Zhihua Zhang

Multivariate Time Series Analysis in Climate and Environmental Research

Springer

Multivariate Time Series Analysis in Climate and Environmental Research

Zhihua Zhang

Multivariate Time Series Analysis in Climate and Environmental Research

Springer

Zhihua Zhang
College of Global Change and Earth System
Science
Beijing Normal University
Beijing
China

ISBN 978-3-319-67339-4 ISBN 978-3-319-67340-0 (eBook)
https://doi.org/10.1007/978-3-319-67340-0

Library of Congress Control Number: 2017954476

Printed on acid-free paper

This Springer imprint is published by Springer Nature
The registered company is Springer International Publishing AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

The Earth's climate is a complex, multidimensional multiscale system in which different physical processes act on different temporal and spatial scales. Due to the increasing atmospheric greenhouse gas concentrations, global average temperatures increase with time as a result of interactions among components of the climate system. These interactions and the resulting variations in various climate parameters occur on a variety of timescales ranging from seasonal cycles, yearly cycles to those with times measured in hundreds of years. Climatologists and environmentalists are striking to extract meaningful information from huge amount of observational record and simulation data for the climate system. Classic univariate time series analysis is not capable to handle well these complex multidimensional data. Recently, the techniques and methods of multivariate time series analysis have gained great important in revealing mechanisms of climate change, modeling tempo-spatial evolution of climate change and predicting the trend of future climate change.

This book covers the comprehensive range of theory, models, and algorithms of state-of-the-art multivariate time series analysis which have been widely used in monitoring, modeling, and prediction of climate and environmental change. Each chapter focuses on a specific issue of importance. Chapter 1 discusses artificial neural networks which can make full use of some unknown information hidden in high-dimensional climate data, although these information cannot be extracted directly; Chap. 2 discusses multivariate Harmonic analysis which can determine how the total variance of multivariate time series is distributed in frequency. Main techniques and methods include Fourier transform, fractional Fourier transform, space–frequency representation, sparse approximation, spherical harmonics, and harmonic analysis on graphs; Chap. 3 discusses wavelet representation for multivariate time series with time-dependent dominant cycles. Main techniques and methods include multiresolution analysis and wavelets, discrete wavelet transform, wavelet packet, wavelet variance, significant tests, wavelet shrinkage, and shearlets, bandelets, and curvelets. Chapter 4 focuses on stochastic representation and modeling, including stationarity and trend tests, principal component analysis, factor analysis, cluster analysis, discriminant analysis, canonical correlation analysis,

multidimensional scaling, vector ARMA models, Monte Carlo methods, Black–Scholes model, and stochastic optimization; Chap. 5 discusses multivariate spectral analysis and estimation, including periodogram method, Blackman–Tukey method, maximum entropy method, multitaper method, vector ARMA spectrum, and multichannel SSA; Chap. 6 focuses on the development of climate models and related experiments to understand the climate system and climate change; Chap. 7 gives some latest case studies on regional climate change to demonstrate how the methods and tools in Chaps. 1–6 are used; Chap. 8 discusses basic models and key indices on ecosystem and global carbon cycle; Chap. 9 discusses the methods used to reconstruct paleoclimates from proxy data. Chapter 10 introduces three methods to analyze multivariate time series in climate change economics and related latest researches.

Current climate and environmental research is facing the challenge of complex multidimensional data. This book on multivariate time series analysis starts from first principles, always explains various techniques and methods step by step, and shows clearly how to reveal physical meaning from the analysis of observed and stimulated multidimensional data. It has a comprehensive cover and also includes many of the author's unpublished researches. This book is accessible for researchers and advanced students who want to grasp state-of-the-art techniques and methods in multivariate time series analysis. This book builds a cross-disciplinary bridge between various analysis techniques and methods and latest published studies in the wide branches of climatology and environmental science.

Beijing, China Zhihua Zhang

Contents

Chapter 1
Artificial Neural Network

Multivariate time series analysis in climate and environmental research always requires to process huge amount of data. Inspired by human nervous system, the *artificial neural network* methodology is a powerful tool to handle this kind of difficult and challenge problems and has been widely used to investigate mechanism of climate change and predict the climate change trend. The main advantage is that artificial neural networks make full use of some unknown information hidden in climate data although they cannot extract it. In this chapter, we will introduce various neural networks, including linear networks, radial basis function networks, generalized regression networks, Kohonen self-organizing networks, learning vector quantization networks, and Hopfield networks.

1.1 Network Architectures

Artificial neural network is a structure of interconnected units of large number of *neurons*. Each neuron in the network is able to receive input signals, to process them, and to send an output signal. It consists of a set of the weighted synapses, an adder for summing the input data weighted by the respective synaptic strength, and an activation function for limiting the amplitude of the output of the neuron. Network architectures have two fundamentally different classes including multilayer feedforward networks and recurrent networks.

1.1.1 Multilayer Feedforward Networks

Feedforward networks are currently being used in a variety of climate and environment applications with great success. It consists of a number of neurons organized in

Z. Zhang, *Multivariate Time Series Analysis in Climate and Environmental Research*, https://doi.org/10.1007/978-3-319-67340-0_1

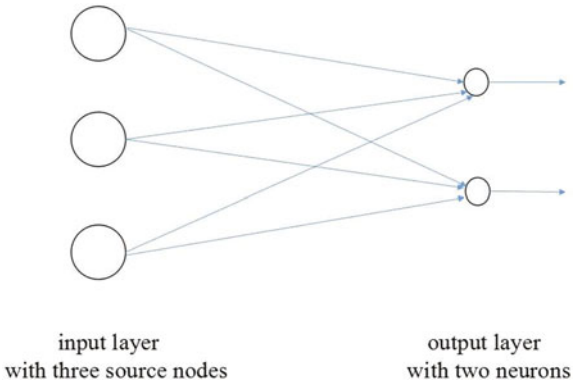

Fig. 1.1 Single-layer feedforward networks

layers. Every neuron in a layer is connected with all the neurons in the previous layer. These connections are not all equal; each connection may have a different strength or weight.

The oldest and simplest artificial neural networks is a single-layer feedforward neural network (see Fig. 1.1). It consists of an input layer of source nodes and an output layer of neurons, where source nodes are projected directly onto the output layer of neurons. The word "single-layer" means that the neural network has only a layer. The layer of source nodes is not counted because no computation is performed.

A multilayer feedforward network consists of an input layer of source nodes, one or more *hidden layers*, and an output layer of neurons (see Fig. 1.2). The hidden layers in the network are not seen directly from either the input or output layer of the network. These hidden layers enable the neural network to extract the higher-order statistical features from its input. Neurons in hidden layers are correspondingly called *hidden neurons*. These hidden neurons have a function to intervene between external input and the network output in some useful manner.

