amplitude and phase responses of the channel are uniquely related to each other. Otherwise, the channel is nonminimum phase.

The conventional form of the equalizer is based on linear adaptive filter theory, with its weights being adjusted by means of the simple least-mean-square (LMS) algorithm discussed in Chapter 5. In particular, the equalizer design represents the “best” compromise between eliminating intersymbol interference and enhancing receiver noise. In the absence of noise and for minimum-phase channels, the equalizer operates as an *inverse model* such that the combination of the channel and the equalizer provides “distortionless” transmission. When, however, noise is present and/or the channel is nonminimum phase, the use of an inverse model is no longer optimum.

An alternative viewpoint to that of inverse filtering is to approach the equalization process as a *pattern-classification* problem (Theodoridis et al., 1992). For the simple case of bipolar data transmission, the received samples, corrupted by ISI and noise, would have to be classified as \(-1\) and \(+1\). The function of the equalizer is therefore to assign each received sample to the correct decision region. According to this viewpoint, a linear equalizer is equivalent to a linear classifier. However, in realistic situations where noise is present in the received signal and/or the communication channel is nonminimum phase, it has been shown by Gibson and Cowan (1990) that the optimum classifier is in fact nonlinear.

In order to see how the design of a nonlinear adaptive equalizer can be based on neural networks, consider first the block diagram of Fig. 7.12, which describes a *decision-feedback* equalizer. The equalizer consists of a nonlinear filter with *feedforward* inputs denoted by \( x(n), x(n-1), \ldots, x(n-m) \) and *feedback* inputs denoted by \( a(n-\tau-1), a(n-\tau-2), \ldots, a(n-\tau-k) \), where \( x(n) \) is the channel output at time \( n \), and \( a(n-\tau) \) is the equalizer output representing an estimate of the transmitted symbol \( a(n) \) delayed by \( \tau \) symbols. The equalizer output is produced by passing the nonlinear filter output through a *hard limiter*, whereby decisions are made on a symbol-by-symbol basis. Chen et al. (1992a) have derived an optimum form of the decision feedback equalizer shown in Fig. 7.12 for the channel described in Eq. (7.140), as described next.

For a channel modeled as a finite-duration impulse response (FIR) filter of length \( l \) as in Eq. (7.140) and an equalizer of feedforward order \( m \) as shown in Fig. 7.12, there are

\[
N = 2^{l+m}
\]  

(7.142)

combinations of the channel input sequence.

**FIGURE 7.12** Block diagram of decision-feedback equalizer using a nonlinear filter.
Thus, the channel output vector
\[ s(n) = [s(n), s(n - 1), \ldots, s(n - m + 1)]^T \]  
(7.144)

has \( N \) desired states; the vector \( s(n) \) is the noise-free version of the vector of feedforward inputs applied to the nonlinear filter in Fig. 7.12. It is sufficient to consider a feedback order
\[ k = l + m - 1 - \tau \]  
(7.145)

where \( \tau \) is an integer representing the number of symbols by which the equalizer output is delayed with respect to the channel input. Let it be initially assumed that the decision-feedback equalizer in Fig. 7.12 uses the correct feedback vector
\[ a_f(n - \tau) = \{a(n - \tau - 1), a(n - \tau - 2), \ldots, a(n - \tau - k)\}^T \]  
(7.146)

Let
\[ N_j = 2^k \]  
(7.147)

denote the number of combinations of the feedback vector \( a_f(n - \tau) \), which are written as
\[ \{a_{ij} | j = 1, 2, \ldots, N_j\} \]  
(7.148)

