5

RADIAL BASIS FUNCTIONS

The network models discussed in Chapters 3 and 4 are based on units which compute a non-linear function of the scalar product of the input vector and a weight vector. Here we consider the other major class of neural network model, in which the activation of a hidden unit is determined by the *distance* between the input vector and a prototype vector.

An interesting and important property of these radial basis function networks is that they form a unifying link between a number of disparate concepts as we shall demonstrate in this chapter. In particular, we shall motivate the use of radial basis functions from the point of view of function approximation, regularization, noisy interpolation, density estimation, optimal classification theory, and potential functions.

One consequence of this unifying viewpoint is that it motivates procedures for training radial basis function networks which can be substantially faster than the methods used to train multi-layer perceptron networks. This follows from the interpretation which can be given to the internal representations formed by the hidden units, and leads to a two-stage training procedure. In the first stage, the parameters governing the basis functions (corresponding to hidden units) are determined using relatively fast, unsupervised methods (i.e. methods which use only the input data and not the target data). The second stage of training then involves the determination of the final-layer weights, which requires the solution of a linear problem, and which is therefore also fast.

5.1 Exact interpolation

Radial basis function methods have their origins in techniques for performing exact interpolation of a set of data points in a multi-dimensional space (Powell, 1987). The exact interpolation problem requires every input vector to be mapped exactly onto the corresponding target vector, and forms a convenient starting point for our discussion of radial basis function networks.

Consider a mapping from a d-dimensional input space $\mathbf{x}$ to a one-dimensional target space t. The data set consists of N input vectors $\mathbf{x}^n$, together with corresponding targets t^n. The goal is to find a function $h(\mathbf{x})$ such that

$$h(\mathbf{x}^n) = t^n, \qquad n = 1, \ldots, N. \tag{5.1}$$

The radial basis function approach (Powell, 1987) introduces a set of N *basis functions*, one for each data point, which take the form $\phi(\|\mathbf{x} - \mathbf{x}^n\|)$ where $\phi(\cdot)$ is some non-linear function whose form will be discussed shortly. Thus the nth such function depends on the distance $\|\mathbf{x} - \mathbf{x}^n\|$, usually taken to be Euclidean, between $\mathbf{x}$ and $\mathbf{x}^n$. The output of the mapping is then taken to be a linear combination of the basis functions

$$h(\mathbf{x}) = \sum_n w_n \phi(\|\mathbf{x} - \mathbf{x}^n\|). \tag{5.2}$$

We recognize this as having the same form as the generalized linear discriminant function considered in Section 3.3. The interpolation conditions (5.1) can then be written in matrix form as

$$\mathbf{\Phi w} = \mathbf{t} \tag{5.3}$$

where $\mathbf{t} \equiv (t^n)$, $\mathbf{w} \equiv (w_n)$, and the square matrix $\mathbf{\Phi}$ has elements $\Phi_{nn'} = \phi(\|\mathbf{x}^n - \mathbf{x}^{n'}\|)$. Provided the inverse matrix $\mathbf{\Phi}^{-1}$ exists we can solve (5.3) to give

$$\mathbf{w} = \mathbf{\Phi}^{-1}\mathbf{t}. \tag{5.4}$$

It has been shown (Micchelli, 1986) that, for a large class of functions $\phi(\cdot)$, the matrix $\mathbf{\Phi}$ is indeed non-singular provided the data points are distinct. When the weights in (5.2) are set to the values given by (5.4), the function $h(\mathbf{x})$ represents a continuous differentiable surface which passes exactly through each data point.

Both theoretical and empirical studies (Powell, 1987) show that, in the context of the exact interpolation problem, many properties of the interpolating function are relatively insensitive to the precise form of the non-linear function $\phi(\cdot)$. Several forms of basis function have been considered, the most common being the Gaussian

$$\phi(x) = \exp\left(-\frac{x^2}{2\sigma^2}\right) \tag{5.5}$$

where σ is a parameter whose value controls the smoothness properties of the interpolating function. The Gaussian (5.5) is a *localized* basis function with the property that $\phi \to 0$ as $|x| \to \infty$. Another choice of basis function with the same property is the function

$$\phi(x) = \left(x^2 + \sigma^2\right)^{-\alpha}, \qquad \alpha > 0. \tag{5.6}$$

It is not, however, necessary for the functions to be localized, and other possible choices are the thin-plate spline function

$$\phi(x) = x^2 \ln(x), \tag{5.7}$$

the function

$$\phi(x) = \left(x^2 + \sigma^2\right)^\beta, \qquad 0 < \beta < 1, \tag{5.8}$$

which for $\beta = 1/2$ is known as the multi-quadric function, the cubic

$$\phi(x) = x^3, \tag{5.9}$$

and the 'linear' function

$$\phi(x) = x \tag{5.10}$$

which all have the property that $\phi \to \infty$ as $x \to \infty$. Note that (5.10) linear in $x = \|\mathbf{x} - \mathbf{x}^n\|$ and so is still a non-linear function of the components of $\mathbf{x}$. In one dimension, it leads to a piecewise-linear interpolating function which represents the simplest form of exact interpolation. As we shall see, in the context of neural network mappings there are reasons for considering localized basis functions. We shall focus most of our attention on Gaussian basis functions since, as well as being localized, they have a number of useful analytical properties. The technique of radial basis functions for exact interpolation is illustrated in Figure 5.1 for a simple one-input, one-output mapping.

The generalization to several output variables is straightforward. Each input vector $\mathbf{x}^n$ must be mapped exactly onto an output vector $\mathbf{t}^n$ having components t_k^n so that (5.1) becomes

$$h_k(\mathbf{x}^n) = t_k^n, \qquad n = 1, \ldots, N \tag{5.11}$$

where the $h_k(\mathbf{x})$ are obtained by linear superposition of the same N basis functions as used for the single-output case

$$h_k(\mathbf{x}) = \sum_n w_{kn} \phi(\|\mathbf{x} - \mathbf{x}^n\|). \tag{5.12}$$

The weight parameters are obtained by analogy with (5.4) in the form

$$w_{kn} = \sum_{n'} (\Phi^{-1})_{nn'} t_k^{n'}. \tag{5.13}$$

Note that in (5.13) the same matrix Φ^{-1} is used for each of the output functions.

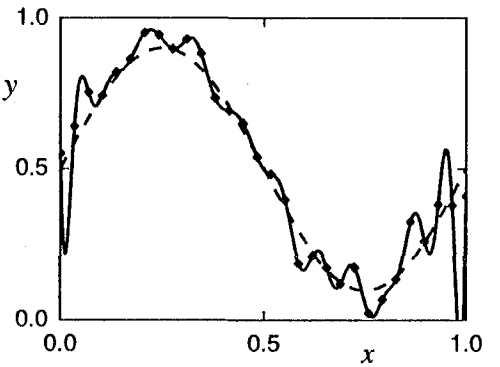

Figure 5.1. A simple example of exact interpolation using radial basis functions. A set of 30 data points was generated by sampling the function $y = 0.5 + 0.4\sin(2\pi x)$, shown by the dashed curve, and adding Gaussian noise with standard deviation 0.05. The solid curve shows the interpolating function which results from using Gaussian basis functions of the form (5.5) with width parameter $\sigma = 0.067$ which corresponds to roughly twice the spacing of the data points. Values for the second-layer weights were found using matrix inversion techniques as discussed in the text.

5.2 Radial basis function networks

The radial basis function mappings discussed so far provide an interpolating function which passes exactly through every data point. As the example in Figure 5.1 illustrates, the exact interpolating function for noisy data is typically a highly oscillatory function. Such interpolating functions are generally undesirable. As discussed in Section 1.5.1, when there is noise present on the data, the interpolating function which gives the best generalization is one which is typically much smoother and which averages over the noise on the data. An additional limitation of the exact interpolation procedure discussed above is that the number of basis functions is equal to the number of patterns in the data set, and so for large data sets the mapping function can become very costly to evaluate.

By introducing a number of modifications to the exact interpolation procedure we obtain the radial basis function neural network model (Broomhead and Lowe, 1988; Moody and Darken, 1989). This provides a smooth interpolating function in which the number of basis functions is determined by the complexity of the mapping to be represented rather than by the size of the data set. The modifications which are required are as follows:

1. The number M of basis functions need not equal the number N of data points, and is typically much less than N.
2. The centres of the basis functions are no longer constrained to be given by

input data vectors. Instead, the determination of suitable centres becomes part of the training process.

3. Instead of having a common width parameter σ, each basis function is given its own width σ_j whose value is also determined during training.
4. Bias parameters are included in the linear sum. They compensate for the difference between the average value over the data set of the basis function activations and the corresponding average value of the targets, as discussed in Section 3.4.3.

