35. Radial Basis Functions for Data Mining

This chapter deals with the design and applications of the radial basis function (RBF) model. It is organized into three parts. The first part, consisting of Sect. 35.1, describes the two data mining activities addressed here: classification and regression. Next, we discuss the important issue of bias–variance tradeoff and its relationship to model complexity. The second part consists of Sects. 35.2 to 35.4. Section 35.2 describes the RBF model architecture and its parameters. In Sect. 35.3.1 we briefly describe the four common algorithms used for its design: clustering, orthogonal least squares, regularization, and gradient descent. In Sect. 35.3.2 we discuss an algebraic algorithm, the SG algorithm, which provides a step-by-step approach to RBF design. Section 35.4 presents a detailed example to illustrate the use of the SG algorithm on a small data set. The third part consists of Sects. 35.5 and 35.6. In Sect. 35.5 we describe the development of RBF classifiers for a well-known benchmark problem to determine whether Pima Indians have diabetes. We describe the need for and importance of partitioning the data into training, validation, and test sets. The training set is employed to develop candidate models, the validation set is used to select a model, and the generalization performance of the selected model is assessed using the test set. Section 35.6 describes a recent data mining application in bioinformatics, where the objective is to analyze the gene expression profiles of Leukemia data from patients whose classes are known to predict the target cancer class. Finally, Sect. 35.7 provides concluding remarks and directs the reader to related literature. Although the material in this chapter is applicable to other types of basis funktions, we have used only the Gaussian function for illustrations and case studies because of its popularity and good mathematical properties.

Neural networks have been used extensively to model unknown functional relationships between input and output data. The radial basis function RBF model is a special type of neural network consisting of three layers: input, hidden, and output. It represents two sequential mappings. The first nonlinearly maps the input data via basis functions in the hidden layer. The second, a weighted mapping of the basis function outputs, generates the model output. The two mappings are usually treated separately, which makes RBF a very versatile modeling technique. There has been some debate about whether RBF is biologically plausible, and hence whether it really is a neural network model. Nevertheless, it has become an established model for diverse classification and regression problems. For example, it has been successfully employed in areas such as data mining, medical diagnosis, face and speech recognition, robotics, forecasting stock prices, cataloging objects in the sky, and bioinformatics.

RBF networks have their theoretical roots in regularization theory and were originally developed by Russian mathematicians in the 1960s. They were used for strict interpolation among multidimensional data [35.1], where it is required that every input be mapped to a corresponding output. *Broomhead* and *Lowe* [35.2] used the RBF model for approximation. The relation-

ship between the use of RBF for strict interpolation and approximation is of special interest in this chapter. Further, they have been shown to possess very important mathematical properties of universal and best approximation [35.3]. A function approximation scheme is said to have the property of universal approximation if the set of functions supported by the approximation scheme is dense in the space of the continuous functions defined on the input domain, and it has the property of best approximation if there is one function among this set that has the lowest approximating error for any given function to be approximated. This provides a strong mathematical justification for their practical application, since the popular multilayer perceptrons approach, for example, does not possess the property of best approximation.

In this chapter we describe the radial basis network architecture, its design and applications. The data mining problem we address is described in Sect. 35.1. The RBF model and its parameters are described in Sect. 35.2. Sect. 35.3 presents some common design algorithms. An important design problem relates to determining the number of basis functions in the hidden layer and their parameters. A new algebraic algorithm, the SG algorithm, provides a systematic methodology for doing so and is also discussed in Sect. 35.3. An illustrative modeling example is described in Sect. 35.4, and a benchmark case study about diabetes classification is presented in Sect. 35.5. An important data mining application for cancer class prediction based on microarray data analysis is described in Sect. 35.6. Finally, some concluding remarks are presented in Sect. 35.7.

35.1 Problem Statement

Knowledge discovery applications are aimed at extracting accurate, previously unknown, useful, and actionable information from databases, and the related discipline is known as knowledge discovery in databases KDD. The usual steps in this process, often followed iteratively, are: selection of the target data from the raw databases; its processing and transformation; information extraction (called "data mining") using algorithmic approaches; interpretation of the information gained; and its useful application. Data mining is the key phase in this process and is main interest in this chapter. For a detailed description of this discipline, see [35.4–6].

The data available is a collection of records, each record itself being a collection of fields or data items. This tabular data is the input to a data mining algorithm, the output of which is the desired information or the knowledge sought. Usual data mining applications include data characterization, pattern recognition, rule extraction, clustering, trend analysis, and visualization. In this chapter, we address only pattern recognition. The pattern recognition task can be described as the construction of a model for input-output mapping on the basis of available tabular data. Such data are called the training sample. The inputs are d-dimensional independent variables or features ($\boldsymbol{x}$'s), and the output is a one-dimensional dependent variable (y). The two common pattern recognition tasks are classification and regression. In the case of classification, y represents one of a possible L classes. Most applications, however, are binary classification problems; in other words $L = 2$ in most practical cases. In regression problems y is a continuous variable. The constructed model is employed to predict the output y for future observed input $\boldsymbol{x}$'s. The objective is to seek a data mining algorithm that predicts y as accurately as possible. In other words, we seek to minimize the prediction error on future data.