The set of desired channel states \( S_{m,\tau} \) may now be divided into \( N_j \) subsets conditioned on \( a_f(n - \tau) \), as shown by
\[ S_{m,\tau} = \bigcup_{1 \leq j \leq N_j} S_{m,\tau,j} \]  
(7.149)

where
\[ S_{m,\tau,j} = S^+_{m,\tau,j} \bigcup S^-_{m,\tau,j} \]  
(7.150)

with \( S^+_{m,\tau,j} \) and \( S^-_{m,\tau,j} \) themselves defined by, respectively,
\[ S^+_{m,\tau,j} = \left\{ s(n) | a(n - \tau) = +1 \bigcap a_j(n - \tau) = a_{ij} \right\} \]  
(7.151)

and
\[ S^-_{m,\tau,j} = \left\{ s(n) | a(n - \tau) = -1 \bigcap a_j(n - \tau) = a_{ij} \right\} \]  
(7.152)

We are now ready to define an optimum solution for the equalizer output \( a(n - \tau) \) that is conditioned on \( a_f(n - \tau) = a_{ij} \). Specifically, following Chen et al. (1992a, b), we may write
\[ a(n - \tau) = \begin{cases} 1, & F_b[x(n)|a_j(n - \tau) = a_{ij}] \geq 0 \\ -1, & F_b[x(n)|a_j(n - \tau) = a_{ij}] < 0 \end{cases} \]  
(7.153)

where \( F_b(\cdot, \cdot) \) is an optimum conditional decision function defined by
\[ F_b(x(n)|a_j(n - \tau) = a_{ij}) = \sum_{x_j \in S_{m,\tau,j}} \alpha \exp \left( -\frac{1}{2\sigma^2} \|x(n) - x_j\|^2 \right) \]  

\[ - \sum_{x_j \in S_{m,\tau,j}} \alpha \exp \left( -\frac{1}{2\sigma^2} \|x(n) - x_j\|^2 \right) \]  
(7.154)
where $\alpha$ is an arbitrary constant, $\sigma_v^2$ is the variance of the noise $\nu(n)$, and $\mathbf{x}(n)$ is the feedforward input vector of the equalizer,
\[
\mathbf{x}(n) = [x(n), x(-1), \ldots, x(n - m + 1)]^T
\]  
(7.155)

The important point to note is that the feedback vector does not enter Eq. (7.154); it is merely used to narrow down the number of desired states that would have to be considered at each time $n$ (Chen et al., 1992a, b). In particular, at each $n$ only $2^{r+1}$ states are required to compute the decision function of Eq. (7.154). On the other hand, all the $N_r$ states would be required without decision feedback. Furthermore, it is shown in Chen et al. (1992a, b) that $m = \tau + 1$ is sufficient for decision feedback equalization. In general, we have $m > \tau + 1$ in the case of an adaptive equalizer without decision feedback. Hence, the use of decision feedback can result in a reduction in computational complexity of the adaptive equalizer by a factor larger than $2^l$, where $l$ is the memory span of the channel. According to Chen et al. (1992a, b), decision feedback improves the equalizer's performance by increasing the minimum distance between the Gaussian centers $\mathbf{x}_i^+$ and $\mathbf{x}_i^-$ at each time $n$.

The decision function defined in Eq. (7.154) is readily implemented using an RBF network. Specifically, comparing Eq. (7.154) with (7.153), we may make the following observations on the constitution of the RBF network (Chen et al., 1992a, b):

- The number $M$ of hidden units is set equal to $N_r$, where $N_r$ is defined by Eq. (7.142).
- The hidden units are grouped in accordance with the conditional subsets $S_{m,r,l}$, and their centers $t_i$ define the corresponding channel states $s_i \in S_{m,r,l}$. In effect, the feedback vector determines which subset of hidden units should be active at any particular time $n$.
- The radial-basis functions are Gaussian functions centered on $t_i$, and with all their widths chosen equal to $\sigma_v$ (i.e., the standard deviation of the receiver noise).
- The weights of the output layer are set equal to $-1$ or $+1$.

Thus, the RBF decision-feedback equalizer described here represents a novel solution to the adaptive equalization problem. It is of interest to note that this equalizer structure is similar to the general block decision-feedback equalizer proposed by Williamson et al. (1992).