The source nodes in the input layer supply respective elements of the activation pattern to constitute input signals applied to neurons in the first hidden layer. The output signals of neurons in the first hidden layer only are used as input signals to neurons in the second hidden layer. Generally, the output signals of neurons in each hidden layer only are used as input signals to neurons in the adjacent forward hidden layer. There is no connection among neurons in the same layer. Finally, the output signals of neurons in the last hidden layer only are used as input signals to neurons in the output layer. The set of output signals of the neurons in the output layer of the network constitute the overall response of the network to the activation pattern supplied by the source nodes in the input layer of the network.

If every neuron in each layer of the multilayer feedforward network is connected to every neuron in the next layer, then this kind of neural network is called *fully connected*. The simplest fully connected multilayer feedforward network is the network with one hidden layer and one output layer. If such network has m source nodes, h hidden neurons, and n output neurons, for the sake of brevity, this fully connected

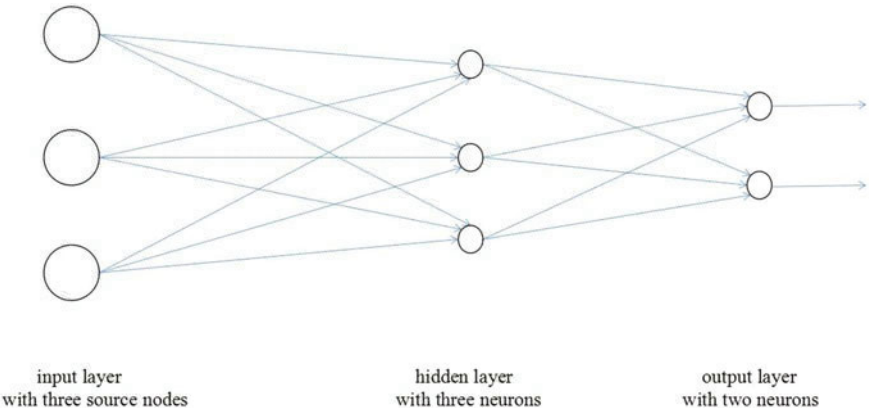

Fig. 1.2 Multilayer feedforward network with a single hidden layer (3-3-2 Network)

multilayer feedforward network is referred to as an $m - h - n$ network. In general, for a fully connected multilayer network with k hidden layers and one output layer, if it has m source nodes in the input layer, h_1 neurons in the first hidden layer, h_2 neurons in the second hidden layer, ..., h_k neurons in the kth hidden layer, and n output neurons, then it is referred to as an $m - h_1 - h_2 - \cdots - h_k - n$ network. If some synaptic connections are missing from the multilayer feedforward network, then the network is called *partially connected*.

1.1.2 Recurrent Networks

Recurrent networks have self-feedback loops or not, the feedback loops involve the use of particular branches composed of unit-delay elements, and recurrent networks also have hidden neurons or not. A recurrent neural network distinguishes from feedforward networks in that it has at least one feedback loop. This offers a lot of flexibility and can approximate arbitrary dynamical systems with arbitrary precision. The network with a single-loop feedback is called a *single-loop feedback network*.

Consider a single-loop feedback network. Denote its input signal by $x(n)$, internal signal by $\widetilde{x}(n)$, and output signal by $y(n)$, where $x(n)$, $\widetilde{x}(n)$, $y(n)$ are dependent on the discrete-time variable n. Assume that the network consists of a forward path $\mathbf{A}$ and a feedback path $\mathbf{B}$, where $\mathbf{A}$ and $\mathbf{B}$ are operators (see Fig. 1.3).

Assume that the output of the forward channel determines in part its own output through the feedback channel. Then, the input and output satisfy the following relationship:

$$y(n) = \mathbf{A}[\widetilde{x}(n)],$$
$$\widetilde{x}(n) = x(n) + \mathbf{B}[y(n)],$$

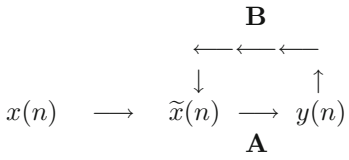

Fig. 1.3 Single-loop feedback system

where $\mathbf{A}[\widetilde{x}(n)]$ and $\mathbf{B}[y(n)]$ mean that the operators $\mathbf{A}$ and $\mathbf{B}$ act on $\widetilde{x}(n)$ and $y(n)$, respectively. Eliminating $\widetilde{x}(n)$ from both equations, we get

$$y(n) = \frac{\mathbf{A}}{1 - \mathbf{AB}}[x(n)], \tag{1.1.1}$$

where $\frac{\mathbf{A}}{1-\mathbf{AB}}$ is called a *closed-loop operator* and $\mathbf{AB}$ is called an *open-loop operator*. In general, $\mathbf{AB} \neq \mathbf{BA}$.

If the operator $\mathbf{A}$ is a fixed weight w and the operator $\mathbf{B}$ is a unit-delay operator $\mathbf{z}^{-1}$ whose output is delayed with respect to the input by one time unit, then the closed-loop operator becomes

$$\frac{\mathbf{A}}{1 - \mathbf{AB}} = \frac{w}{1 - w\mathbf{z}^{-1}} = \sum_{l=0}^{\infty} w^{l+1}\mathbf{z}^{-l}.$$

Substituting it into (1.1.1), we get

$$y(n) = \sum_{l=0}^{\infty} w^{l+1}\mathbf{z}^{-l}[x(n)],$$

where $\mathbf{z}^{-l}[x(n)]$ means that the operator $\mathbf{z}^{-l}$ acts on $x(n)$. Since $\mathbf{z}^{-1}$ is a unit-delay operator,

$$\mathbf{z}^{-l}[x(n)] = x(n - l),$$

Furthermore,

$$y(n) = \sum_{l=0}^{\infty} w^{l+1}x(n - l).$$

From this, the dynamic behavior of the single-loop feedback network with the fixed weight w and the unit-delay operator $\mathbf{z}^{-1}$ is determined by the weight w. When $|w| < 1$, the output signal $y(n)$ is *convergent* exponentially. In this case, the system is *stable*. When $|w| \geq 1$, the output signal $y(n)$ is *divergent*. In this case, the system is *unstable*. If $|w| = 1$, the divergence is linear. If $|w| > 1$, the divergence is exponential.

1.2 Perceptrons

The perceptron is a kind of neural networks that can decide whether an input belongs to some specific class. It was the first algorithmically described neural network. Perceptrons can be classified into Rosenblatt's perceptron and multilayer perceptrons.

1.2.1 Rosenblatt's Perceptron

Rosenblatt's perceptron is a network with m input source nodes and a single output neuron, a more general computational model than McCulloch–Pitts model. The perceptron belongs to basically a single-layer neural network and consists of a single neuron with adjustable synaptic weight and bias.