When these changes are made to the exact interpolation formula (5.12), we arrive at the following form for the radial basis function neural network mapping

$$y_k(\mathbf{x}) = \sum_{j=1}^{M} w_{kj}\phi_j(\mathbf{x}) + w_{k0}. \tag{5.14}$$

If desired, the biases w_{k0} can be absorbed into the summation by including an extra basis function ϕ_0 whose activation is set to 1. For the case of Gaussian basis functions we have

$$\phi_j(\mathbf{x}) = \exp\left(-\frac{\|\mathbf{x} - \boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right) \tag{5.15}$$

where $\mathbf{x}$ is the d-dimensional input vector with elements x_i, and $\boldsymbol{\mu}_j$ is the vector determining the centre of basis function ϕ_j and has elements μ_{ji}. Note that the Gaussian basis functions in (5.15) are not normalized, as was the case for Gaussian density models in Chapter 2 for example, since any overall factors can be absorbed into the weights in (5.14) without loss of generality. This mapping function can be represented as a neural network diagram as shown in Figure 5.2. Note that more general topologies of radial basis function network (more than one hidden layer for instance) are not normally considered.

In discussing the representational properties of multi-layer perceptron networks in Section 4.3.1, we appealed to intuition to suggest that a linear superposition of localized functions, as in (5.14) and (5.15), is capable of universal approximation. Hartman *et al.* (1990) give a formal proof of this property for networks with Gaussian basis functions in which the widths of the Gaussians are treated as adjustable parameters. A more general result was obtained by Park and Sandberg (1991) who show that, with only mild restrictions on the form of the kernel functions, the universal approximation property still holds. Further generalizations of this results are given in (Park and Sandberg, 1993). As with the corresponding proofs for multi-layer perceptron networks, these are existence proofs which rely on the availability of an arbitrarily large number of hidden units, and they do not offer practical procedures for constructing the networks. Nevertheless, these theorems are crucial in providing a theoretical foundation on which practical applications can be based with confidence.

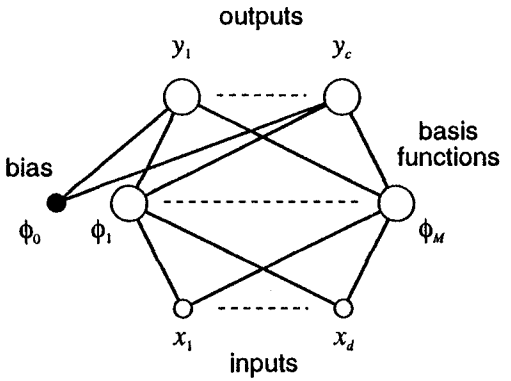

Figure 5.2. Architecture of a radial basis function neural network, corresponding to (5.14). Each basis function acts like a hidden unit. The lines connecting basis function ϕ_j to the inputs represent the corresponding elements μ_{ji} of the vector $\boldsymbol{\mu}_j$. The weights w_{kj} are shown as lines from the basis functions to the output units, and the biases are shown as weights from an extra 'basis function' ϕ_0 whose output is fixed at 1.

Girosi and Poggio (1990) have shown that radial basis function networks possess the property of *best approximation*. An approximation scheme has this property if, in the set of approximating functions (i.e. the set of functions corresponding to all possible choices of the adjustable parameters) there is one function which has minimum approximating error for any given function to be approximated. They also showed that this property is not shared by multi-layer perceptrons.

The Gaussian radial basis functions considered above can be generalized to allow for arbitrary covariance matrices $\boldsymbol{\Sigma}_j$, as discussed for normal probability density functions in Section 2.1.1. Thus we take the basis functions to have the form

$$\phi_j(\mathbf{x}) = \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu}_j)^{\mathrm{T}}\boldsymbol{\Sigma}_j^{-1}(\mathbf{x}-\boldsymbol{\mu}_j)\right\}. \tag{5.16}$$

Since the covariance matrices $\boldsymbol{\Sigma}_j$ are symmetric, this means that each basis function has $d(d+3)/2$ independent adjustable parameters (where d is the dimensionality of the input space), as compared with the $(d+1)$ independent parameters for the basis functions (5.15). In practice there is a trade-off to be considered between using a smaller number of basis with many adjustable parameters and a larger number of less flexible functions.

5.3 Network training

A key aspect of radial basis function networks is the distinction between the roles of the first and second layers of weights. As we shall see, the basis functions can be interpreted in a way which allows the first-layer weights (i.e. the parameters governing the basis functions) to be determined by unsupervised training techniques. This leads to the following two-stage training procedure for training radial basis function networks. In the first stage the input data set $\{\mathbf{x}^n\}$ alone is used to determine the parameters of the basis functions (e.g. $\boldsymbol{\mu}_j$ and σ_j for the spherical Gaussian basis functions considered above). The basis functions are then kept fixed while the second-layer weights are found in the second phase of training. Techniques for optimizing the basis functions are discussed at length in Section 5.9. Here we shall assume that the basis function parameters have already been chosen, and we discuss the problem of optimizing the second-layer weights. Note that, if there are fewer basis functions than data points, then in general it will no longer possible to find a set of weight values for which the mapping function fits the data points exactly.

We begin by considering the radial basis function network mapping in (5.14) and we absorb the bias parameters into the weights to give

$$y_k(\mathbf{x}) = \sum_{j=0}^{M} w_{kj}\phi_j(\mathbf{x}) \tag{5.17}$$

where ϕ_0 is an extra 'basis function' with activation value fixed at $\phi_0 = 1$. This can be written in matrix notation as

$$\mathbf{y}(\mathbf{x}) = \mathbf{W}\boldsymbol{\phi} \tag{5.18}$$

where $\mathbf{W} = (w_{kj})$ and $\boldsymbol{\phi} = (\phi_j)$. Since the basis functions are considered fixed, the network is equivalent to a single-layer network of the kind considered in Section 3.3 in the context of classification problems, where it is termed a generalized linear discriminant. As discussed in earlier chapters, we can optimize the weights by minimization of a suitable error function. It is particularly convenient, as we shall see, to consider a sum-of-squares error function given by

$$E = \frac{1}{2}\sum_n \sum_k \{y_k(\mathbf{x}^n) - t_k^n\}^2 \tag{5.19}$$

where t_k^n is the target value for output unit k when the network is presented with input vector $\mathbf{x}^n$. Since the error function is a quadratic function of the weights, its minimum can be found in terms of the solution of a set of linear equations. This problem was discussed in detail in Section 3.4.3, from which we see that the weights are determined by the linear equations

$$\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\mathbf{W}^{\mathrm{T}} = \mathbf{\Phi}^{\mathrm{T}}\mathbf{T} \tag{5.20}$$

where $(\mathbf{T})_{nk} = t_k^n$ and $(\mathbf{\Phi})_{nj} = \phi_j(\mathbf{x}^n)$. The formal solution for the weights is given by

$$\mathbf{W}^{\mathrm{T}} = \mathbf{\Phi}^{\dagger}\mathbf{T} \tag{5.21}$$

where the notation $\mathbf{\Phi}^{\dagger}$ denotes the pseudo-inverse of $\mathbf{\Phi}$ (Section 3.4.3). In practice, the equations (5.20) are solved using singular value decomposition, to avoid problems due to possible ill-conditioning of the matrix $\mathbf{\Phi}$. Thus, we see that the second-layer weights can be found by fast, linear matrix inversion techniques.

For the most part we shall consider radial basis function networks in which the dependence of the network function on the second-layer weights is linear, and in which the error function is given by the sum-of-squares. It is possible to consider the use of non-linear activation functions applied to the output units, or other choices for the error function. However, the determination of the second-layer weights is then no longer a linear problem, and hence a non-linear optimization of these weights is then required. As we have indicated, one of the major advantages of radial basis function networks is the possibility of avoiding the need for such an optimization during network training.

As a simple illustration of the use of radial basis function networks, we return to the data set shown in Figure 5.1 and consider the mapping obtained by using a radial basis function network in which the number of basis functions is smaller than the number of data points, as shown in Figure 5.3

The width parameter σ in Figure 5.3 was chosen to be roughly twice the average spacing between the basis functions. Techniques for setting the basis function parameters, including σ_j, are discussed in detail in Section 5.9. Here we simply note the effect of poor choices of σ. Figure 5.4 shows the result of choosing too small a value for σ, while the effect of having σ too large is illustrated in Figure 5.5.