Two adaptive approaches are described by Chen et al. (1992a,b) to realize decision feedback equalization using the RBF network. The first method estimates the channel model based on the LMS algorithm, and then uses the channel estimate to compute a subset of centers for use in the RBF network. The second method uses a clustering technique to estimate the channel states directly. The first method requires a shorter training sequence, while the second method offers a lower computational load and greater immunity to nonlinear channel distortion.

Chen et al. (1992b) have used computer simulation to investigate the performance of an RBF decision feedback equalizer and compare its performance with that of a standard decision-feedback equalizer and a maximum-likelihood sequential estimator known as the Viterbi algorithm. The investigations were carried out for both stationary and nonstationary communication channels; the highly nonstationary channel considered was chosen to be representative of a mobile radio environment. The results of the investigations reported by Chen et al. (1992b) may be summarized as follows:

- The maximum-likelihood sequential estimator provides the best attainable performance for the case of a stationary channel; the corresponding performance of the RBF decision-feedback equalizer is worse off by about 2 dB, but better than that of the standard decision-feedback equalizer by roughly an equal amount.
In the case of a highly nonstationary channel, the RBF decision-feedback equalizer outperforms the maximum-likelihood sequential estimator; it is suggested that performance degradation in the latter case may be due to the accumulation of tracking errors.

The results of this study appear to show that the RBF decision-feedback equalizer is robust and provides a viable solution for the equalization of highly nonstationary channels.

**PROBLEMS**

7.1 In Example 2 we pointed out that for the differential operator

\[ \mathbf{P} F(x) = \frac{d^2}{dx^2} F(x) \]

the function \( F(x) \) that minimizes the cost functional \( \mathcal{E}(F) \) consists of a cubic spline. Develop the details of this statement.

7.2 The multiquadric

\[ \varphi(r) = (r^2 + c^2)^{1/2}, \quad c > 0; \quad r \geq 0 \]

and the inverse multiquadric

\[ \varphi(r) = \frac{1}{(r^2 + c^2)^{1/2}}, \quad c > 0; \quad r \geq 0 \]

provide two possible choices for radial-basis functions. An RBF network using the inverse multiquadric constructs local approximation to nonlinear input–output mapping. On the other hand, the use of a multiquadric represents a counterexample to this property of \( \mathbf{I} \mathbf{U}^3 \mathbf{F} \) networks. Justify the validity of these two statements.

7.3 The set of values given in Section 7.8 for the weight vector \( \mathbf{w} \) of the RBF network of Fig. 7.6 presents one possible solution for the XOR problem. Investigate another set of values for the weight vector \( \mathbf{w} \) for solving this problem.

7.4 In Section 7.8 we presented a solution of the XOR problem using an RBF network with two hidden units. In this problem we consider an exact solution of the XOR problem using an RBF network with four hidden units, with each radial-basis function center being determined by each piece of input data. The four possible input patterns are defined by \((0,0), (0,1), (1,1), (1,0)\), which represent the cyclically ordered corners of a square.

(a) Construct the interpolation matrix \( \Phi \) for the resulting RBF network. Hence, compute the inverse matrix \( \Phi^{-1} \).

(b) Calculate the linear weights of the output layer of the network.

7.5 The method of successive approximations (He and Lapedes, 1991) is based on partitioning the \( N \) centers given for strict interpolation into \( N' \) groups of \( L \) centers each. The \( L \) centers of each group are selected randomly from the training input data, which are used to compute the first set of linear weights \( \{ \mathbf{b}_j \}_{i=1}^L \); \( j = 1, \ldots, N' \). Next, each group of basis functions and its associated set of linear weights is viewed to be a composite basis function. The method of fixed centers is again applied to the set of \( N' \) composite basis functions to yield a second set of linear weights \( \{ \mathbf{a}_j \}_{j=1}^{N'} \). The overall result, for the case of a single output node, is thus described as follows:

\[ F(x) = \sum_{j=1}^{N'} \left[ \mathbf{a}_j \sum_{i=1}^L \mathbf{b}_{ij} G_{ij}(x) \right] \]
where

\[ G_{ij}(x) = G(x; x_i) \]

(a) Show that the learning-time complexity for the successive approximation method is roughly \( N'NL' + NN' \) compared to \( N^3 \) for strict interpolation.