Rosenblatt's perceptron can be described mathematically by the pair of equations:

$$\begin{cases} v = \sum_{i=1}^{m} w_i x_i + b, \\ y = \varphi(v) \end{cases} \tag{1.2.1}$$

or by an equivalent equation:

$$y = \varphi\left(\sum_{i=1}^{m} w_i x_i + b \right),$$

where x_i $(i = 1, \ldots, m)$ are the input signals applied to the perceptron, w_i $(i = 1, \ldots, m)$ are the synaptic weights of the perceptron, b is the externally applied bias, v is called the *induced local field* (or linear combiner), φ is the activation function (or hard limiter), and y is the output signal of the perceptron (see Fig. 1.4). The Rosenblatt's perceptron consists of a linear combiner followed by a hard limiter, and the hard limiter $\varphi(v)$ determines the output of the perceptron in terms of the induced local field v.

Let $x_0 = 1$ and $w_0 = b$ be the input and the weight of a new synapse. An equivalent form of (1.2.1) is

$$\begin{cases} v = \sum_{i=0}^{m} w_i x_i, \\ y = \varphi(v) \end{cases} \quad \text{or} \quad y = \varphi\left(\sum_{i=0}^{m} w_i x_i \right).$$

The activation function in Rosenblatt's perceptron is a threshold function as follows:

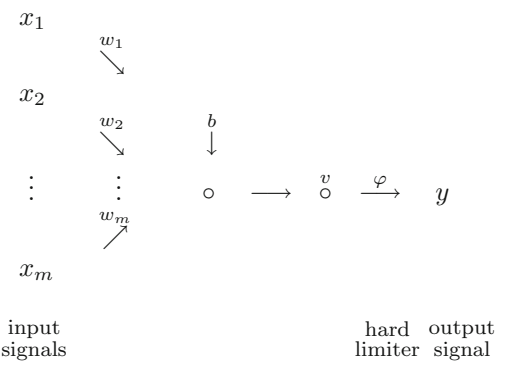

Fig. 1.4 Rosenblatt's perceptron

(a) The Heaviside function is defined as

$$\varphi(v) = \begin{cases} 1 \text{ if } v > 0, \\ 0 \text{ if } v \leq 0. \end{cases}$$

If v is the induced local field and $v = \sum_{i=1}^{m} w_i x_i + b$, then the corresponding output is expressed as

$$y = \begin{cases} 1 \text{ if } v > 0, \\ 0 \text{ if } v \leq 0. \end{cases}$$

(b) The signum function is defined as

$$\varphi(v) = \begin{cases} 1 & \text{if } v > 0, \\ 0 & \text{if } v = 0, \\ -1 \text{ if } v < 0. \end{cases}$$

If v is the induced local field and $v = \sum_{i=1}^{m} w_i x_i + b$, then the corresponding output is expressed as

$$y = \begin{cases} 1 & \text{if } v > 0, \\ 0 & \text{if } v = 0, \\ -1 \text{ if } v < 0. \end{cases}$$

Based on the feature of the harder limiter, Rosenblatt's perceptron can classify correctly the set of externally applied stimuli $x_1, \ldots, x_m$ into one of two classes. The decision rule for the classification is as follows:

- If the input to the hard limiter $v > 0$, when the activation function is the Heaviside function (or the signum function), the points represented by stimuli $x_1, \ldots, x_m$ are assigned to Class 1;
- If the input to the hard limiter $v < 0$, when the activation function is Heaviside function (or the signum function), the points represented by stimuli $x_1, \ldots, x_m$ are assigned to Class 0 (or Class -1).

Here, Class k is the set consisting of points represented by the stimuli $x_1, \ldots, x_m$ having the output $y = k$.

The input to the hard limiter $v = 0$, i.e., the hyperplane

$$\sum_{i=1}^{m} w_i x_i + b = 0$$

is referred to as the *decision boundary* of two classes. The simplest Rosenblatt's perceptron with two source nodes and a single output neuron can classify the set of externally applied stimuli x_1, x_2 into one of two classes, and the decision boundary of these two classes is a straight line:

$$w_1 x_1 + w_2 x_2 + b = 0, \tag{1.2.2}$$

where $w_i (i = 1, 2)$ and b are the synaptic weights and bias of the perceptron, respectively. For example, assume that $w_1 = 1$, $w_2 = 0.5$, $b = -0.5$, and the sampling set of externally applied stimuli x_1, x_2 consists of seven points:

$$\begin{array}{llll} \mathbf{x}^{(1)} = (-1, 2), & \mathbf{x}^{(2)} = (1, 2), & \mathbf{x}^{(3)} = (2, -1), & \\ \mathbf{x}^{(4)} = (1, 1), & \mathbf{x}^{(5)} = (-2, 1), & \mathbf{x}^{(6)} = (-1, -1), & \mathbf{x}^{(7)} = (2, 0). \end{array}$$

By (1.2.2), the decision boundary of two classes is $x_1 + 0.5x_2 - 0.5 = 0$. It is seen that four points $\mathbf{x}^{(2)}, \mathbf{x}^{(3)}, \mathbf{x}^{(4)}, \mathbf{x}^{(7)}$ lie above the straight line and the rest $\mathbf{x}^{(1)}, \mathbf{x}^{(5)}, \mathbf{x}^{(6)}$ lie below the straight line. The decision rule for the classification shows that these seven points can be classified into two classes.

1.2.2 Multilayer Perceptron

The architecture of multilayer perceptrons is very different from the single-layer perceptron. It is a perceptron consisting of an input layer of m source nodes, i hidden layers of h_i neurons, and an output layer of n neurons. Each neuron of the network has a differentiable nonlinear activation function. The architecture of a fully connected multilayer perceptron is that a neuron node in any layer of the network is connected to all neuron nodes in the previous layer and the signal flow through the network progresses in a forward direction from left to right and on a layer-by-layer basis.

The activation function used commonly in the multilayer perceptron is the sigmoid function. The sigmoid function is mathematically convenient and is close to linear near the origin while saturating rather quickly when getting away from the origin. This allows multilayer perceptrons to model well both strongly and mildly nonlinear relations. The logistic function and the hyperbolic tangent function are two popular sigmoid functions. The logistic function is defined as

$$\varphi(v) = \frac{1}{1+e^{-av}} \qquad (a > 0),$$

where a is its slop parameter. Logistic functions of different slopes can be obtained by varying this parameter. The logistic function is differentiable, and its derivative is

$$\varphi'(v) = \frac{ae^{-av}}{(1+e^{-av})^2} = \frac{a}{1+e^{-av}}\left(1 - \frac{1}{1+e^{-av}}\right) = a\varphi(v)(1-\varphi(v)).$$

The hyperbolic tangent function is defined as

$$\varphi(v) = a\tanh(bv) \qquad (a > 0,\ b > 0).$$

The hyperbolic tangent function is also differentiable, and its derivative is

$$\begin{aligned}\varphi'(v) &= ab\,\mathrm{sech}^2(bv) = ab(1-\tanh^2(bv)) = \tfrac{b}{a}(a^2 - \varphi^2(v)) \\ &= \tfrac{b}{a}(a-\varphi(v))(a+\varphi(v)).\end{aligned}$$

The process of the multilayer perceptron includes forward propagation of function signals and backward propagation of error signals which are identified. *Forward propagation of function signals* is such a process that an input signal comes in at the input end of the network, propagates forward neuron by neuron through the network, and emerges at the output end of the network as an output signal. *Backward propagation of error signals* is such a process that the error signal originates at an output neuron of the network and propagates backward layer-by-layer through the network.