5.4 Regularization theory

An alternative motivation for radial basis function expansions comes from the theory of regularization (Poggio and Girosi, 1990a, 1990b). In Section 1.6 the technique of regularization was introduced as a way of controlling the smoothness properties of a mapping function. It involves adding to the error function an extra term which is designed to penalize mappings which are not smooth. For simplicity of notation we shall consider networks having a single output y, so that with a sum-of-squares error, the total error function to be minimized becomes

$$E = \frac{1}{2}\sum_n \{y(\mathbf{x}^n) - t^n\}^2 + \frac{\nu}{2}\int |Py|^2 \, d\mathbf{x} \tag{5.22}$$

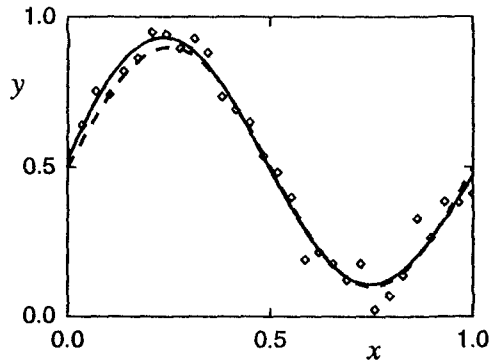

Figure 5.3. This shows the same set of 30 data points as in Figure 5.1, together with a network mapping (solid curve) in which the number of basis functions has been set to 5, which is significantly fewer than the number of data points. The centres of the basis functions have been set to a random subset of the data set input vectors, and the width parameters of the basis functions have been set to a common value of $\sigma = 0.4$, which again is roughly equal to twice the average spacing between the centres. The second-layer weights are found by minimizing a sum-of-squares error function using singular value decomposition.

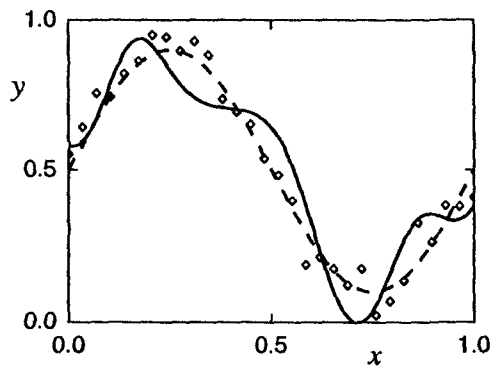

Figure 5.4. As in Figure 5.3, but in which the width parameter has been set to $\sigma = 0.08$. The resulting network function is insufficiently smooth and gives a poor representation of the underlying function which generated the data.

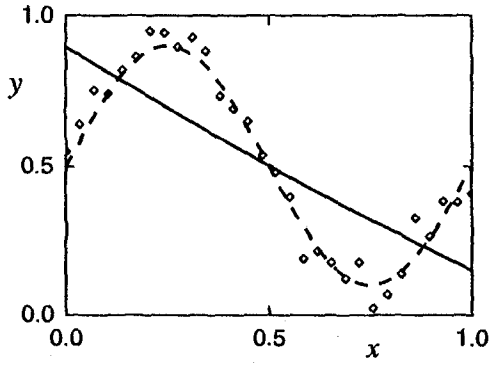

Figure 5.5. As in Figure 5.3, but in which the width parameter has been set to $\sigma = 10.0$. This leads to a network function which is over-smoothed, and which again gives a poor representation of the underlying function which generated the data.

where P is some differential operator, and ν is called a regularization parameter. Network mapping functions $y(\mathbf{x})$ which have large curvature will typically give rise to large values of $|Py|^2$ and hence to a large penalty in the total error function. The value of ν controls the relative importance of the regularization term, and hence the degree of smoothness of the function $y(\mathbf{x})$.

We can solve the regularized least-squares problem of (5.22) by using calculus of variations (Appendix D) as follows. Setting the functional derivative of (5.22) with respect to $y(\mathbf{x})$ to zero we obtain

$$\sum_n \{y(\mathbf{x}^n) - t^n\}\delta(\mathbf{x} - \mathbf{x}^n) + \nu \widehat{P}Py(\mathbf{x}) = 0 \tag{5.23}$$

where $\widehat{P}$ is the *adjoint* differential operator to P and $\delta(\mathbf{x})$ is the Dirac delta function. The equations (5.23) are the *Euler–Lagrange* equations corresponding to (5.22). A formal solution to these equations can be written down in terms of the *Green's functions* of the operator $\widehat{P}P$, which are the functions $G(\mathbf{x}, \mathbf{x}')$ which satisfy

$$\widehat{P}PG(\mathbf{x}, \mathbf{x}') = \delta(\mathbf{x} - \mathbf{x}'). \tag{5.24}$$

If the operator P is translationally and rotationally invariant, then the Green's functions depend only on the distance $\|\mathbf{x} - \mathbf{x}'\|$, and hence they are radial functions. The formal solution to (5.23) can then be written as

$$y(\mathbf{x}) = \sum_n w_n G(\|\mathbf{x} - \mathbf{x}^n\|) \tag{5.25}$$

which has the form of a linear expansion in radial basis functions. Substituting (5.25) into (5.23) and using (5.24) we obtain

$$\sum_n \{y(\mathbf{x}^n) - t^n\}\delta(\mathbf{x} - \mathbf{x}^n) + \nu \sum_n w_n \delta(\mathbf{x} - \mathbf{x}^n) = 0 \tag{5.26}$$

Integrating over a small region around $\mathbf{x}^n$ shows that the coefficients w_n satisfy

$$y(\mathbf{x}^n) - t^n + \nu w_n = 0. \tag{5.27}$$

Values for the coefficients w_n can be found by evaluating (5.25) at the values of the training data points $\mathbf{x}^n$ and substituting into (5.27). This gives the values of w_n as the solutions of the linear equation

$$(\mathbf{G} + \nu\mathbf{I})\mathbf{w} = \mathbf{t} \tag{5.28}$$

where $(\mathbf{G})_{nn'} = G(\|\mathbf{x}^{n'} - \mathbf{x}^n\|)$, $(\mathbf{w})_n = w_n$, $(\mathbf{t})_n = t^n$ and $\mathbf{I}$ denotes the unit matrix.

If the operator P is chosen to have the particular form

$$\int |Py|^2 d\mathbf{x} = \sum_{l=0}^{\infty} \frac{\sigma^{2l}}{l!2^l} \int \left|D^l y(\mathbf{x})\right|^2 d\mathbf{x} \tag{5.29}$$

where $D^{2l} = (\nabla^2)^l$ and $D^{2l+1} = \nabla(\nabla^2)^l$, with ∇ and ∇^2 denoting the gradient and Laplacian operators respectively, then the Green's functions are Gaussians with width parameters σ (Exercise 5.3).

We see that there is a very close similarity between this form of basis function expansion, and the one discussed in the context of exact interpolation in Section 5.1. Here the Greens functions $G(\|\mathbf{x}-\mathbf{x}^n\|)$ correspond to the basis functions $\phi(\|\mathbf{x} - \mathbf{x}^n\|)$, and there is one such function centred on each data point in the training set. Also, we see that (5.28) reduces to the exact interpolation result (5.3) when the regularization parameter ν is zero. When the regularization parameter is greater than zero, however, we no longer have an exact interpolating function. The effect of the regularization term is to force a smoother network mapping function, as illustrated in Figure 5.6.

In practice, regularization can also be applied to radial basis function networks in which the basis functions are not constrained to be centred on the data points, and in which the number of basis functions need not equal the number of data points. Also, regularization terms can be considered for which the basis functions are not necessarily the Green's functions. Provided the regularization term is a quadratic function of the network mapping, the second-layer weights

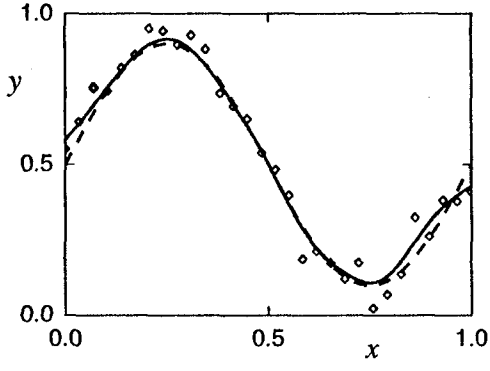

Figure 5.6. This shows the same data set as in Figure 5.1, again with one basis function centred on each data point, and a width parameter $\sigma = 0.067$. In this case, however, a regularization term is used, with coefficient $\nu = 40$, leading to a smoother mapping (shown by the solid curve) which no longer gives an exact fit to the data, but which now gives a much better approximation to the underlying function which generated the data (shown by the dashed curve).

can again be found by the solution of a set of linear equations which minimize a sum-of-squares error. For example, the regularizer

$$\frac{\nu}{2} \sum_n \sum_k \sum_i \left(\frac{\partial^2 y_k^n}{\partial x_i^2} \right)^2 \tag{5.30}$$

penalizes mappings which have large curvature (Bishop, 1991b). This regularizer leads to second-layer weights which are found by solution of

$$\mathbf{M}\mathbf{W} = \mathbf{\Phi}^{\mathrm{T}}\mathbf{T} \tag{5.31}$$

where

$$(\mathbf{M})_{jj'} = \sum_n \left\{ \phi_j^n \phi_{j'}^n + \nu \sum_i \left(\frac{\partial^2 \phi_j^n}{\partial x_i^2} \frac{\partial^2 \phi_{j'}^n}{\partial x_i^2} \right) \right\} \tag{5.32}$$

and $\mathbf{\Phi} = (\phi_j^n)$ as before. When $\nu = 0$ (5.31) reduces to the previous result (5.20). The inclusion of the regularization term adds little to the computational cost, since most of the time is spent in solving the coupled linear equations (5.31).