In classification problems, a commonly used prediction error measure is the "classification error" (CE), which is defined as the ratio of misclassified objects to the total number of objects. For regression problems, the mean squared error (MSE) is generally employed. It is the averaged sum of squared discrepancies between the actual and the predicted values. The model performance is computed for the training data. An independent data set that is representative of the data used for training and is called the "validation set" is employed to compare the performance of the derived models. Then, yet another independent data set, called the "test set", is employed to assess the test error of the selected model as a performance measure on future data, for which the model was developed in the first place.

In the design and selection of RBF models, we prefer a parsimonious model; in other words, one with the smallest number of terms that provides the desired performance. However, it is well known that a model with too few terms can suffer from underfitting, while one with too many can result in overfitting and will therefore fail to generalize well on future data. This problem

is also known as the "bias-variance dilemma" in machine learning and statistics literature [35.5,7,8]. Simple models tend to have high bias and low variance, while complex models tend to have low bias and high variance. A graphical illustration of this phenomenon is shown in Fig. 35.1. The objective of modeling is to seek a compromise or tradeoff between bias and variance, and to find a model of just the right complexity. For the RBF model, as we discuss below, this means that we seek a model with just enough basis functions in the hidden layer. In other words, we seek a model that has a low training error and a low generalization error as assessed by the error on test data. In Fig. 35.1 this idealized situation is labeled as the "best model".

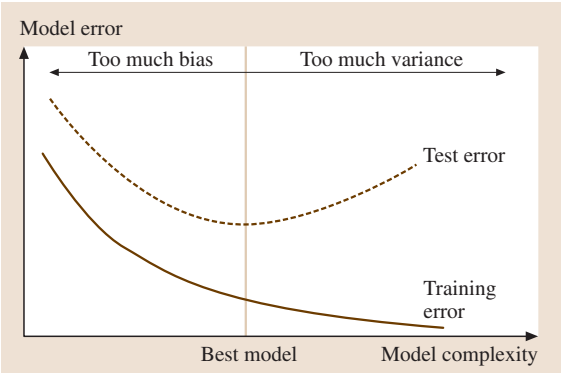

Fig. 35.1 Typical behavior of training error and test error

35.2 RBF Model and Parameters

A typical RBF network is shown in Fig. 35.2. It has three layers: input, hidden and output. The input layer consists of an $n \times d$ input data matrix $\mathbf{X}$:

$$\mathbf{X} = (\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_n)^T \in \mathbb{R}^{n \times d} , \tag{35.1}$$

where $\boldsymbol{x}_i$, $i = 1, 2, \ldots, n$ are the d-dimensional vectors and n is the size of the data. The hidden layer has m basis functions $\phi_1(\cdot)$, $\phi_2(\cdot)$, $\ldots$, $\phi_m(\cdot)$ centered at basis function centers $\boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \ldots, \boldsymbol{\mu}_m$, respectively, and connected to the output layer with weights $w_1, w_2, \ldots, w_m$, respectively. The basis functions transform the input data matrix via nonlinear mappings based on using the Euclidean distance between the input vector $\boldsymbol{x}$ and the prototype vectors $\boldsymbol{\mu}_j, j = 1, \ldots,$ m. This mapping can be represented as follows, where $\|\cdot\|$ is the Euclidean norm:

$$\phi_j(\boldsymbol{x}) = \phi\left(\|\boldsymbol{x} - \boldsymbol{\mu}_j\|\right), \; j = 1, 2, ..., m. \tag{35.2}$$

The $n \times d$ input matrix is thus transformed by the m basis functions into the following $n \times m$ design matrix $\boldsymbol{\Phi}$. In this matrix, the jth column represents the outputs from the jth basis function, $j = 1, 2, \ldots,$m.

$$\boldsymbol{\Phi} = \begin{array}{c} \begin{matrix} \Phi_1 & \hbar & \Phi_j & \hbar & \Phi_m \end{matrix} \\ \begin{pmatrix} \phi_1(x_1) & \hbar & \phi_j(x_1) & \hbar & \phi_m(x_1) \\ \phi_1(x_2) & \hbar & \phi_j(x_2) & \hbar & \phi_m(x_2) \\ \hbar & \hbar & \hbar & \hbar & \hbar \\ \phi_1(x_n) & \hbar & \phi_j(x_n) & \hbar & \phi_m(x_n) \end{pmatrix} \end{array} . \tag{35.3}$$

Several types of basis function have been considered in the literature. The common ones are Gaussian, thin plate spline, inverse multiquadratic, and cubic [35.7,9]. However, the basis function most commonly used for most applications is the Gaussian. Its form is $\phi(r) = \exp\left(-\frac{r^2}{2\sigma^2}\right)$, where σ is a parameter that controls the smoothness properties of the approximating function. The expression for the jth Gaussian function mapping can be explicitly written as

$$\phi_j(\boldsymbol{x}) = \exp\left(-\frac{\|\boldsymbol{x} - \boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right) , \tag{35.4}$$

where $\boldsymbol{\mu}_j$ is the center and σ_j is the width of the jth basis function, $j = 1, 2, \ldots,$m. On substituting in (35.3), we