Due to one or more hidden layers, the multilayer perceptron can classify nonlinearly separable patterns, while the single-layer perceptron cannot.

Example 1.2.1 Assume that a sampling set consists of four points:

$$\begin{aligned}\mathbf{x}^{(1)} &= (0,0), \qquad \mathbf{x}^{(2)} = (0,1), \\ \mathbf{x}^{(3)} &= (1,0), \qquad \mathbf{x}^{(4)} = (1,1).\end{aligned}$$

The XOR problem is to use a perceptron to classify these four points into two classes such that points $\mathbf{x}^{(1)}$, $\mathbf{x}^{(4)}$ are assigned to a class, and points $\mathbf{x}^{(2)}$, $\mathbf{x}^{(3)}$ are assigned to another class. There is no possibility to solve the XOR problem using any single-layer perceptron. In fact, assume that a following single-layer perceptron can solve it:

$$\begin{cases} v = w_1x_1 + w_2x_2 + b, \\ y = \varphi(v), \end{cases}$$

where $w_i (i = 1, 2)$ and b are synaptic weights and bias, respectively, and the activation function φ is the Heaviside function. Since $\mathbf{x}^{(1)} = (0,0)$ and $\mathbf{x}^{(4)} = (1,1)$ are assigned to Class 0, it is clear that $b < 0$ and $w_1 + w_2 + b < 0$. Adding them together gives

$$w_1 + w_2 + 2b < 0. \tag{1.2.3}$$

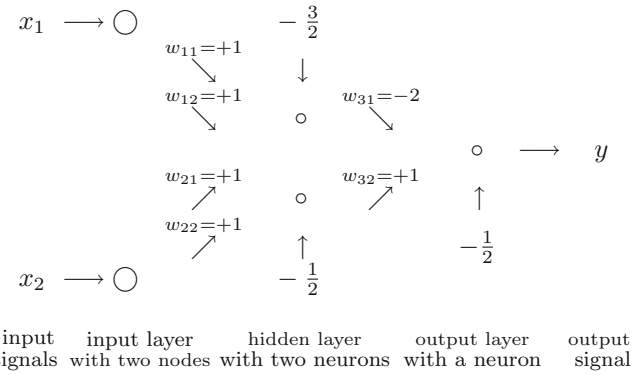

Fig. 1.5 Touretzky–Pomerleau perceptron

Since $\mathbf{x}^{(2)} = (0, 1)$ and $\mathbf{x}^{(3)} = (1, 0)$ are assigned to Class 1, then

$$\begin{aligned} w_2 + b &> 0, \\ w_1 + b &> 0. \end{aligned}$$

Adding them together, we get

$$w_1 + w_2 + 2b > 0.$$

This is contrary to (1.2.3). Thus, there is no possibility to solve the XOR problem using a single-layer perceptron.

Touretzky–Pomerleau perceptron is a multilayer perceptron (see Fig. 1.5), where each "$\circ$" represents a neuron and each neuron has Heaviside function as the activation function. Touretzky–Pomerleau perceptron can solve the XOR problem given by Example 1.2.1.

There are three neurons in Touretzky–Pomerleau perceptron, two of them are hidden neurons in the hidden layer and the remainder one is the output neuron in the output layer. For the top hidden neuron, the synaptic weights and bias are, respectively,

$$\begin{aligned} w_{11} &= w_{12} = 1, \\ b_1 &= -\tfrac{3}{2}. \end{aligned}$$

The straight line

$$l_1 : \quad x_1 + x_2 - \frac{3}{2} = 0.$$

is the decision boundary formed by the top hidden neuron. It is clear that the point $\mathbf{x}^{(4)}$ lying above the line l_1 is assigned to Class 1, and points $\mathbf{x}^{(1)}, \mathbf{x}^{(2)}, \mathbf{x}^{(3)}$ lying below the line l_1 are assigned to Class 0.

For the bottom hidden neuron, the synaptic weights and bias are, respectively,

$$\begin{aligned} &w_{21} = w_{22} = 1, \\ &b_2 = -\tfrac{1}{2}. \end{aligned}$$

The straight line

$$l_2 : \quad x_1 + x_2 - \frac{1}{2} = 0$$

is the decision boundary formed by the bottom hidden neuron. The points $\mathbf{x}^{(2)}$, $\mathbf{x}^{(3)}$, $\mathbf{x}^{(4)}$ lying above the line l_2 are assigned to Class 1, and the point $\mathbf{x}^{(1)}$ lying below the line l_2 is assigned to Class 0.

For the output neuron, the synaptic weights and bias are, respectively,

$$\begin{aligned} &w_{31} = -2, \qquad w_{32} = 1, \\ &b_3 = -\tfrac{1}{2}. \end{aligned}$$

The output neuron constructs a linear combination of decision boundaries formed by two hidden neurons. The decision boundary formed by the output neuron is a straight line:

$$-2y_1 + y_2 - \frac{1}{2} = 0$$

where y_1 is the output from the top hidden neuron and y_2 is the output from the bottom neuron. The decision rule for the classification is that a point that lies above the line l_1 or below the line l_2 is assigned to Class 0 and a point that lies both below the line l_1 and above the line l_2 is assigned to Class 1. Therefore, according to the decision rule for the classification, the points $\mathbf{x}^{(1)}$, $\mathbf{x}^{(4)}$ are assigned to Class 0 and the points $\mathbf{x}^{(2)}$, $\mathbf{x}^{(3)}$ are assigned to Class 1. So the XOR problem for four points $\mathbf{x}^{(i)}$ is solved using the Touretzky–Pomerleau perceptron.