5.5 Noisy interpolation theory

Yet another viewpoint on the origin of radial basis function expansions comes from the theory of interpolation of noisy data (Webb, 1994). Consider a mapping from a single input variable x to a single output variable y in which the target data is generated from a smooth noise-free function $h(x)$ but in which the *input* data is corrupted by additive noise. The sum-of-squares error, in the limit of infinite data, takes the form

$$E = \frac{1}{2}\int\int \{y(x+\xi) - h(x)\}^2 \widetilde{p}(\xi) p(x)\, d\xi\, dx \tag{5.33}$$

where $p(x)$ is the probability density function of the input data, and $\widetilde{p}(\xi)$ is the probability density function of the noise. Changing variables using $z = x + \xi$ we have

$$E = \frac{1}{2}\int\int \{y(z) - h(x)\}^2 \widetilde{p}(z-x) p(x)\, dz\, dx. \tag{5.34}$$

A formal expression for the minimum of the error can then be obtained using variational techniques (Appendix D) by setting the functional derivative of E with respect to $y(z)$ to zero, to give

$$y(z) = \frac{\int h(x)\widetilde{p}(z-x)p(x)\, dx}{\int \widetilde{p}(z-x)p(x)\, dx}. \tag{5.35}$$

If we consider the case of a finite number of data points $\{x^n\}$ drawn from the distribution $p(x)$, we can approximate (5.35) by

$$y(x) = \frac{\sum_n h(x^n)\widetilde{p}(x-x^n)}{\sum_n \widetilde{p}(x-x^n)} \tag{5.36}$$

which we recognize as being an expansion in radial basis functions, in which $h(x^n)$ are the expansion coefficients, and the basis functions are given by

$$\phi(x-x^n) = \frac{\widetilde{p}(x-x^n)}{\sum_n \widetilde{p}(x-x^n)}. \tag{5.37}$$

Since the function $h(x)$ is unknown, the coefficients $h(x^n)$ should be regarded as parameters to be determined from the data. To do this we note that $h(x)$ is noise-free and so we have $h(x^n) = t^n$. Thus (5.36) becomes an expansion in basis functions in which the coefficients are given by the target values. Note that this form of basis function expansion differs from that introduced in (5.14) and (5.15)

in that the basis functions are normalized (Moody and Darken, 1989). Strictly speaking, the normalization in (5.36) would require lateral connections between different hidden units in a network diagram. If the distribution of the noise is normal, so that $\widetilde{p}(\xi) \propto \exp(-\xi^2/2\sigma^2)$, then we obtain an expansion in Gaussian basis functions

$$y(x) = \frac{\sum_n h(x^n) \exp\{-(x - x^n)^2/2\sigma^2\}}{\sum_n \exp\{-(x - x^n)^2/2\sigma^2\}}. \tag{5.38}$$

The extension of this result to several output variables is straightforward and gives

$$y_k(x) = \frac{\sum_n h_k(x^n) \exp\{-(x - x^n)^2/2\sigma^2\}}{\sum_n \exp\{-(x - x^n)^2/2\sigma^2\}}. \tag{5.39}$$

Note that (5.36) will only be a good approximation to (5.35) if the integrand is sufficiently smooth. This implies that the width of the basis functions should be large in relation to the spacing of the data, which is a useful rule of thumb when designing networks with good generalization properties.

5.6 Relation to kernel regression

Further motivation for the use of radial basis functions for function approximation comes from the theory of kernel regression (Scott, 1992). This is a technique for estimating regression functions from noisy data, based on the methods of kernel density estimation discussed in Section 2.5.3. Consider a mapping from an input vector $\mathbf{x}$ to an output vector $\mathbf{y}$, and suppose we are given a set of training data $\{\mathbf{x}^n, \mathbf{t}^n\}$ where $n = 1, \ldots, N$. A complete description of the statistical properties of the generator of the data is given by the probability density $p(\mathbf{x}, \mathbf{t})$ in the joint input-target space. We can model this density by using a Parzen kernel estimator constructed from the data set. If we consider Gaussian kernel functions, this estimator takes the form

$$\widehat{p}(\mathbf{x}, \mathbf{t}) = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{(2\pi h^2)^{(d+c)/2}} \exp\left\{-\frac{\|\mathbf{x} - \mathbf{x}^n\|^2}{2h^2} - \frac{\|\mathbf{t} - \mathbf{t}^n\|^2}{2h^2}\right\} \tag{5.40}$$

where d and c are the dimensionalities of the input and output spaces respectively. This is illustrated schematically, for the case of one input variable and one output variable, in Figure 5.7.

As we have already seen, the goal of learning is to find a smooth mapping from $\mathbf{x}$ to $\mathbf{y}$ which captures the underlying systematic aspects of the data, without fitting the noise on the data. In Section 6.1.3 it is shown that, under many circumstances, the optimal mapping is given by forming the regression, or conditional average $\langle \mathbf{t}|\mathbf{x} \rangle$, of the target data, conditioned on the input variables. This

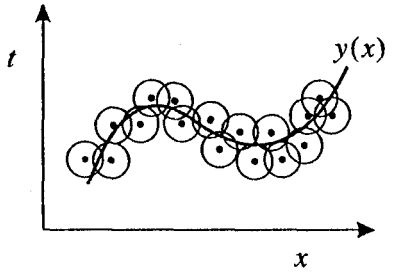

Figure 5.7. Schematic illustration of the use of a kernel estimator to model the joint probability density in the input-output space. The dots show the data points, and the circles represent Gaussian kernel functions centred on the data points, while the curve shows the regression function given by the conditional average of t as a function of x.

can be expressed in terms of the conditional density $p(\mathbf{t}|\mathbf{x})$, and hence in terms of the joint density $p(\mathbf{x}, \mathbf{t})$, as follows:

$$\begin{aligned} \mathbf{y}(\mathbf{x}) &= \langle \mathbf{t}|\mathbf{x} \rangle \\ &= \int \mathbf{t} p(\mathbf{t}|\mathbf{x})\, d\mathbf{t} \\ &= \frac{\int \mathbf{t} p(\mathbf{x}, \mathbf{t})\, d\mathbf{t}}{\int p(\mathbf{x}, \mathbf{t})\, d\mathbf{t}}. \end{aligned} \tag{5.41}$$

If we now substitute our density estimate (5.40) into (5.41) we obtain the following expression for the regression of the target data

$$\mathbf{y}(\mathbf{x}) = \frac{\sum_n \mathbf{t}^n \exp\left\{-\|\mathbf{x} - \mathbf{x}^n\|^2/2h^2\right\}}{\sum_n \exp\left\{-\|\mathbf{x} - \mathbf{x}^n\|^2/2h^2\right\}}. \tag{5.42}$$

This is known as the *Nadaraya–Watson* estimator (Nadaraya, 1964; Watson, 1964), and has been re-discovered relatively recently in the context of neural networks (Specht, 1990; Schiøler and Hartmann, 1992). We see that (5.42) has the form of a normalized expansion in Gaussian radial basis functions defined in the input space, and should be compared with the form (5.38) obtained earlier from the perspective of additive noise on the input data. Each basis function is centred on a data point, and the coefficients in the expansion are given by the target values $\mathbf{t}^n$. Note that this construction provides values for the hidden-to-

output unit weights which are just given by the target data values.