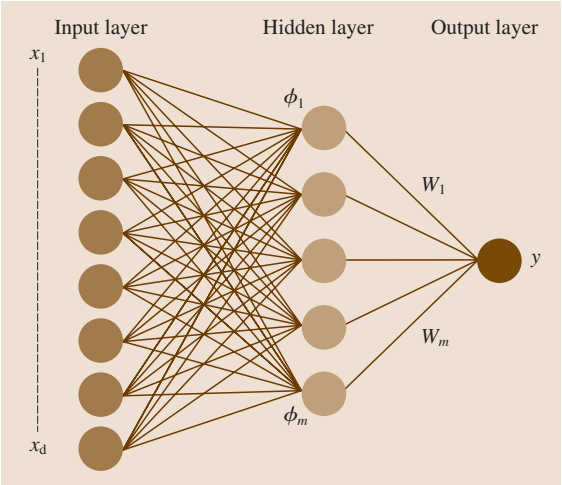

Fig. 35.2 Radial basis function network

can write the expression for a Gaussian design matrix, $\boldsymbol{\Phi}$, as

$$\boldsymbol{\Phi} = \begin{array}{c} \begin{array}{ccccc} \Phi_1 & \hbar & \Phi_j & \hbar & \Phi_m \end{array} \\ \begin{pmatrix} \exp\left(-\frac{\|\boldsymbol{x}_1-\boldsymbol{\mu}_1\|^2}{2\sigma_1^2}\right) & \hbar & \exp\left(-\frac{\|\boldsymbol{x}_1-\boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right) & \hbar & \exp\left(-\frac{\|\boldsymbol{x}_1-\boldsymbol{\mu}_m\|^2}{2\sigma_m^2}\right) \\ \exp\left(-\frac{\|\boldsymbol{x}_2-\boldsymbol{\mu}_1\|^2}{2\sigma_1^2}\right) & \hbar & \exp\left(-\frac{\|\boldsymbol{x}_2-\boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right) & \hbar & \exp\left(-\frac{\|\boldsymbol{x}_2-\boldsymbol{\mu}_m\|^2}{2\sigma_m^2}\right) \\ \hbar & \hbar & \hbar & \hbar & \hbar \\ \exp\left(-\frac{\|\boldsymbol{x}_n-\boldsymbol{\mu}_1\|^2}{2\sigma_1^2}\right) & \hbar & \exp\left(-\frac{\|\boldsymbol{x}_n-\boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right) & \hbar & \exp\left(-\frac{\|\boldsymbol{x}_n-\boldsymbol{\mu}_m\|^2}{2\sigma_m^2}\right) \end{pmatrix} \end{array} \tag{35.5}$$

For the special case where the number of basis functions equals the number of data vectors (when $m = n$), and if we use the n input data vectors as the basis function centers, the matrix in (35.5) is called the interpolation matrix. It is this matrix that is employed for the strict interpolation problem mentioned earlier and is of special interest in the SG algorithm. To compute the output, the entries in the matrix given by (35.5) are combined linearly according to the weights. The resulting values at the output node are then given by

$$f(\boldsymbol{x}) = \sum_{j=1}^{m} w_j \exp\left(-\frac{\|\boldsymbol{x}-\boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right), \tag{35.6}$$

where $f(\boldsymbol{x})$ represents the Gaussian RBF output for the input vector $\boldsymbol{x}$. Thus, for n input vectors $(\boldsymbol{x}_1, \boldsymbol{x}_2, \ldots, \boldsymbol{x}_n)^{\mathrm{T}}$, the output layer consists of n outputs, one for each $\boldsymbol{x}$, as indicated below,

$$\left[f(\boldsymbol{x}_1),\, f(\boldsymbol{x}_2),\, ...,\, f(\boldsymbol{x}_n)\right]^{\mathrm{T}} = \boldsymbol{\Phi}\boldsymbol{w}\,. \tag{35.7}$$

Thus, we see that a Gaussian RBF model is fully defined by the number of basis functions (m), their centers $[\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_m)]$, widths $[\boldsymbol{\sigma} = (\sigma_1, \sigma_2, \ldots, \sigma_m)]$, and the weights $[\boldsymbol{w} = (w_1, w_2, \ldots, w_m)]$ to the output layer. In most applications, and in this chapter, a global width σ is used for each basis function. The parameters m, $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ define the hidden layer (the nonlinearity of the RBF model). The weights ($\boldsymbol{w}$) define the linear part as indicated in Fig. 35.2. This completes discussion of the radial basis function model and its parameters. We now move on to discuss RBF model development; in other words, the determination of its parameters from the training sample or the given input-output data set.

35.3 Design Algorithms

A common characteristic of most design or training algorithms used for RBF models is that they employ a two-stage training procedure. In the first stage, only the input data is used to determine the basis function parameters. For the Gaussian case, these are the number of basis functions, their centers, and their widths. Once the basis function parameters are determined, the weights are found in the second stage to minimize some error measure. There are a large number of procedures available in the literature for RBF design. We first describe the four commonly used ones and then a relatively new algorithm called the SG algorithm.

35.3.1 Common Algorithms

Clustering: [35.10] A set of centers for basis functions can be obtained by employing clustering techniques on the input data. The k-means clustering algorithm [35.5] is used to locate a set of k basis function centers. For a specified k, the algorithm seeks to partition the input data into k disjoint subsets, each of which corresponds to a cluster. Once the cluster membership is determined, the averages of the data points in these clusters are chosen as the centers of the k basis functions in the RBF model, and m is taken to equal k. Next, the widths of the basis functions are determined by a P-nearest neighbor heuristic [35.5]. Thus, if P=1, the width of the jth basis function is set to be the Euclidean distance between its own center and the center of its nearest neighbor.