1.3 Linear Network and Bayes Classifier

Linear neural network distinguishes from Rosenblatt's perceptron in that activation function is a linear function. So the output of the linear network may be any value. A linear network of a neuron can be described by a pair of equations:

$$\begin{cases} v = \sum_{i=1}^{m} w_i x_i + b, \\ y = \varphi(v) = v \end{cases}$$

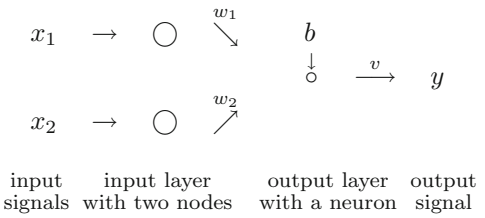

Fig. 1.6 Linear network with double input signals

or simply by an equation:

$$y = \sum_{i=1}^{m} w_i x_i + b = \sum_{i=0}^{m} w_i x_i,$$

where $x_0 = 1$, x_i $(l = 1, \ldots, m)$ are the input signals, w_i $(l = 1, \ldots, m)$ are the synaptic weights, $w_0 = b$ is the bias, and y is the output signal of the neuron. Similar to Rosenblatt's perceptrons, linear neural network is used for classifying linearly separable patterns. For example, a linear network of a neuron with two source nodes is shown as in Fig. 1.6, where x_1, x_2 are two input signals, y is the output signal, w_1, w_2 are two synaptic weights, b is the bias, and $\circ$ represents the neuron. This network can be described by a pair of equations:

$$\begin{cases} v = w_1 x_1 + w_2 x_2 + b, \\ y = v \end{cases}$$

or simply by an equation: $y = w_1 x_1 + w_2 x_2 + b$.
The corresponding decision boundary is a straight line $w_1 x_1 + w_2 x_2 + b = 0$, and the decision rule for classification is that all points that lie above the straight line are assigned to one class and all points that lie below the straight line are assigned to another class.

Below, we discuss *the Bayes classifier*. Consider a two-class problem represented by classes $\mathcal{B}_i$ in the subspace $\mathbb{R}_i$ $(i = 1, 2)$, where $\widetilde{\mathcal{R}} = \mathcal{R}_1 + \mathcal{R}_2$. Denote by $p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_i)$ $(i = 1, 2)$ the conditional probability density function of a random vector $\mathbf{X}$, given that the observation vector $\mathbf{x}$ is drawn from subspace $\mathbb{R}_i$. In the Bayes classifier, Van Trees defined an *average risk* (AR) of the two-class problem by

$$\begin{aligned} \mathrm{AR} = {} & c_{11} p_1 \int_{\mathcal{R}_1} p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_1) \mathrm{d}\mathbf{X} + c_{22} p_2 \int_{\mathcal{R}_2} p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_2) \mathrm{d}\mathbf{X} \\ & + c_{21} p_1 \int_{\mathcal{R}_2} p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_1) \mathrm{d}\mathbf{X} + c_{12} p_2 \int_{\mathcal{R}_1} p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_2) \mathrm{d}\mathbf{X}, \end{aligned}$$

where c_{ij} is the cost of deciding in favor of class $\mathcal{B}_i$ and p_i is the prior probability of the observation vector $\mathbf{x}$, and $p_1 + p_2 = 1$. Note that

$$\widetilde{\mathcal{R}} = \mathcal{R}_1 + \mathcal{R}_2.$$

The equivalent form of the average risk is

$$\begin{aligned}\mathrm{AR} &= c_{11}p_1\int_{\mathcal{R}_1} p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_1)\mathrm{d}\mathbf{X} + c_{22}p_2\int_{\widetilde{\mathcal{R}}-\mathcal{R}_1} p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_2)\mathrm{d}\mathbf{X}\\ &\quad + c_{21}p_1\int_{\widetilde{\mathcal{R}}-\mathcal{R}_1} p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_1)\mathrm{d}\mathbf{X} + c_{12}p_2\int_{\mathcal{R}_1} p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_2))\mathrm{d}\mathbf{X}\\ &= c_{11}p_1\int_{\mathcal{R}_1} p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_1)\mathrm{d}\mathbf{X} + c_{22}p_2\int_{\widetilde{\mathcal{R}}} p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_2)\mathrm{d}\mathbf{X} - c_{22}p_2\int_{\mathcal{R}_1} p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_2)\mathrm{d}\mathbf{X}\\ &\quad + c_{21}p_1\int_{\widetilde{\mathcal{R}}} p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_1)\mathrm{d}\mathbf{X} - c_{21}p_1\int_{\mathcal{R}_1} p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_1)\mathrm{d}\mathbf{X} + c_{12}p_2\int_{\mathcal{R}_1} p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_2))\mathrm{d}\mathbf{X}.\end{aligned}$$

From this and

$$\begin{aligned}&\textstyle\int_{\widetilde{\mathcal{R}}} p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_1)\mathrm{d}\mathbf{X} = 1,\\ &\textstyle\int_{\widetilde{\mathcal{R}}} p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_2)\mathrm{d}\mathbf{X} = 1,\end{aligned}$$

it follows that the average risk is

$$\mathrm{AR} = c_{21}p_1 + c_{22}p_2 + \int_{\mathcal{R}_1}[p_2(c_{12}-c_{22})p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_2) - p_1(c_{21}-c_{11})p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_1)]\mathrm{d}\mathbf{X},$$

where $c_{21}p_1 + c_{22}p_2$ represents a fixed cost. The Bayes classifier requires that the integrand of the last integral is greater than zero, i.e., $p_2(c_{12}-c_{22})p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_2) - p_1(c_{21}-c_{11})p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_1) > 0$ or

$$\frac{p_2(c_{12}-c_{22})}{p_1(c_{21}-c_{11})} > \frac{p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_1)}{p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_2)}.$$

The quantities

$$\Lambda(\mathbf{X}) = \frac{p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_1)}{p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_2)}, \qquad \xi = \frac{p_2(c_{12}-c_{22})}{p_1(c_{21}-c_{11})}.$$

are called *likelihood ratio* and *threshold* of the test, respectively. The likelihood ratio $\Lambda(\mathbf{x})$ and the threshold ξ are both positive. Taking the logarithm of $\Lambda(\mathbf{x})$,

$$\log\Lambda(\mathbf{X}) = \log p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_1) - \log p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_2). \tag{1.3.1}$$

The quantity $\log\Lambda(\mathbf{X})$ is called $\log-$*likelihood ratio*. Since the logarithm function is a monotonic function, it is more convenient to compute the $\log-$likelihood ratio than the likelihood ratio.