This approach can be extended by replacing the kernel estimator with an adaptive mixture model, as discussed in Section 2.6. The parameters of the mixture model can be found using, for instance, the EM (expectation–maximization) algorithm (Section 2.6.2). For a mixture of M spherical Gaussian functions, we can write the joint density in the form

$$\widehat{p}(\mathbf{x}, \mathbf{t}) = \sum_{j=1}^{M} P(j) \frac{1}{(2\pi h^2)^{(d+c)/2}} \exp\left\{ -\frac{\|\mathbf{x} - \boldsymbol{\mu}_j\|^2}{2h^2} - \frac{\|\mathbf{t} - \boldsymbol{\nu}_j\|^2}{2h^2} \right\}. \tag{5.43}$$

Following the same line of argument as before, we arrive at the following expression for the regression:

$$\mathbf{y}(\mathbf{x}) = \frac{\sum_j P(j) \boldsymbol{\nu}_j \exp\left\{-\|\mathbf{x} - \boldsymbol{\mu}_j\|^2 / 2h^2\right\}}{\sum_j P(j) \exp\left\{-\|\mathbf{x} - \boldsymbol{\mu}_j\|^2 / 2h^2\right\}} \tag{5.44}$$

which can be viewed as a normalized radial basis function expansion in which the number of basis functions is typically much smaller than the number of data points, and in which the basis function centres are no longer constrained to coincide with the data points. This result can be extended to Gaussian functions with general covariance matrices (Ghahramani and Jordan, 1994b).

5.7 Radial basis function networks for classification

A further key insight into the nature of the radial basis function network is obtained by considering the use of such networks for classification problems (Lowe, 1995). Suppose we have a data set which falls into three classes as shown in Figure 5.8. A multi-layer perceptron can separate the classes by using hidden units which form hyperplanes in the input space, as indicated in Figure 5.8(a). An alternative approach is to model the separate class distributions by local kernel functions, as indicated in (b). This latter type of representation is related to the radial basis function network.

Suppose we model the data in each class $\mathcal{C}_k$ using a single kernel function, which we write as $p(\mathbf{x}|\mathcal{C}_k)$. In a classification problem our goal is to model the posterior probabilities $p(\mathcal{C}_k|\mathbf{x})$ for each of the classes. These probabilities can be obtained through Bayes' theorem, using prior probabilities $p(\mathcal{C}_k)$, as follows:

$$P(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k)}{p(\mathbf{x})} \tag{5.45}$$

$$= \frac{p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k)}{\sum_{k'} p(\mathbf{x}|\mathcal{C}_{k'})P(\mathcal{C}_{k'})}. \tag{5.46}$$

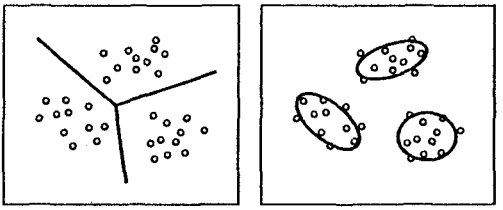

Figure 5.8. Schematic example of data points in two dimensions which fall into three distinct classes. One way to separate the classes is to use hyperplanes, shown in (a), as used in a multi-layer perceptron. An alternative approach, shown in (b), is to fit each class with a kernel function, which gives the type of representation formed by a radial basis function network.

This can be viewed as a simple form of basis function network with normalized basis functions given by

$$\phi_k(\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)}{\sum_{k'} p(\mathbf{x}|\mathcal{C}_{k'})P(\mathcal{C}_{k'})} \tag{5.47}$$

and second-layer connections which consist of one weight from each hidden unit going to the corresponding output unit, with value $p(\mathcal{C}_k)$. The outputs of this network represent approximations to the posterior probabilities.

In most applications a single kernel function will not give a particularly good representation of the class-conditional distributions $p(\mathbf{x}|\mathcal{C}_k)$. A better representation could be obtained by using a separate mixture model to represent each of the conditional densities. However, a computationally more efficient approach, and one which may help to reduce the number of adjustable parameters in the model, is to use a common pool of M basis functions, labelled by an index j, to represent all of the class-conditional densities. Thus, we write

$$p(\mathbf{x}|\mathcal{C}_k) = \sum_{j=1}^{M} p(\mathbf{x}|j)P(j|\mathcal{C}_k). \tag{5.48}$$

An expression for the unconditional density $p(\mathbf{x})$ can be found from (5.48) by summing over all classes

$$p(\mathbf{x}) = \sum_k p(\mathbf{x}|\mathcal{C}_k)P(\mathcal{C}_k) \tag{5.49}$$

$$= \sum_{j=1}^{M} p(\mathbf{x}|j)P(j) \tag{5.50}$$

where we have defined priors for the basis functions given by

$$P(j) = \sum_{k} P(j|\mathcal{C}_k)P(\mathcal{C}_k). \tag{5.51}$$

Again, the quantities we are interested in are the posterior probabilities of class membership. These can be obtained by substituting the expressions (5.48) and (5.50) into Bayes' theorem (5.45) to give

$$P(\mathcal{C}_k|\mathbf{x}) = \frac{\sum_{j=1}^{M} P(j|\mathcal{C}_k)p(\mathbf{x}|j)P(\mathcal{C}_k)}{\sum_{j'=1}^{M} p(\mathbf{x}|j')P(j')} \frac{P(j)}{P(j)} \tag{5.52}$$

$$= \sum_{j=1}^{M} w_{kj}\phi_j(\mathbf{x}) \tag{5.53}$$

where we have inserted an extra factor of $1 = P(j)/P(j)$ into (5.52). The expression (5.53) represents a radial basis function network, in which the normalized basis functions are given by

$$\phi_j(\mathbf{x}) = \frac{p(\mathbf{x}|j)P(j)}{\sum_{j'=1}^{M} p(\mathbf{x}|j')P(j')} \tag{5.54}$$

$$= P(j|\mathbf{x}) \tag{5.55}$$

and the second-layer weights are given by

$$w_{kj} = \frac{P(j|\mathcal{C}_k)P(\mathcal{C}_k)}{P(j)} \tag{5.56}$$

$$= P(\mathcal{C}_k|j). \tag{5.57}$$

Thus, the activations of the basis functions can be interpreted as the posterior probabilities of the presence of corresponding features in the input space, and the weights can similarly be interpreted as the posterior probabilities of class membership, given the presence of the features. The activations of the hidden units in a multi-layer perceptron (with logistic sigmoid activation functions) can be given a similar interpretation as posterior probabilities of the presence of features, as discussed in Section 6.7.1.

Note from (5.50) that the unconditional density of the input data is expressed

in terms of a mixture model, in which the component densities are given by the basis functions. This motivates the use of mixture density estimation as a procedure for finding the basis function parameters, as discussed in Section 5.9.4.

It should be emphasized that the outputs of this network also have a precise interpretation as the posterior probabilities of class membership. The ability to interpret network outputs in this way is of central importance in the effective application of neural networks, and is discussed at length in Chapter 6.

Finally, for completeness, we point out that radial basis functions are also closely related to the method of *potential functions* (Aizerman *et al.*, 1964; Niranjan *et al.*, 1989). This is a way of finding a linear discriminant function from a training set of data points, based on an analogy with electrostatics. Imagine we place a unit of positive charge at each point in input space at which there is a training vector from class $\mathcal{C}_1$, and a unit of negative charge at each point where there is a training vector from class $\mathcal{C}_2$. These charges give rise to an electrostatic potential field which can be treated as a discriminant function. The kernel function which is used to compute the contribution to the potential from each charge need not be that of conventional electrostatics, but can be some other function of the radial distance from the data point.

5.8 Comparison with the multi-layer perceptron

Radial basis function networks and multi-layer perceptrons play very similar roles in that they both provide techniques for approximating arbitrary non-linear functional mappings between multidimensional spaces. In both cases the mappings are expressed in terms of parametrized compositions of functions of single variables. The particular structures of the two networks are very different, however, and so it is interesting to compare them in more detail. Some of the important differences between the multi-layer perceptron and radial basis function networks are as follows:

1. The hidden unit representations of the multi-layer perceptron depend on weighted linear summations of the inputs, transformed by monotonic activation functions. Thus the activation of a hidden unit in a multi-layer perceptron is constant on surfaces which consist of parallel $(d-1)$-dimensional hyperplanes in d-dimensional input space. By contrast, the hidden units in a radial basis function network use distance to a prototype vector followed by transformation with a (usually) localized function. The activation of a basis function is therefore constant on concentric $(d-1)$-dimensional hyperspheres (or more generally on $(d-1)$-dimensional hyperellipsoids).
2. A multi-layer perceptron can be said to form a *distributed representation* in the space of activation values for the hidden units since, for a given input vector, many hidden units will typically contribute to the determination of the output value. During training, the functions represented by the hidden units must be such that, when linearly combined by the final layer of weights, they generate the correct outputs for a range of possible input values. The interference and cross-coupling between the hidden units which

this requires results in the network training process being highly non-linear with problems of local minima, or nearly flat regions in the error function arising from near cancellations in the effects of different weights. This can lead to very slow convergence of the training procedure even with advanced optimization strategies. By contrast, a radial basis function network with localized basis functions forms a representation in the space of hidden units which is *local* with respect to the input space because, for a given input vector, typically only a few hidden units will have significant activations.