Orthogonal Least Square: [35.11] In this procedure a set of vectors is constructed in the space spanned by the vectors of hidden unit outputs for the training set and then by directly finding the center of an additional basis function such that it gives the greatest reduction in residual sum-of-square error. The stopping criterion employed is a threshold on the fraction of the variance explained by the model.

Regularization: [35.7] These procedures are motivated by the theory of regularization. A regularization parameter is used to control the smoothness properties of a mapping function by adding an extra term to the minimized error function that is designed to penalize mappings that are not smooth.

Gradient Descent: [35.10] Such training algorithms are fully supervised gradient-descent methods over some error measure. Specifically, the model parameters are updated as a function of this error measure according to some specified learning rates associated with the RBF parameters.

35.3.2 SG Algorithm

In the SG algorithm, the parameters $(m, \boldsymbol{\sigma}, \boldsymbol{\mu})$ of the nonlinear mapping are first determined from the input matrix $\mathbf{X}$ without referencing the output values. Then, the linear parameters $(\boldsymbol{w})$ are determined by referencing the output y's. The SG algorithm consists of four steps, given below. These steps are shown schematically in Fig. 35.3.

- Step 1: Select a range of values for global width, $\boldsymbol{\sigma}$, and a representation capability measure, δ, according to the heuristics given below.
- Step 2: Determine a value of m that satisfies the δ criterion. This step involves singular value decomposition of the interpolation matrix computed from the input data matrix $\mathbf{X}$ for a chosen $\boldsymbol{X}\sigma$.
- Step 3: Determine centers for the m basis functions that maximize structural stability provided by the selected model complexity, m. This step involves the use of QR factorization.
- Step 4: Compute weights using the pseudoinverse and estimate the output values.

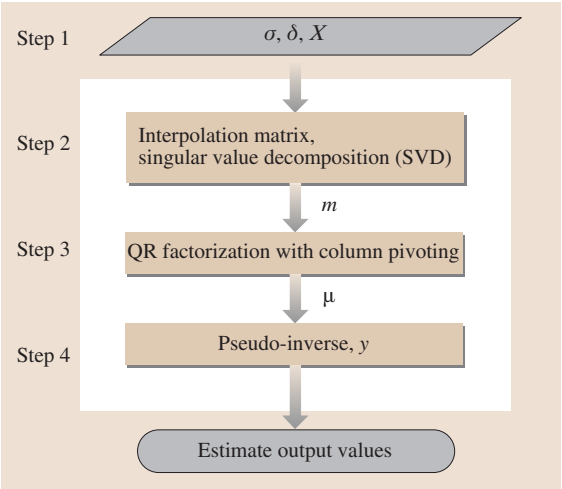

Fig. 35.3 The four steps of the SG RBF modeling algorithm

Note that the choice of parameters in Step 1 affects the quality of the developed model. Heuristically, we take $\boldsymbol{\sigma}$ to be approximately in the range of 0 to $0.75\sqrt{d/2}$, and δ is taken to be in the range 0.1% to 1.0% [35.12], where d is the dimensionality of the input data points. RBF models are then developed for a few judicially chosen values of these parameters. The performance of these models is assessed using some prespecified approach, and the most appropriate model is selected. This process is illustrated in Sects. 35.5 and 35.6 for two real-world data sets.

35.4 Illustrative Example

In this section we illustrate the use of the SG algorithm on a small data set generated from the following sine function [35.7]:

$$h(x) = 0.5 + 0.4\sin(2\pi x)\,.$$

Five values of the above function are computed at equal intervals of x in the range 0.0 to 1.0. Then random noise values, generated from a Gaussian distribution with mean zero and variance 0.25, are added to $h(x)$ to obtain five data points. The data set is listed in Table 35.1 along with the true $h(x)$ values. Our objective is to seek a good approximation for the unknown function $h(x)$, based only on the x and observed y data. A plot of the true $h(x)$ and the observed y values is shown in Fig. 35.4. Also shown is an approximated or estimated function found using interpolation, as discussed next.

Table 35.1 Dataset for illustrative example

i	x_i	$h(x_i)$	y_i
1	0.00	0.50	0.5582
2	0.25	0.90	0.9313
3	0.50	0.50	0.5038
4	0.75	0.10	0.1176
5	1.00	0.50	0.4632

First, we consider the strict interpolation problem for this data set, then we illustrate the use of the SG algorithm for approximation. The interpolation problem is to determine a Gaussian RBF that gives exact outputs for each x; in other words, we seek a model whose output is exactly equal to the y value corresponding to that x given in Table 35.1. For this we construct an inter-

polation matrix with five basis functions, one centered at each input value x. Suppose we use a global width $\sigma = 0.4$. Then the five Gaussian basis functions, each with $\sigma = 0.4$, will be centered at the five x values of Table 35.1 and will map the input data into an interpolation matrix according to the expressions in (35.5) with $m = n = 5$. For example, the column Φ_2 in matrix Φ is obtained according to (35.3) and (35.4) by substituting $\mu = 0.25$ and the five x values given below.