Consider a special two-class problem:

$$\begin{aligned}&\text{Class } \mathcal{B}_1\text{: } E[\mathbf{X}] = \mu_1,\\ &\qquad\qquad E[(\mathbf{X}-\mu_1)(\mathbf{X}-\mu_1)^T] = C;\\ &\text{Class } \mathcal{B}_2\text{: } E[\mathbf{X}] = \mu_2,\\ &\qquad\qquad E[(\mathbf{X}-\mu_2)(\mathbf{X}-\mu_2)^T] = C,\end{aligned} \tag{1.3.2}$$

where $\mathbf{X}$ is a random vector. In the two-class problem, the mean values of the random vector are different but their covariance matrix of the random vector is the same. Denote by C^{-1} the inverse matrix of the covariance matrix C. Then, the conditional probability density function may be represented as the multivariate Gaussian distribution:

$$p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_i) = \frac{1}{(2\pi)^{\frac{m}{2}}(\det(C))^{\frac{1}{2}}} \exp\left(-\frac{1}{2}(\mathbf{x}-\mu_i)^T C^{-1}(\mathbf{x}-\mu_i)\right) \qquad (i=1,2),$$

where m is the dimensionality of the observation vector $\mathbf{x}$ and $\det(C)$ represents the determinant of the matrix C. Assume further in the average risk that

$$p_1 = p_2 = \frac{1}{2}, \quad c_{12} = c_{21}, \quad c_{11} = c_{22} = 0.$$

Note that

$$\begin{aligned}
\log p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_1) &= -\tfrac{1}{2}(\mathbf{x}-\mu_1)^T C^{-1}(\mathbf{x}-\mu_1) - \log((2\pi)^{\frac{m}{2}}(\det(C))^{\frac{1}{2}}),\\
\log p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_2) &= -\tfrac{1}{2}(\mathbf{x}-\mu_2)^T C^{-1}(\mathbf{x}-\mu_2) - \log((2\pi)^{\frac{m}{2}}(\det(C))^{\frac{1}{2}}).
\end{aligned}$$

By (1.3.1), the log −likelihood ratio is

$$\begin{aligned}
\log \Lambda(\mathbf{X}) &= \log p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_1) - \log p_{\mathbf{X}}(\mathbf{X}|\mathcal{B}_2)\\
&= \left(-\tfrac{1}{2}(\mathbf{x}-\mu_1)^T C^{-1}(\mathbf{x}-\mu_1) - \log((2\pi)^{\frac{m}{2}}(\det(C))^{\frac{1}{2}})\right)\\
&\quad - \left(-\tfrac{1}{2}(\mathbf{x}-\mu_2)^T C^{-1}(\mathbf{x}-\mu_2) - \log((2\pi)^{\frac{m}{2}}(\det(C))^{\frac{1}{2}})\right).
\end{aligned}$$

A direct computation shows that

$$\begin{aligned}
\log \Lambda(\mathbf{X}) &= -\tfrac{1}{2}(\mathbf{x}^T-\mu_1^T)C^{-1}(\mathbf{x}-\mu_1) + \tfrac{1}{2}(\mathbf{x}^T-\mu_2^T)C^{-1}(\mathbf{x}-\mu_2)\\
&= -\tfrac{1}{2}\mathbf{x}^T C^{-1}\mathbf{x} + \tfrac{1}{2}\mu_1^T C^{-1}\mathbf{x} + \tfrac{1}{2}\mathbf{x}^T C^{-1}\mu_1 - \tfrac{1}{2}\mu_1^T C^{-1}\mu_1\\
&\quad + \tfrac{1}{2}\mathbf{x}^T C^{-1}\mathbf{x} - \tfrac{1}{2}\mu_2^T C^{-1}\mathbf{x} - \tfrac{1}{2}\mathbf{x}^T C^{-1}\mu_2 + \tfrac{1}{2}\mu_2^T C^{-1}\mu_2\\
&= \tfrac{1}{2}(\mu_1-\mu_2)^T C^{-1}\mathbf{x} + \tfrac{1}{2}\mathbf{x}^T C^{-1}(\mu_1-\mu_2) + \tfrac{1}{2}(\mu_2^T C^{-1}\mu_2 - \mu_1^T C^{-1}\mu_1)\\
&= (\mu_1-\mu_2)^T C^{-1}\mathbf{x} + \tfrac{1}{2}(\mu_2^T C^{-1}\mu_2 - \mu_1^T C^{-1}\mu_1).
\end{aligned} \tag{1.3.3}$$

By $c_{12} = c_{21}$, $c_{11} = c_{22} = 0$, and $p_1 = p_2 = \frac{1}{2}$, the threshold of the test is

$$\xi = \frac{p_2(c_{12}-c_{22})}{p_1(c_{21}-c_{11})} = \frac{\frac{1}{2}(c_{12}-0)}{\frac{1}{2}(c_{21}-0)} = 1.$$

In (1.3.3), let

$$\begin{aligned}
y &= \log \Lambda(\mathbf{X}),\\
W^T &= (\mu_1-\mu_2)^T C^{-1},\\
b &= \tfrac{1}{2}(\mu_2^T C^{-1}\mu_2 - \mu_1^T C^{-1}\mu_1).
\end{aligned} \tag{1.3.4}$$

Then, Bayes classifier for multivariate Gaussian distribution, or simply, *Gaussian-distribution classifier*, is

$$y = W^T \mathbf{x} + b. \tag{1.3.5}$$

This equation shows that the Gaussian-distribution classifier is a linear network with the synaptic weight W and the bias b. It is seen from (1.3.5) that the decision boundary of the special two classes $\mathcal{B}_1, \mathcal{B}_2$ is the hyperplane:

$$W^T \mathbf{x} + b = 0.$$

In the Gaussian-distribution classifier, assume that the covariance matrix C is given by $C = aI$, where $a > 0$ is a positive constant and I is the identity matrix. We will find the synaptic-weight vector and bias as well as the representation of the Gaussian-distribution classifier.

Note that $C^{-1} = (aI)^{-1} = \frac{1}{a} I^{-1} = \frac{1}{a} I$. By (1.3.4), the weight vector is equal to

$$W = C^{-1}(\mu_1 - \mu_2) = \frac{1}{a} I(\mu_1 - \mu_2) = \frac{1}{a}(\mu_1 - \mu_2)$$

and the bias is equal to

$$b = \frac{1}{2}(\mu_2^T C^{-1} \mu_2 - \mu_1^T C^{-1} \mu_1) = \frac{1}{2a}(\mu_2^T \mu_2 - \mu_1^T \mu_1).$$

By (1.3.5), the Gaussian-distribution classifier becomes

$$y = W^T \mathbf{x} + b = \frac{1}{a}\left(\mu_1^T - \mu_2^T\right)\mathbf{x} + \frac{1}{2a}(\mu_2^T \mu_2 - \mu_1^T \mu_1).$$