3. A multi-layer perceptron often has many layers of weights, and a complex pattern of connectivity, so that not all possible weights in any given layer are present. Also, a variety of different activation functions may be used within the same network. A radial basis function network, however, generally has a simple architecture consisting of two layers of weights, in which the first layer contains the parameters of the basis functions, and the second layer forms linear combinations of the activations of the basis functions to generate the outputs.
4. All of the parameters in a multi-layer perceptron are usually determined at the same time as part of a single global training strategy involving supervised training. A radial basis function network, however, is typically trained in two stages, with the basis functions being determined first by unsupervised techniques using the input data alone, and the second-layer weights subsequently being found by fast linear supervised methods.

5.9 Basis function optimization

One of the principal advantages of radial basis function neural networks, as compared with the multi-layer perceptron, is the possibility of choosing suitable parameters for the hidden units without having to perform a full non-linear optimization of the network. In this section we shall discuss several possible strategies for selecting the parameters of the basis functions. The problem of selecting the appropriate number of basis functions, however, is discussed in the context of model order selection and generalization in Chapter 9.

We have motivated radial basis functions from the perspectives of function approximation, regularization, noisy interpolation, kernel regression, and the estimation of posterior class probabilities for classification problems. All of these viewpoints suggest that the basis function parameters should be chosen to form a representation of the probability density of the input data. This leads to an unsupervised procedure for optimizing the basis function parameters which depends only on the input data from the training set, and which ignores any target information. The basis function centres $\boldsymbol{\mu}_j$ can then be regarded as *prototypes* of the input vectors. In this section we discuss a number of possible strategies for optimizing the basis functions which are motivated by these considerations.

There are many potential applications for neural networks where unlabelled input data is plentiful, but where labelled data is in short supply. For instance, it may be easy to collect examples of raw input data for the network, but the

labelling of the data with target variables may require the time of a human expert which therefore limits the amount of data which can be labelled in a reasonable time. With such applications, the two-stage training process for a radial basis function network can be particularly advantageous since the determination of the non-linear representation given by first layer of the network can be done using a large quantity of unlabelled data, leaving a relatively small number of parameters in the second layer to be determined using the labelled data. At each stage of the training process, we can ensure that the number of data points is large compared with the number of parameters to be determined, as required for good generalization.

One of the major potential difficulties with radial basis function networks, however, also stems from the localized nature of the hidden unit representation. It concerns the way in which such a network addresses the curse of dimensionality discussed in Section 1.4. There we saw that the number of hypercubes which are needed to fill out a compact region of a d-dimensional space grows exponentially with d. When the data is confined to some lower-dimensional sub-space, d is to be interpreted as the effective dimensionality of the sub-space, known as the *intrinsic dimensionality* of the data. If the basis function centres are used to fill out the sub-space then the number of basis function centres will be an exponential function of d (Hartman *et al.*, 1990). As well as increasing the computation time, a large number of basis functions leads to a requirement for large numbers of training patterns in order to ensure that the network parameters are properly determined.

The problem is particularly severe if there are input variables which have significant variance but which play little role in determining the appropriate output variables. Such irrelevant inputs are not uncommon in practical applications. When the basis function centres are chosen using the input data alone, there is no way to distinguish relevant from irrelevant inputs. This problem is illustrated in Figure 5.9 where we see a variable y which is a non-linear function of an input variable x_1. We wish to use radial basis function network network to approximate this function. The basis functions are chosen to cover the region of the x_1 axis where data is observed. Suppose that a second input variable x_2 is introduced which is uncorrelated with x_1. Then the number of basis functions needed to cover the required region of input space increases dramatically as indicated in Figure 5.10. If y is independent of x_2 then these extra basis functions have no useful role in determining the value of y. Simulations using artificial data (Hartman *et al.*, 1990), in which 19 out of 20 input variables consisted of noise uncorrelated with the output, showed that a multi-layer perceptron could learn to ignore the irrelevant inputs and obtain accurate results with a small number of hidden units, while radial basis function networks showed large error which decreased only slowly as the number of hidden units was increased.

Problems arising from the curse of dimensionality may be much less severe if basis functions with full covariance matrices are used, as in (5.16), rather than spherical basis functions of the form (5.15). However, the number of parameters per basis function is then much greater.

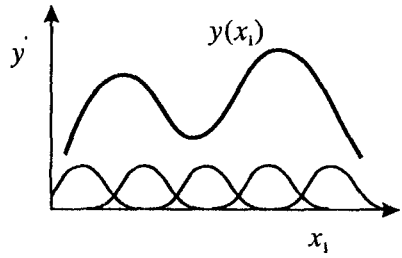

Figure 5.9. A schematic example of a function $y(x_1)$ of an input variable x_1 which has been modelled using a set of radial basis functions.

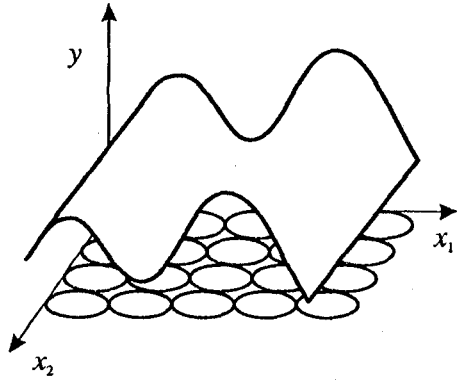

Figure 5.10. As in Figure 5.9, but in which an extra, irrelevant variable x_2 has been introduced. Note that the number of basis functions, whose locations are determined using the input data alone, has increased dramatically, even though x_2 carries no useful information for determining the output variable.

We have provided compelling reasons for using unsupervised methods to determine the first-layer parameters in a radial basis function network by modelling the density of input data. Such method have also proven to be very powerful in practice. However, it should be emphasized that the optimal choice of basis function parameters for density estimation need not be optimal for representing the mapping to the output variables. Figure 5.11 shows a simple example of a problem for which the use of density estimation to set the basis function parameters clearly gives a sub-optimal solution.

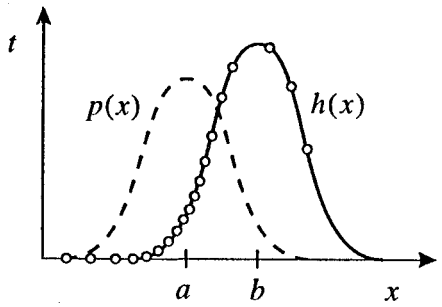

Figure 5.11. A simple example to illustrate why the use of unsupervised methods based on density estimation to determine the basis function parameters need not be optimal for approximating the target function. Data in one dimension (shown by the circles) is generated from a Gaussian distribution $p(x)$ shown by the dashed curve. Unsupervised training of one Gaussian basis function would cause it to be centred at $x = a$, giving a good approximation to $p(x)$. Target values for the input data are generated from a Gaussian function centred at b shown by the solid curve. The basis function centred at a can only give a very poor representation of $h(x)$. By contrast, if the basis function were centred at b it could represent the function $h(x)$ exactly.

5.9.1 *Subsets of data points*

One simple procedure for selecting the basis function centres $\boldsymbol{\mu}_j$ is to set them equal to a random subset of the input vectors from the training set, as was done for the example shown in Figure 5.3. Clearly this is not an optimal procedure so far as density estimation is concerned, and may also lead to the use of an unnecessarily large number of basis functions in order to achieve adequate performance on the training data. This method is often used, however, to provide a set of starting values for many of the iterative adaptive procedures to be discussed shortly.

Another approach is to start with all data points as basis functions centres and then selectively remove centres in such a way as to have minimum disruption on the performance of the system. Such an approach was introduced into the K-nearest-neighbour classification scheme by Devijver and Kittler (1982) and applied to radial basis function networks used for classification by Kraaijveld and Duin (1991). A procedure for selecting a subset of the basis functions so as to preserve the best estimator of the unconditional density is given in Fukunaga and Hayes (1989).

These techniques only set the basis function centres, and the width parameters σ_j must be chosen using some other procedure. One heuristic approach is to choose all the σ_j to be equal and to be given by some multiple of the average distance between the basis function centres. This ensures that the basis func-

tions overlap to some degree and hence give a relatively smooth representation of the distribution of training data. We might also recognize that the optimal width may be different for basis functions in different regions of input space. For instance, the widths may be determined from the average distance of each basis function to its L nearest neighbours, where L is typically small. Such *ad hoc* procedures for choosing the basis function parameters are very fast, and allow a radial basis function network to be set up very quickly, but are likely to be significantly sub-optimal.