$$\Phi_2(x) = \begin{pmatrix} \phi_2(x_1) \\ \phi_2(x_2) \\ \phi_2(x_3) \\ \phi_2(x_4) \\ \phi_2(x_5) \end{pmatrix} = \begin{pmatrix} \exp\left(-\frac{\|0.00-0.25\|^2}{2(0.4)^2}\right) \\ \exp\left(-\frac{\|0.25-0.25\|^2}{2(0.4)^2}\right) \\ \exp\left(-\frac{\|0.50-0.25\|^2}{2(0.4)^2}\right) \\ \exp\left(-\frac{\|0.75-0.25\|^2}{2(0.4)^2}\right) \\ \exp\left(-\frac{\|1.00-0.25\|^2}{2(0.4)^2}\right) \end{pmatrix}$$

$$= \begin{pmatrix} 0.8226 \\ 1.0000 \\ 0.8226 \\ 0.4578 \\ 0.1724 \end{pmatrix} .$$

Other columns of the matrix are similarly computed for different μ's. The final 5×5 interpolation matrix is given below.

$$\Phi = \begin{array}{ccccc} \Phi_1 & \Phi_2 & \Phi_3 & \Phi_4 & \Phi_5 \\ 1.0000 & 0.8226 & 0.4578 & 0.1724 & 0.0439 \\ 0.8226 & 1.0000 & 0.8226 & 0.4578 & 0.1724 \\ 0.4578 & 0.8226 & 1.0000 & 0.8226 & 0.4578 \\ 0.1724 & 0.4578 & 0.8226 & 1.0000 & 0.8226 \\ 0.0439 & 0.1724 & 0.4578 & 0.8226 & 1.0000 \end{array} \quad (35.8)$$

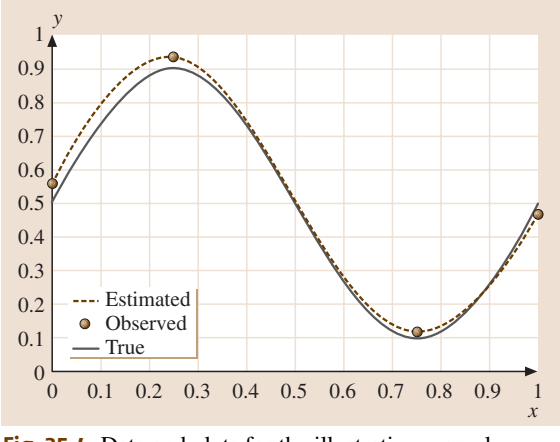

Fig. 35.4 Data and plots for the illustrative example

This matrix is symmetric because we have chosen a global width for the basis functions. Also, its diagonal values are 1.0 because the x values are listed in increasing order and because the height of the basis function at its center is always 1.0. The above interpolation matrix and the y_i vector in Table 35.1 are used to compute weights $\boldsymbol{w} = (w_1, w_2, \ldots, w_5)^T$ of the links to the output node using the pseudoinverse. Finally, the interpolated function is obtained from (35.6) and (35.7) as

$$\begin{aligned} \hat{y} = & -2.09 \exp\left(-\frac{\|x-0.0\|^2}{2(0.4)^2}\right) \\ & +3.87 \exp\left(-\frac{\|x-0.25\|^2}{2(0.4)^2}\right) \\ & -0.06 \exp\left(-\frac{\|x-0.50\|^2}{2(0.4)^2}\right) \\ & +3.63 \exp\left(-\frac{\|x-0.75\|^2}{2(0.4)^2}\right) \\ & +2.91 \exp\left(-\frac{\|x-1.0\|^2}{2(0.4)^2}\right) . \end{aligned}$$

Note that here $\hat{y}$ is the weighted sum of the basis function outputs and consists of five terms, one corresponding to each basis function. A plot of this function is shown in Fig. 35.4, where the estimated values are exactly equal to the observed y's since we are dealing with exact interpolation.

In practical problems, exact interpolation is undesirable because it represents extreme overfitting. Referring to Fig. 35.1, this indicates a very complex model that will not perform well on new data in the sense that it will exhibit a high generalization error. In practice, we seek an approximate model according to the guidelines discussed in Sect. 35.1. To achieve this goal we use the SG algorithm, where the user controls the tradeoff between underfitting and overfitting or between bias and variance by specifying the values of δ [35.12]. As indicated above, for practical applications, $\delta = 0.1\%$ to 1% seems to be a good set of values to consider. The RBF model is then designed according to the above four-step procedure. We provide a description of this procedure for the sine data below. However, details of the singular value decomposition and QR factorization are beyond the scope of this chapter and can be found in [35.12, 13].

The starting point in the SG algorithm is the selection of σ and δ. For the sine data, suppose

$\sigma = 0.4$ and $\delta = 0.01$. The 5×5 interpolation matrix for this σ is computed as shown above. In step 2, its singular value decomposition yields five singular values. Using these and $\delta = 0.01$, for this example data, we obtain $m = 4$. Then, in step 3, QR factorization identifies the four centers for the basis functions as being $\mu_1 = 1.00$, $\mu_2 = 0.00$, $\mu_3 = 0.75$ and $\mu_4 = 0.25$. Finally, the weights are obtained in step 4 as $w_1 = -2.08$, $w_2 = 3.82$, $w_3 = -3.68$ and $w_4 = 2.92$. Thus, an approximation of the unknown function $h(x)$ based on the available data x and y in Table 35.1 is provided by an RBF model

$$\hat{y} = \sum_{j=1}^{4} w_j \exp\left(-\frac{\|x - \mu_j\|^2}{2\sigma_j^2}\right),$$

where w_j and μ_j are as listed above and $\sigma_j = 0.4$ for $j = 1, 2, 3, 4$. This estimate of y is based on the four basis functions selected by the SG algorithm for $\delta = 0.01$. Note that in this simple example we obtained a value of m that is almost the same as n. However, in practical applications with real-world data, the design value of m is generally much smaller than the n as seen in Sects. 35.5 and 35.6.