(b) After training is completed, the two layers of linear weights can be collapsed into a single layer. Compute the linear weights of such an equivalent layer.

7.6 Compare the Boolean limiting version of an RBF network with that of a multilayer perceptron.

7.7 In this problem we continue with the computer experiment in Section 7.12 to study the method of strict interpolation for the design of an RBF network used as a binary pattern classifier. The purpose of the experiment is twofold:

- To demonstrate that the generalization performance of the network so trained is relatively poor
- To demonstrate that the generalization performance of the network can be improved using regularization

The network is intended to solve the binary pattern-recognition problem described in Section 7.12, where the requirement is to classify data drawn from a mixture model consisting of two equiprobable overlapping two-dimensional Gaussian distributions. One distribution has a mean vector \([0, 0]^T\) and common variance 1, whereas the other distribution has a mean vector \([0, 2]^T\) and common variance 4. The “select class with maximum function output” decision rule is used for the classification.

(a) Consider a strict interpolation network using \( N = 20 \) centers (selected at random). Compute the mean, standard deviation, minimum and maximum values of the average probability of correct classification \( P_c \) for different values of regularization parameter \( \lambda = 0, 0.1, 1, 10, 100, 1000 \). For the computation of ensemble statistics, use 50 independent network trials per ensemble, with each one tested against a fixed reference set of 1000 patterns.

(b) Construct the decision boundary computed for the configuration described in part (a) for regularization parameter \( \lambda = 0 \).

(c) Repeat the computations described in part (a) for \( N = 100 \) centers (selected at random).

(d) Construct the decision boundary computed for part (c) for regularization parameter \( \lambda = 10 \).

(e) In light of your results, discuss the merit of strict interpolation as a method for the design of RBF networks, and the role of regularization in the performance of the network as a pattern classifier.

(f) Compare your results with those presented in Section 7.12 that were computed using the method of fixed centers selected at random. Hence, confirm the notion that the latter method is superior to strict interpolation for the design of RBF networks.

7.8 It may be argued that in the case of the experiment described in Section 7.12 involving the classification of a pair of Gaussian-distributed classes, the RBF network considered there was able to perform well since it is using Gaussian radial-basis functions to approximate the underlying Gaussian class conditional distributions. In this problem we use a computer experiment to explore the design of a strict-interpolation Gaussian RBF network for distinctly discontinuous class conditional distributions. Specifically, consider two equiprobable classes \( \mathcal{C}_1 \) and \( \mathcal{C}_2 \) whose distributions
are described by, respectively:

- \( U(\mathbb{C}_1) \), where \( \mathbb{C}_1 = \Omega_1 \) is a circle of radius \( r = 2.34 \) centered at \( x_c = [-2, 30]^T \)
- \( U(\mathbb{C}_2) \), where \( \mathbb{C}_2 \subset \mathbb{R}^2 \) is a square region centered at \( x_c \) with side length \( r = \sqrt{2\pi} \)

Here \( U(\Omega) \) denotes a uniform distribution over \( \Omega \subset \mathbb{R}^2 \). These parameters are chosen so that the decision region for class \( \mathbb{C}_1 \) is the same as in the Gaussian-distributed case considered in Section 7.12.

**Investigate the use of regularization as a means of improving the classification performance of a Gaussian RBF network using strict interpolation.**

7.9 It may be argued, by virtue of the central limit theorem, that for an RBF network the output produced by the network in response to a random input vector may be approximated by a Gaussian distribution, and that the approximation gets better as the number of centers in the network is increased. Investigate the validity of this notion.