5.9.2 *Orthogonal least squares*

A more principled approach to selecting a sub-set of the data points as basis function centres is based on the technique of *orthogonal least squares*. To motivate this approach consider the following procedure for selecting basis functions. We start by considering a network with just one basis function. For each data point in turn we set the basis function centre to the input vector for that data point, and then set the second-layer weights by pseudo-inverse techniques using the complete training set of N data points. The basis function centre which gives rise to the smallest residual error is retained. In subsequent steps of the algorithm, the number of basis functions is then increased incrementally. If at some point in the algorithm l of the data points have been selected as basis function centres, then $N-l$ networks are trained in which each of the remaining $N-l$ data points in turn is selected as the centre for the additional basis function. The extra basis function which gives the smallest value for the residual sum-of-squares error is then retained, and the algorithm proceeds to the next stage.

Such an approach would be computationally intensive since at each step it would be necessary to obtain a complete pseudo-inverse solution for each possible choice of basis functions. A much more efficient procedure for achieving the same result is that of *orthogonal least squares* (Chen *et al.*, 1989, 1991). In outline, the algorithm involves the sequential addition of new basis functions, each centred on one of the data points, as described above. This is done by constructing a set of orthogonal vectors in the space S spanned by the vectors of hidden unit activations for each pattern in the training set (Section 3.4.2). It is then possible to calculate directly which data point should be chosen as the next basis function centre in order to produce the greatest reduction in residual sum-of-squares error. Values for the second-layer weights are also determined at the same time. If the algorithm is continued long enough then all data points will be selected, and the residual error will be zero. In order to achieve good generalization, the algorithm must be stopped before this occurs. This is the problem of model-order selection, and is discussed at length in Chapters 9 and 10.

5.9.3 *Clustering algorithms*

As an improvement on simply choosing a subset of the data points as the basis function centres, we can use clustering techniques to find a set of centres which more accurately reflects the distribution of the data points. Moody and Darken (1989) use the *K-means clustering algorithm*, in which the number K of centres

must be decided in advance. The algorithm involves a simple re-estimation procedure, as follows. Suppose there are N data points $\mathbf{x}^n$ in total, and we wish to find a set of K representative vectors $\boldsymbol{\mu}_j$ where $j = 1, \ldots, K$. The algorithm seeks to partition the data points $\{\mathbf{x}^n\}$ into K disjoint subsets $\mathcal{S}_j$ containing N_j data points, in such a way as to minimize the sum-of-squares clustering function given by

$$J = \sum_{j=1}^{K} \sum_{n \in \mathcal{S}_j} \|\mathbf{x}^n - \boldsymbol{\mu}_j\|^2 \tag{5.58}$$

where $\boldsymbol{\mu}_j$ is the mean of the data points in set $\mathcal{S}_j$ and is given by

$$\boldsymbol{\mu}_j = \frac{1}{N_j} \sum_{n \in \mathcal{S}_j} \mathbf{x}^n. \tag{5.59}$$

The batch version of K-means (Lloyd, 1982) begins by assigning the points at random to K sets and then computing the mean vectors of the points in each set. Next, each point is re-assigned to a new set according to which is the nearest mean vector. The means of the sets are then recomputed. This procedure is repeated until there is no further change in the grouping of the data points. It can be shown (Linde *et al.*, 1980) that at each such iteration the value of J will not increase. The calculation of the means can also be formulated as a stochastic on-line process (MacQueen, 1967; Moody and Darken, 1989). In this case, the initial centres are randomly chosen from the data points, and as each data point $\mathbf{x}^n$ is presented, the nearest $\boldsymbol{\mu}_j$ is updated using

$$\Delta \boldsymbol{\mu}_j = \eta(\mathbf{x}^n - \boldsymbol{\mu}_j) \tag{5.60}$$

where η is the learning rate parameter. Note that this is simply the Robbins–Monro procedure (Section 2.4.1) for finding the root of a regression function given by the derivative of J with respect to $\boldsymbol{\mu}_j$. Once the centres of the basis functions have been found in this way, the covariance matrices of the basis functions can be set to the covariances of the points assigned to the corresponding clusters.

Another unsupervised technique which has been used for assigning basis function centres is the Kohonen topographic feature map, also called a *self-organizing feature map* (Kohonen, 1982). This algorithm leads to placement of a set of prototype vectors in input space, each of which corresponds to a point on a regular grid in a (usually two-dimensional) feature-map space. When the algorithm has converged, prototype vectors corresponding to nearby points on the feature map grid have nearby locations in input space. This leads to a number of applications for this algorithm including the projection of data into a two-dimensional space for visualization purposes. However, the imposition of the topographic property, particularly if the data is not intrinsically two-dimensional (Section 8.6.1), may

lead to suboptimal placement of vectors.

5.9.4 *Gaussian mixture models*

We have already discussed a number of heuristic procedures for setting the basis function parameters such that the basis functions approximate the distribution of the input data. A more principled approach, however, is to recognize that this is essentially the mixture density estimation problem, which is discussed at length in Section 2.6. The basis functions of the neural network can be regarded as the components of a mixture density model, whose parameters are to be optimized by maximum likelihood. We therefore model the density of the input data by a mixture model of the form

$$p(\mathbf{x}) = \sum_{j=1}^{M} P(j)\phi_j(\mathbf{x}) \tag{5.61}$$

where the parameters $P(j)$ are the mixing coefficients, and $\phi_j(\mathbf{x})$ are the basis functions of the network. Note that the mixing coefficients can be regarded as prior probabilities for the data points to have been generated from the jth component of the mixture. The likelihood function is given by

$$\mathcal{L} = \prod_n p(\mathbf{x}^n) \tag{5.62}$$

and is maximized both with respect to the mixing coefficients $P(j)$, and with respect to the parameters of the basis functions. This maximization can be performed by computing the derivatives of $\mathcal{L}$ with respect to the parameters and using these derivatives in standard non-linear optimization algorithms (Chapter 7). Alternatively, the parameters can be found by re-estimation procedures based on the EM (expectation–maximization) algorithm, described in Section 2.6.2.

Once the mixture model has been optimized, the mixing coefficients $P(j)$ can be discarded, and the basis functions then used in the radial basis function network in which the second-layer weights are found by supervised training. By retaining the mixing coefficients, however, the density model $p(\mathbf{x})$ in (5.61) can be used to assign error bars to the network outputs, based on the degree of *novelty* of the input vectors (Bishop, 1994b).

It is interesting to note that the K-means algorithm can be seen as a particular limit of the EM optimization of a Gaussian mixture model. From Section 2.6.2, the EM update formula for a basis function centre is given by

$$\boldsymbol{\mu}_j^{\text{new}} = \frac{\sum_n P(j|\mathbf{x}^n)\mathbf{x}^n}{\sum_{n'} P(j|\mathbf{x}^{n'})} \tag{5.63}$$

where $P(j|\mathbf{x})$ is the posterior probability of basis function j, and is given in terms of the basis functions and the mixing coefficients, using Bayes' theorem, in the

form

$$P(j|\mathbf{x}) = \frac{P(j)\phi_j(\mathbf{x})}{p(\mathbf{x})} \tag{5.64}$$

where $p(\mathbf{x})$ is given by (5.61). Suppose we consider spherical Gaussian basis functions having a common width parameter σ. Then the ratio of the posterior probabilities of two of the basis functions, for a particular data point $\mathbf{x}^n$, is given by

$$\frac{P(j|\mathbf{x}^n)}{P(k|\mathbf{x}^n)} = \exp\left\{-\frac{\|\mathbf{x}^n - \boldsymbol{\mu}_j\|^2}{2\sigma^2} + \frac{\|\mathbf{x}^n - \boldsymbol{\mu}_k\|^2}{2\sigma^2}\right\}\frac{P(j)}{P(k)}. \tag{5.65}$$

If we now take the limit $\sigma \to 0$, we see that

$$\frac{P(j|\mathbf{x}^n)}{P(k|\mathbf{x}^n)} \to 0 \quad \text{if} \quad \|\mathbf{x}^n - \boldsymbol{\mu}_j\|^2 > \|\mathbf{x}^n - \boldsymbol{\mu}_k\|^2. \tag{5.66}$$

Thus, the probabilities for all of the kernels is zero except for the kernel whose centre vector $\boldsymbol{\mu}_k$ is closest to $\mathbf{x}^n$. In this limit, therefore, the EM update formula (5.63) reduces to the K-means update formula (5.59).

5.10 Supervised training

As we have already remarked, the use of unsupervised techniques to determine the basis function parameters is not in general an optimal procedure so far as the subsequent supervised training is concerned. The difficulty arises because the setting up of the basis functions using density estimation on the input data takes no account of the target labels associated with that data. In order to set the parameters of the basis functions to give optimal performance in computing the required network outputs we should include the target data in the training procedure. That is, we should perform supervised, rather than unsupervised, training.