35.5 Diabetes Disease Classification

This benchmark problem is taken from the Proben1 data set of the UCI repository [35.14]. It was studied in Lim [35.15] using the SG algorithm. The objective is to develop a classification model to determine whether diabetes of Pima Indians is positive or negative, based on personal data such as age, number of times pregnant, and so on. Other factors considered include the results from medical examinations, such as data on blood pressure, body mass index, and results of glucose tolerance tests. There are eight inputs, two outputs, 768 examples, and no missing values in this data set. A summary of the input and output attributes and the encoding scheme employed for data processing is given in Table 35.2. Here, all inputs are continuous and each is normalized to a range of 0 to 1 for data preprocessing. Attribute number 9 is the output, consisting of two values, diabetes or no diabetes. Before proceeding with RBF model development we need to decide upon an approach to model evaluation. This point is discussed next.

The generalization performance of an RBF model relates to its predictive ability on some future data. Therefore, we need to be able to assess this performance during the model building process. For applications where we have adequate data, the best approach is to randomly divide the available data into three sets: training set, validation set, and test set [35.5]. We use the training set for model development, the validation set to compare the developed models and, usually, select the model with the smallest error on the validation set. The test set is not used until after the final model is selected. The performance of the selected model on the test set is used as a measure of its generalization performance. A common practice is to split the data into 50% for training and 25% each for validation and test sets. Using this

Table 35.2 Data description for the diabetes example

Inputs (8)			
Attribute No.	**No. of attributes**	**Attribute meaning**	**Values and encoding**
1	1	Number of times pregnant	0 ... 17 → 0 ... 1
2	1	Plasma glucose concentration after 2 h in an oral glucose tolerance test	0 ... 199 → 0 ... 1
3	1	Diastolic blood pressure (mm Hg)	0 ... 122 → 0 ... 1
4	1	Triceps skin fold thickness (mm)	0 ... 99 → 0 ... 1
5	1	2-hour serum insulin (mu U/ml)	0 ... 846 → 0 ... 1
6	1	Body mass index (weight in kg/(height in m)^2)	0 ... 67.1 → 0 ... 1
7	1	Diabetes pedigree function	0.078 ... 2.42 → 0 ... 1
8	1	Age (years)	21 ... 81 → 0 ... 1
Output (2)			
9	2	No diabetes Diabetes	1 0 0 1

Table 35.3 RBF models for the diabetes example

$\delta = 0.01$ Model	m	σ	Classification error (CE) (%)		
			Training	Validation	Test
A	18	0.6	20.32	23.44	24.48
B	9	0.7	21.88	21.88	22.92
C	9	0.8	22.66	21.35	23.44
D	8	0.9	22.92	21.88	25.52
E	8	1.0	23.44	21.88	25.52
F	7	1.1	26.04	30.21	30.21
G	6	1.2	25.78	28.13	28.13
H	5	1.3	25.26	31.25	30.73

split for this application, we divide the set of 768 patients into 384 for training, 192 for validation and 192 for the test set.

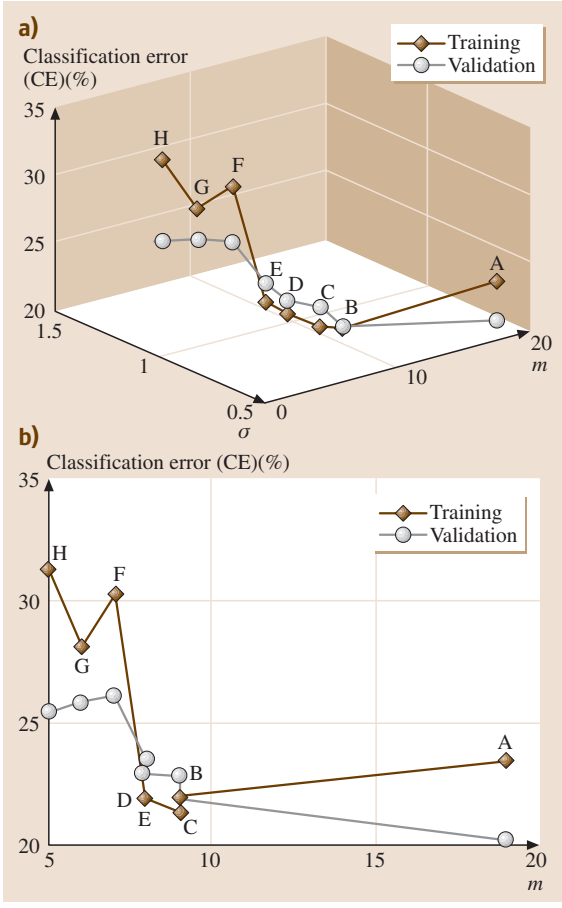

Fig. 35.5 Plots of training and validation errors for the diabetes example ($\delta = 0.01$)