7.10 Consider the cost functional

\[
\mathcal{E}(F^*) = \sum_{i=1}^{N} \left[ d_i - \sum_{j=1}^{M} w_j G(\|x_j - t_i\|) \right]^2 + \lambda \|PF^*\|^2
\]

which refers to the approximating function

\[
F^*(x) = \sum_{i=1}^{M} w_i G(\|x - t_i\|)
\]

Using the Fréchet differential, show that the cost functional \( \mathcal{E}(F^*) \) is minimized when

\[
(G^T G + \lambda G_0)w = G^T d
\]

where the N-by-M matrix \( G \), the M-by-M matrix \( G_0 \), the M-by-1 vector \( w \), and the N-by-1 vector \( d \) are defined by Eqs. (7.82), (7.85), (7.83), and (7.49), respectively.

7.11 Suppose that we define

\[
(P^*P)_U = \sum_{k=0}^{p} (-1)^k \frac{\nabla_U^k}{k!2^k}
\]

where

\[
\nabla_U^k = \sum_{j=1}^{p} \sum_{i=1}^{p} u_{ji} \frac{\partial^k}{\partial x_j \partial x_i}
\]

The p-by-p matrix \( U \), with its jth element denoted by \( u_{ji} \), is symmetric and positive definite. Hence the inverse matrix \( U^{-1} \) exists, and so it permits the following decomposition via the similarity transformation:

\[
U^{-1} = V^T D V = V^T D^{1/2} D^{1/2} V = C^T C
\]

where \( V \) is an orthogonal matrix, \( D \) is a diagonal matrix, \( D^{1/2} \) is the square root of \( D \), and the matrix \( C \) is defined by

\[
C = D^{1/2} V
\]
The problem is to solve for the Green’s function \( G(\mathbf{x}; t) \) that satisfies the following condition (in the distributional sense):

\[
(P^*P)G(\mathbf{x}; t) = \delta(\mathbf{x} - \mathbf{t})
\]

Using the multidimensional Fourier transform to solve this equation for \( G(\mathbf{x}; t) \), show that

\[
G(\mathbf{x}; t) = \exp \left( -\frac{1}{2} ||\mathbf{x} - \mathbf{t}||_E^2 \right)
\]

where \( ||\mathbf{x}||_E^2 = \mathbf{x}^T C^T C \mathbf{x} \)

7.12 Consider the cost functional

\[
\mathcal{E} = \frac{1}{2} \sum_{j=1}^{N} e_j^2
\]

where

\[
e_j = d_j - F^*(x_j)
\]

\[
= d_j - \sum_{i=1}^{M} w_i G(||x_j - t_i||_c)
\]

The free parameters are the linear weights \( w_i \), the centers \( t_i \) of the Green’s functions, and the inverse covariance matrix \( \Sigma_i^{-1} = C_i^T C_i \), where \( C_i \) is the norm weighting matrix. The problem is to find the values of these free parameters that minimize the cost functional \( \mathcal{E} \). Derive the following partial derivatives:

(a) \( \frac{\partial \mathcal{E}}{\partial w_i} = \sum_{j=1}^{N} e_j G(||x_j - t_i||_c) \)

(b) \( \frac{\partial \mathcal{E}}{\partial t_i} = 2w_i \sum_{j=1}^{N} e_j G(||x_j - t_i||_c) \Sigma_i^{-1}(x_j - t_i) \)

(c) \( \frac{\partial \mathcal{E}}{\partial \Sigma_i^{-1}} = -w_i \sum_{j=1}^{N} e_j G(||x_j - t_i||_c) Q_{ji} \)

where \( G'(\cdot) \) is the derivative of \( G(\cdot) \) with respect to its argument, and

\[
Q_{ji} = (x_j - t_i)(x_j - t_i)^T
\]

For differentiation with respect to a vector as in part (b), you may use the rule: If \( v = \mathbf{x}^T \mathbf{x} \), then

\[
\frac{\partial v}{\partial \mathbf{x}} = 2 \mathbf{x}
\]

For differentiation with respect to a matrix as in part (c), you may use the rule: If \( v = \mathbf{x}^T \mathbf{A} \mathbf{x} \), then

\[
\frac{\partial v}{\partial \mathbf{A}} = 2 \mathbf{x} \mathbf{x}^T
\]