Example 1.3.1 In the one-dimensional case, consider the following two-class problem:

$$\begin{aligned} \text{Class } \mathcal{B}_1\text{: } & E[X] = \mu_1, \\ & E[(X - \mu_1)^2] = C, \\ \text{Class } \mathcal{B}_2\text{: } & E[X] = \mu_2, \\ & E[(X - \mu_2)^2] = C, \end{aligned} \tag{1.3.6}$$

where X is a random variable and μ_1, μ_2, C are all real numbers. So $\mu_1^T = \mu_1$, $\mu_2^T = \mu_2$, and $C^{-1} = \frac{1}{C}$. By (1.3.4), the synaptic weight and the bias are all real numbers and

$$\begin{aligned} W &= C^{-1}(\mu_1 - \mu_2) = \tfrac{\mu_1 - \mu_2}{C}, \\ b &= \tfrac{1}{2}(\mu_2^T C^{-1} \mu_2 - \mu_1^T C^{-1} \mu_1) = \tfrac{1}{2C}(\mu_2^2 - \mu_1^2). \end{aligned}$$

By (1.3.5), the univariate Gaussian-distribution classifier is

$$y = W^T x + b = \frac{\mu_1 - \mu_2}{C} x + \frac{1}{2C}(\mu_2^2 - \mu_1^2)$$

and the decision boundary of the two-class problem (1.3.6) is a point:

$$x^* = -\frac{\frac{1}{2C}(\mu_2^2 - \mu_1^2)}{\frac{\mu_1 - \mu_2}{C}} = \frac{\mu_1 + \mu_2}{2}.$$

Assume that $\mu_1 = -10$, $\mu_2 = 10$, and $C = 1$. For the two-class problem (1.3.6), the synaptic weight $W = -20$ and the bias $b = 0$, the univariate Gaussian-distribution classifier is $y = -20x$, and the decision boundary is $x = 0$. The point $x(x > 0)$ and the point $x(x < 0)$ are assigned to Class $\mathcal{B}_1$ and $\mathcal{B}_2$, respectively.

Example 1.3.2 In the two-dimensional case, the means $\mu_1,\ \mu_2$, and the inverse matrix of the covariance matrix are denoted by

$$\mu_1 = (\mu_{11},\ \mu_{12})^T, \qquad \mu_2 = (\mu_{21},\ \mu_{22})^T,$$
$$C^{-1} = \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix}.$$

By (1.3.4), the synaptic weight $W = (W_1, W_2)^T$ is

$$\begin{aligned} W = C^{-1}(\mu_1 - \mu_2) &= \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} \begin{pmatrix} \mu_{11} - \mu_{21} \\ \mu_{12} - \mu_{22} \end{pmatrix} \\ &= \begin{pmatrix} c_{11}(\mu_{11} - \mu_{21}) + c_{12}(\mu_{12} - \mu_{22}) \\ c_{21}(\mu_{11} - \mu_{21}) + c_{22}(\mu_{12} - \mu_{22}) \end{pmatrix}. \end{aligned}$$

Let $\mathbf{x} = (x_1, x_2)^T$. Then,

$$\begin{aligned} W^T\mathbf{x} &= (c_{11}(\mu_{11} - \mu_{21}) + c_{12}(\mu_{12} - \mu_{22}),\ c_{21}(\mu_{11} - \mu_{21}) + c_{22}(\mu_{12} - \mu_{22})) \begin{pmatrix} x_1 \\ x_2 \end{pmatrix} \\ &= c_{11}(\mu_{11} - \mu_{21})x_1 + c_{12}(\mu_{12} - \mu_{22})x_1 + c_{21}(\mu_{11} - \mu_{21})x_2 + c_{22}(\mu_{12} - \mu_{22})x_2 \\ &= (\mu_{11} - \mu_{21})(c_{11}x_1 + c_{21}x_2) + (\mu_{12} - \mu_{22})(c_{12}x_1 + c_{22}x_2). \end{aligned}$$

By (1.3.4), the bias is

$$b = \frac{1}{2}(\mu_2^T C^{-1}\mu_2 - \mu_1^T C^{-1}\mu_1).$$

Two terms $\mu_2^T C^{-1}\mu_2$ and $\mu_1^T C^{-1}\mu_1$ are computed, respectively, as follows:

$$\begin{aligned} \mu_2^T C^{-1}\mu_2 &= (\mu_{21}, \mu_{22}) \begin{pmatrix} c_{11} & c_{12} \\ c_{21} & c_{22} \end{pmatrix} \begin{pmatrix} \mu_{21} \\ \mu_{22} \end{pmatrix} \\ &= (\mu_{21}c_{11} + \mu_{22}c_{21},\ \mu_{21}c_{12} + \mu_{22}c_{22}) \begin{pmatrix} \mu_{21} \\ \mu_{22} \end{pmatrix} \\ &= (\mu_{21}c_{11} + \mu_{22}c_{21})\mu_{21} + (\mu_{21}c_{12} + \mu_{22}c_{22})\mu_{22} \\ &= \mu_{21}^2 c_{11} + \mu_{21}\mu_{22}c_{21} + \mu_{21}\mu_{22}c_{12} + \mu_{22}^2 c_{22}. \end{aligned}$$

$$\begin{aligned}\mu_1^T C^{-1}\mu_1 &= (\mu_{11},\mu_{12})\begin{pmatrix}c_{11} & c_{12}\\ c_{21} & c_{22}\end{pmatrix}\begin{pmatrix}\mu_{11}\\ \mu_{12}\end{pmatrix}\\ &= (\mu_{11}c_{11}+\mu_{12}c_{21},\ \mu_{11}c_{12}+\mu_{12}c_{22})\begin{pmatrix}\mu_{11}\\ \mu_{12}\end{pmatrix}\\ &= (\mu_{11}c_{11}+\mu_{12}c_{21})\mu_{11}+(\mu_{11}c_{12}+\mu_{12}c_{22})\mu_{12}\\ &= \mu_{11}^2c_{11}+\mu_{11}\mu_{22}c_{11}+\mu_{11}\mu_{12}c_{12}+\mu_{12}^2c_{22}.\end{aligned}$$

So the bias is

$$\begin{aligned}b &= \tfrac{1}{2}(\mu_{21}^2c_{11}+\mu_{21}\mu_{22}c_{21}+\mu_{21}\mu_{22}c_{12}+\mu_{22}^2c_{22})\\ &\quad -\tfrac{1}{2}(\mu_{11}^2c_{11}+\mu_{11}\mu_{12}c_{21}+\mu_{11}\mu_{12}c_{12}+\mu_{12}^2c_{22})\\ &= \tfrac{1}{2}((\mu_{21}^2-\mu_{11}^2)c_{11}+(\mu_{21}\mu_{22}-\mu_{11}\mu_{12})(c_{21}+c_{12})+(\mu_{22}^2-\mu_{12}^2)c_{22})\end{aligned}$$

By (1.3.5), the bivariate Gaussian-distribution classifier is

$$\begin{aligned}\mathbf{y} &= W^T\mathbf{x}+b\\ &= (\mu_{11}-\mu_{21})(c_{11}x_1+c_{21}x_2)+(\mu_{12}-\mu_{22})(c_{12}x_1+c_{22}x_2)\\ &\quad +\tfrac{1}{2}(\mu_{21}^2-\mu_{11}^2)c_{11}+\tfrac{1}{2}(\mu_{21}\mu_{22}-\mu_{11}\mu_{12})(c_{21}+c_{12})+\tfrac{1}{2}(\mu_{22}^2-\mu_{12}^2)c_{22}.\end{aligned}$$

1.4 Radial Basis Function Network

Radial basis function network is derived from the theory of function approximation and interpolation. It uses radial basis functions as activation functions.