We employ the heuristics given in Sect. 35.3 and select values of width (σ) in the range 0.6 to 1.3. Also, we use $\delta = 0.01$, 0.005 and 0.001. The algorithm is then executed to develop RBF models. For each model, the training and validation classification errors(CE) are also computed. However, to provide a better insight, the test error is also computed here. The classification results for eight models (A to H) for $\delta = 0.01$ are shown in Table 35.3. We note that as σ decreases from 1.3 to 0.6, the m value increases, and the training CE decreases from 25.26% to 20.32%. However, the validation CE first decreases from 31.25% to 21.35% and then increases to 23.44%. The test error first decreases with increasing m and then begins to increase. The validation errors, though used for different purposes, tend to exhibit similar behavior with respect to m. The pattern of CE error behavior is shown graphically in Fig. 35.5. Here, the errors are shown with respect to m and σ in Fig. 35.5a and as a function of m alone in Fig. 35.5b. We note that the training and validation error behavior is quite similar to the theoretical pattern discussed above and that depicted in Fig. 35.1. However, for some models the validation error is smaller than the training error. Also, the error behavior is not monotonic. This can and does happen in practical applications due to random variations in the real-world data assigned to the training, validation, and test sets.

To select the model, we evaluate the validation CE for models A to H. The minimum value occurs for model C, and hence this model is selected as the preferred model. The test CE for this model is 23.44. Next, models were developed for $\delta = 0.005$ and 0.001. However, the details of these models are not shown here. The best models for these two cases and for $\delta = 0.01$ are listed in Table 35.4.

To select the final model, we consult the results in Table 35.4 and note that the RBF model with the smallest

Table 35.4 Selected models and error values for the diabetes example

Classification error (CE) (%)					
δ	m	σ	Training	Validation	Test
0.01	9	0.8	22.66	21.35	23.44
0.005	9	0.8	22.66	21.35	23.44
0.001	10	1.2	22.66	20.83	23.96

validation error is the model with $m = 10, \boldsymbol{\sigma} = 1.2$. This is our final choice for modeling the classification of diabetes. Its test error is 23.96. What this says is that when this model is used to evaluate future patients for diabetes or no diabetes, the model will misclassify, on average, about 24% of the patients. Also, note that the design value of m, the number of basis functions, is only 10, while $n = 384$ for the training data. Thus, we see that a much simpler model than strict interpolation requires provides a good classifier. If we were to use $m = n = 384$ in this case, all patients in the training set would be correctly classified with $CE = 0.0$. However, the n performance of such a model on the validation and test sets is likely to be very poor.

35.6 Analysis of Gene Expression Data

Now we describe a data mining application of the RBF model to binary cancer classification based on gene expression data from DNA microarray hybridization experiments [35.16]. Cancer class prediction is crucial to its treatment, and developing an automated analytical approach for classification based upon the microarray expression is an important task [35.16]. A generic approach to classifying two types of acute leukemia based on the monitoring of gene expression by DNA microarrays was originally pioneered by [35.17]. We employ their data set to illustrate the classifier development process and use sensitivity analyses in order to select an appropriate cancer classification model. The goal is to construct a classifier that distinguishes between two cancer classes based on gene expression data from patients whose class, AML (acute myeloid leukemia) or ALL (acute lymphoblastic leukemia), is known. This classifier will be used to predict the cancer class of a future patient about whom only the gene expression will be known, not the class.

The dataset consists of 72 samples. The number of gene expression levels for each patient in this dataset is 7129. In other words, 7129 attributes or features are used to describe a patient in this dataset. Since the set is relatively small, it is split into 38 training samples and 34 test samples, where each sample represents a patient [35.17]. The test set error is used for model selection here.

The classification results for the training data set, using the SG algorithm of Sect. 35.3, are summarized in Table 35.5 for $\delta = 0.01$. Results for test data are also included. Here we have seven models (A to G) for $\boldsymbol{\sigma}$ ranging from 32 to 20, and the design values of m vary from 12 to 38.

The CE for the training set varies from 0% to 5.26% and for the test set from 14.71% to 26.47%. Next, the behavior of the training and test error in the $(m–\boldsymbol{\sigma})$ plane is shown in Fig. 35.6. Comparing it to Fig. 35.1, we note that the training error decreases as m increases, becoming 0% at $m = n = 38$. This happens when the classification model represents exact interpolation.

Table 35.5 Classification results for the cancer gene example

Model	σ	m	Classification error (%)	
			Training	Test
A	32	12	5.26	26.47
B	30	15	2.63	20.59
C	28	21	2.63	17.65
D	26	29	0	14.71
E	24	34	0	14.71
F	22	38	0	17.65
G	20	38	0	17.65

Next, we discuss model selection based on the test CE in Table 35.5. This error first decreases with increasing m and then increases, a pattern similar to the theoretical behavior depicted in Fig. 35.1. Based on the discussion in Sect. 35.2, the best model is D. Note that here we have a somewhat degenerate case, where the training error is zero and the test error is minimum for the selected model.