The basis function parameters for regression can be found by treating the basis function centres and widths, along with the second-layer weights, as adaptive parameters to be determined by minimization of an error function. For the case of the sum-of-squares error (5.19), and spherical Gaussian basis functions (5.15), we obtain the following expressions for the derivatives of the error function with respect to the basis function parameters

$$\frac{\partial E}{\partial \sigma_j} = \sum_n \sum_k \{y_k(\mathbf{x}^n) - t_k^n\}\, w_{kj} \exp\left(-\frac{\|\mathbf{x}^n - \boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right)\frac{\|\mathbf{x}^n - \boldsymbol{\mu}_j\|^2}{\sigma_j^3} \tag{5.67}$$

$$\frac{\partial E}{\partial \mu_{ji}} = \sum_n \sum_k \{y_k(\mathbf{x}^n) - t_k^n\} w_{kj} \exp\left(-\frac{\|\mathbf{x}^n - \boldsymbol{\mu}_j\|^2}{2\sigma_j^2}\right) \frac{(x_i^n - \mu_{ji})}{\sigma_j^2} \quad (5.68)$$

where μ_{ji} denotes the ith component of $\boldsymbol{\mu}_j$. These expressions for the derivatives can then be used in conjunction with one of the standard optimization strategies discussed in Chapter 7.

The setting of the basis function parameters by supervised learning represents a non-linear optimization problem which will typically be computationally intensive and may be prone to finding local minima of the error function. However, provided the basis functions are reasonably well localized, any given input vector will only generate a significant activation in a small fraction of the basis functions, and so only these functions will be significantly updated in response to that input vector. Training procedures can therefore be speeded up significantly by identifying the relevant basis functions and thereby avoiding unnecessary computation. Techniques for finding these units efficiently are described by Omohundro (1987). Also, one of the unsupervised techniques described above can be used to initialize the basis function parameters, after which they can be 'fine tuned' using supervised procedures. However, one of the drawbacks of supervised training of the basis functions is that there is no guarantee that they will remain localized. Indeed, in numerical simulations it is found that a subset of the basis functions may evolve to have very broad responses (Moody and Darken, 1989). Also, some of the main advantages of radial basis function networks, namely fast two-stage training, and interpretability of the hidden unit representation, are lost if supervised training is adopted.

Exercises

5.1 (⋆) Consider a radial basis function network represented by (5.14) with Gaussian basis functions having full covariance matrices of the form (5.16). Derive expressions for the elements of the Jacobian matrix given by

$$J_{ki} = \frac{\partial y_k}{\partial x_i}. \quad (5.69)$$

5.2 (⋆⋆) Consider a radial basis function network with spherical Gaussian basis of the form (5.15), network outputs given by (5.17) and a sum-of-squares error function of the form (5.19). Derive expressions for elements of the Hessian matrix given by

$$H_{rs} = \frac{\partial^2 E}{\partial w_r \partial w_s} \quad (5.70)$$

where w_r and w_s are any two parameters in the network. Hint: the results can conveniently be set out as six equations, one for each possible pair of weight types (basis function centres, basis function widths, or second-layer weights).

5.3 ($\star\star$) Consider the functional derivative (Appendix D) of the regularization functional given by (5.29), with respect to the function $y(\mathbf{x})$. By using successive integration by parts, and making use of the identities

$$\nabla(ab) = a\nabla b + b\nabla a \tag{5.71}$$

$$\nabla \cdot (a\nabla b) = a\nabla^2 b + \nabla b \cdot \nabla a \tag{5.72}$$

show that the operator $\widehat{P}P$ is given by

$$\widehat{P}Py = \sum_{l=0}^{\infty} \frac{\sigma^{2l}}{l!2^l}(-1)^l(\nabla^2)^l y. \tag{5.73}$$

It should be assumed that 'boundary' terms arising from the integration by parts can be neglected. Now find the Green's function $G(\|\mathbf{x} - \mathbf{x}'\|)$ of this operator, defined by (5.24), as follows. First introduce the multidimensional Fourier transform of G, in the form

$$G(\|\mathbf{x} - \mathbf{x}'\|) = \int \widetilde{G}(\mathbf{s}) \exp\left\{-i\mathbf{s}^{\mathrm{T}}(\mathbf{x} - \mathbf{x}')\right\} d\mathbf{s}. \tag{5.74}$$

By substituting (5.74) into (5.73), and using the following form for the Fourier transform of the delta function

$$\delta(\mathbf{x} - \mathbf{x}') = \frac{1}{(2\pi)^d} \int \exp\left\{-i\mathbf{s}^{\mathrm{T}}(\mathbf{x} - \mathbf{x}')\right\} d\mathbf{s} \tag{5.75}$$

where d is the dimensionality of $\mathbf{x}$ and $\mathbf{s}$, show that the Fourier transform of the Green's function is given by

$$\widetilde{G}(\mathbf{s}) = \exp\left\{-\frac{\sigma^2}{2}\|\mathbf{s}\|^2\right\}. \tag{5.76}$$

Now substitute this result into (5.74) and, by using the results given in Appendix B, show that the Green's function is given by

$$G(\|\mathbf{x} - \mathbf{x}'\|) = \frac{1}{(2\pi\sigma^2)^{d/2}} \exp\left\{-\frac{1}{2\sigma^2}\|\mathbf{x} - \mathbf{x}'\|^2\right\}. \tag{5.77}$$

5.4 ($\star$) Consider general Gaussian basis functions of the form (5.16) and suppose that all of the basis functions in the network share a common covariance matrix $\mathbf{\Sigma}$. Show that the mapping represented by such a network is equivalent to that of a network of spherical Gaussian basis functions of the form (5.15), with a common variance parameter $\sigma^2 = 1$, provided the input vector $\mathbf{x}$ is first transformed by an appropriate linear transformation. By making use of the results of Appendix A, find expressions relating the

transformed input vector $\widetilde{\mathbf{x}}$ and transformed basis function centres $\widetilde{\boldsymbol{\mu}}_j$ to the corresponding original vectors $\mathbf{x}$ and $\boldsymbol{\mu}_j$.

5.5 ($\star$) In a multi-layer perceptron a hidden unit has a constant activation for input vectors which lie on a hyperplanar surface in input space given by $\mathbf{w}^{\mathrm{T}}\mathbf{x} + w_0 = \text{const.}$, while for a radial basis function network, with basis functions given by (5.15), a hidden unit has constant activation on a hyperspherical surface defined by $\|\mathbf{x} - \boldsymbol{\mu}\|^2 = \text{const.}$ Show that, for suitable choices of the parameters, these surfaces coincide if the input vectors are normalized to unit length, so that $\|\mathbf{x}\| = 1$. Illustrate this equivalence geometrically for vectors in a three-dimensional input space.

5.6 ($\star\star\star$) Write a numerical implementation of the K-means clustering algorithm described in Section 5.9.3 using both the batch and on-line versions. Illustrate the operation of the algorithm by generating data sets in two dimensions from a mixture of Gaussian distributions, and plotting the data points together with the trajectories of the estimated means during the course of the algorithm. Investigate how the results depend on the value of K in relation to the number of Gaussian distributions, and how they depend on the variances of the distributions in relation to their separation. Study the performance of the on-line version of the algorithm for different values of the learning rate parameter η in (5.60), and compare the algorithm with the batch version.

5.7 ($\star\star\star$) Implement a radial basis function network for one input variable, one output variable and Gaussian basis functions having a common variance parameter σ^2. Generate a set of data by sampling the function $h(x) = 0.5 + 0.4\sin(2\pi x)$ with added Gaussian noise, and with x values taken randomly from a uniform distribution in the interval $(0, 1)$. Set the basis function centres to a random subset of the x values, and use singular value decomposition (Press *et al.*, 1992) to find the network weights which minimize the sum-of-squares error function. Investigate the dependence of the network function on the number of basis function centres and on the value of the variance parameter. Plot graphs of the form shown in Figure 5.3 to illustrate the results.

5.8 ($\star\star\star$) Write down an analytic expression for the regularized matrix $\mathbf{M}$ in (5.32) for the case of Gaussian basis functions given by (5.15). Extend the software implementation of the previous exercise to include this form of regularization. Consider the case in which the number of basis functions equals the number of data points and in which σ is equal to roughly twice the average separation of the input values. Investigate the effect of using different values for the regularization coefficient λ, and show that, if the value of λ is either too small or too large, then the resulting network mapping gives a poor approximation to the function $h(x)$ from which the data was generated.