1.4.1 Radial Basis Function

For a given set of distinct points $\mathbf{x}_1,\ldots,\mathbf{x}_N\in\mathbb{R}^m$, the radial basis function technique is to find a function $F(\mathbf{x})$ that has the form:

$$F(\mathbf{x})=\sum_{j=1}^{N}w_j\varphi(\|\ \mathbf{x}-\mathbf{x}_j\ \|)=\sum_{j=1}^{N}w_j\varphi(\mathbf{x},\mathbf{x}_j),\tag{1.4.1}$$

where $w_j(j=1,\ldots,N)$ are components of the weight vector $\mathbf{w}$, $\{\varphi(\mathbf{x},\mathbf{x}_j)\}_{j=1,\ldots,N}$ is a set of *radial basis functions*, and $\mathbf{x}_j$ is the *center* of the radial basis function $\varphi(\mathbf{x},\mathbf{x}_j)=\varphi(\|\ \mathbf{x}-\mathbf{x}_j\ \|)$; here, $\|\cdot\|$ is the Euclidean distance.

The radial basis functions used widely are as follows.

(a) Gaussian function:

$$\varphi(x)=\exp\left(-\frac{x^2}{2\sigma^2}\right)\quad\text{for some }\sigma>0\text{ and }x\in\mathbb{R},$$

where σ is the width of the Gaussian function.

(b) Multiquadric function: $\varphi(x) = (x^2 + c^2)^{\frac{1}{2}}$ for some $c > 0$ and $x \in \mathbb{R}$.

(c) Inverse multiquadric function: $\varphi(x) = (x^2 + c^2)^{-\frac{1}{2}}$ for some $c > 0$ and $x \in \mathbb{R}$.

1.4.2 Interpolation

The interpolation technique is used for finding the weight vector $\mathbf{w}$.

Given a set of distinct points $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^m$ and a corresponding set of real numbers $d_1, \ldots, d_N \in \mathbb{R}$. The interpolation problem is to seek a function $F : \mathbb{R}^m \to \mathbb{R}$ satisfying the interpolation condition:

$$F(\mathbf{x}_i) = d_i \qquad (i = 1, \ldots, N), \tag{1.4.2}$$

where $F(\mathbf{x})$ is called the *interpolation surface* (or the *interpolation function*). The interpolation surface $F(\mathbf{x})$ is constrained to pass through all the training data points $\{\mathbf{x}_i, d_i\}_{i=1,\ldots,N}$, where N is called the *size of the training sample*. The combination of (1.4.1) and (1.4.2) gives

$$d_i = \sum_{j=1}^{N} w_j \varphi(\mathbf{x}_i, \mathbf{x}_j) \qquad (i = 1, \ldots, N).$$

In more detail,

$$\begin{aligned} d_1 &= \sum_{j=1}^{N} w_j \varphi(\mathbf{x}_1, \mathbf{x}_j), \\ d_2 &= \sum_{j=1}^{N} w_j \varphi(\mathbf{x}_2, \mathbf{x}_j), \\ &\vdots \\ d_N &= \sum_{j=1}^{N} w_j \varphi(\mathbf{x}_N, \mathbf{x}_j). \end{aligned} \tag{1.4.3}$$

This is a system of N equations with N unknown weights $w_j (j = 1, \ldots, N)$.

Let $\varphi_{ij} = \varphi(\mathbf{x}_i, \mathbf{x}_j)$ $(i, j = 1, \ldots, N)$. Then, the system (1.4.3) with N unknown w_j can be rewritten in the matrix form

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

where N is the size of the training sample.

Let

$$\Phi = \begin{pmatrix} \varphi_{11} & \varphi_{12} & \cdots & \varphi_{1N} \\ \varphi_{21} & \varphi_{22} & \cdots & \varphi_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \varphi_{N1} & \varphi_{N2} & \cdots & \varphi_{NN} \end{pmatrix}, \quad \mathbf{w} = \begin{pmatrix} w_1 \\ w_2 \\ \vdots \\ w_N \end{pmatrix}, \quad \mathbf{d} = \begin{pmatrix} d_1 \\ d_2 \\ \vdots \\ d_N \end{pmatrix}.$$

Then, the system (1.4.3) has the compact form

$$\Phi \mathbf{w} = \mathbf{d}, \tag{1.4.4}$$

where the matrix Φ is called the *interpolation matrix* and the vectors $\mathbf{w}$ and $\mathbf{d}$ are the *linear weight vector* and *desired response vector*, respectively. Micchlli Interpolation Theorem (see Chap. 2) shows that the interpolation matrix Φ is non-singular when $\{\mathbf{x}_i\}_{i=1,\ldots,N}$ is a set of distinct points in $\mathbb{R}^{m_0}$. Therefore, its inverse matrix Φ^{-1} exists. So the weight vector $\mathbf{w}$ is given by $\mathbf{w} = \Phi^{-1}\mathbf{d}$.

The *architecture* of radial basis function network is a feedforward network with layered structure. For example, the architecture of a radial basis function network with an input layer, a single hidden layer, and an output layer consisting of a single unit is described as follows.

Given a set of N distinct points $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^m$, i.e., the size of the input layer is m. The single hidden layer consists of N computation units. Each computational unit is described by a radial basis function:

$$\varphi(\mathbf{x}, \mathbf{x}_j) = \varphi(\| \mathbf{x} - \mathbf{x}_j \|) \qquad (j = 1, \ldots, N),$$

where the jth input data point $\mathbf{x}_j$ is the center of the radial basis function and $\mathbf{x}$ is the signal applied to the input layer. The connections between the source nodes and hidden units are direct connections with no weights. The output layer consists of a single computational unit. The size of the output layer is 1. It is characterized by the weight vector $\mathbf{w} = (w_1, \ldots, w_N)$, and the approximating function is

$$F(\mathbf{x}) = \sum_{j=1}^{N} w_j \varphi(\mathbf{x}, \mathbf{x}_j).$$

However, in practice, since the training sample $\{\mathbf{x}_i, d_i\}_{i=1,\ldots,N}$ is often noisy, it could be wasteful of computational resources to have a hidden layer of the same size as the input layer. The size K of the hidden layer is required to be less than N, and then the corresponding approximating function is the sum of K weighted radial basis functions.