35.7 Concluding Remarks

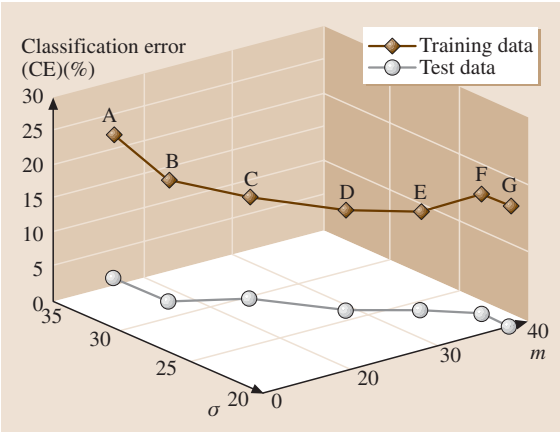

Fig. 35.6 Classification errors for the cancer gene example

In this chapter we introduced the RBF model and provided a detailed discussion of its design by evaluating the training, validation and test errors as surrogates for bias-variance phenomena. A simple example was used for illustration and then a benchmark data set was analyzed. Finally, the RBF model was used for a recent data mining application, cancer class prediction based on gene expression data. In our presentation we used only Gaussian basis functions because of their popularity and good mathematical properties. The methodology, however, is applicable to several other types of basis functions.

There is a vast body of literature on the topic of radial basis functions. The chapters in *Bishop* [35.7], *Haykin* [35.9], and *Kecman* [35.18] provide good coverage. *Buhmann* [35.19] is a rather theoretical book on this subject. A recent collection of methodologies and applications of the RBF model appears in (see *Howlett* and *Jain* [35.20]). Some other applications can be found in *Shin* and *Goel* [35.13, 21]. New developments in the theory and applications of radial basis functions can also be found in most journals and conference proceedings on neural networks and machine learning.

References

35.1 M. J. D. Powell: Radial basis functions for multivariable interpolation: A review. In: *Algorithms for Approximation*, ed. by J. C. Mason, M. G. Cox (Oxford Univ. Press, Oxford 1987) pp. 143–167

35.2 D. S. Broomhead, D. Lowe: Multivariable functional interpolation and adaptive networks, Comp. Sys. **2**, 321–355 (1988)

35.3 F. Girosi, T. Poggio: Networks and the best approximation property, Biol. Cybern. **63**, 169–176 (1990)

35.4 J. Han, M. Kamber: *Data Mining* (Morgan Kauffman, San Francisco 2001)

35.5 T. Hastie, R. Tibshirani, J. Friedman: *The Elements of Statistical Learning: Data Mining, Inference, and Prediction* (Springer, Berlin Heidelberg 2001)

35.6 N. Ye: *The Handbook of Data Mining* (Lawrence Erlbaum Associates, Mahwah, NJ 2003)

35.7 C. M. Bishop: *Neural Networks for Pattern Recognition* (Oxford Univ. Press, Oxford 1995)

35.8 J. Friedman: On bias, variance, 0-1 loss, and the curse of dimensionality, Data Min. Knowl. Disc. **1**, 55–77 (1997)

35.9 S. Haykin: *Neural Networks: A Comprehensive Foundation* (Prentice Hall, New York 1999)

35.10 J. Moody, C. J. Darken: Fast learning in networks of locally-tuned processing units, Neural Comp. **1**, 281–294 (1989)

35.11 S. C. Chen, C. F. N. Cowan, P. M. Grant: Orthogonal least squares learning algorithm for radial basis function networks, IEEE Trans. Neural Networks **2**(2), 302–309 (1991)

35.12 M. Shin: Design and Evaluation of Radial Basis Function Model for Function Approximation. Ph.D. Thesis (Syracuse Univ., Syracuse, N.Y. 1998)

35.13 M. Shin, A. L. Goel: Radial basis functions: An algebraic approach (with data mining applications), Tutorial Notes for the ECML/PKDD Conf. (ECML/PKDD, Pisa 2004)

35.14 L. Prechelt: Proben1–A Set of Neural Network Benchmark Problems and Benchmarking Rules, Interner Bericht, Universitat Karlsruhe, Fakultät für Informatik **21/94** (1994)
35.15 H. Lim: *An Empirical Study of RBF Models Using SG Algorithm* (Syracuse Univ., Syracuse, NY 2002)
35.16 S.M. Lim, K.E. Johnson (Eds.): *Methods of Microarray Data Analysis* (Kluwer, Dordrecht 2002)
35.17 T.R. Golub, D.K. Slonim, P. Tamayo, C. Huard, M. Gaasenbeek, J.P. Mesirov, H. Coller, M.L. Loh, J.R. Downing, M.A. Caligiuri, C.D. Bloomfield, E.S. Lander: Molecular classification of cancer: Class discovery and class prediction by gene expression monitoring, Science **286**, 531–537 (1999)
35.18 V. Kecman: *Learning and Soft Computing* (MIT Press, Cambridge 2000)
35.19 M. D. Buhmann: *Radial Basis Functions* (Cambridge Univ. Press, Cambridge 2003)
35.20 R. J. Howlett, L. C. Jain (Eds.): *Radial Basis Function Networks*, Vol. I,II (Physica, Heidelberg 2001)
35.21 M. Shin, A. L. Goel: Empirical data modeling in software engineering using radial basis functions, IEEE Trans. Software Eng. **6:26**, 567–576 (2002)