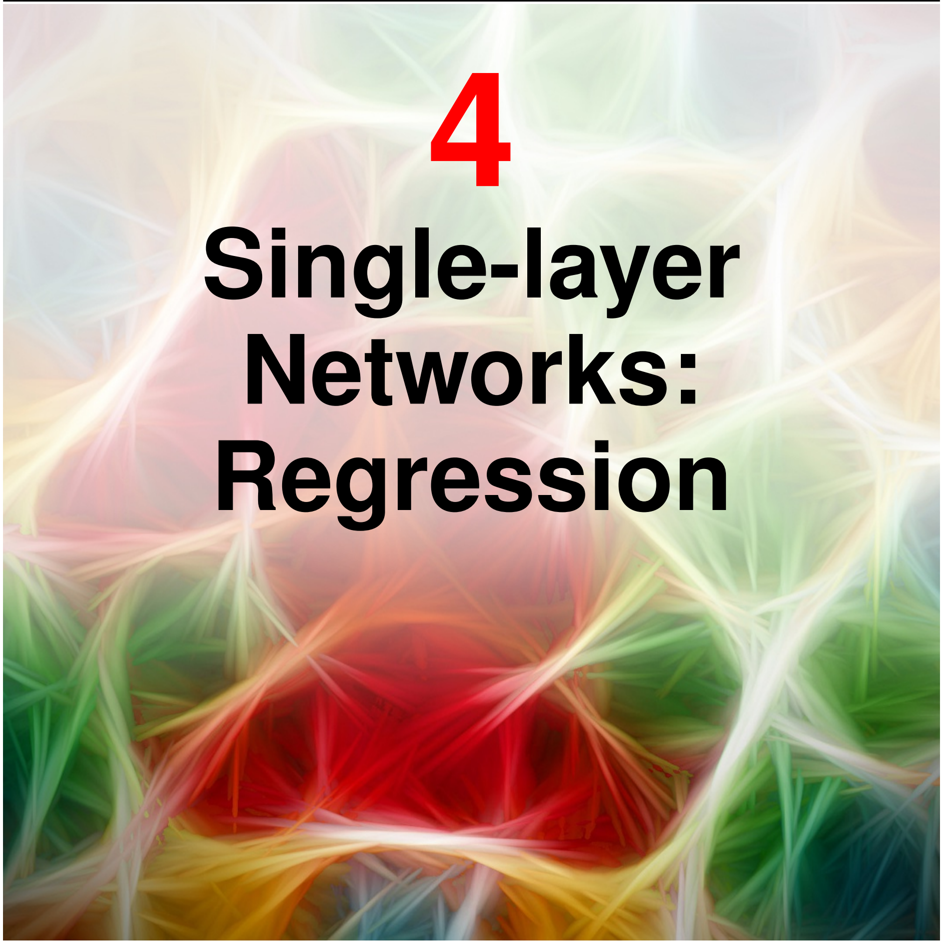

4 Single-layer Networks: Regression

Section 1.2

In this chapter we discuss some of the basic ideas behind neural networks using the framework of linear regression, which we encountered briefly in the context of polynomial curve fitting. We will see that a linear regression model corresponds to a simple form of neural network having a single layer of learnable parameters. Although single-layer networks have very limited practical applicability, they have simple analytical properties and provide an excellent framework for introducing many of the core concepts that will lay a foundation for our discussion of deep neural networks in later chapters.

4.1. Linear Regression

The goal of regression is to predict the value of one or more continuous *target* variables t given the value of a D-dimensional vector $\mathbf{x}$ of *input* variables. Typically we are given a training data set comprising N observations $\{\mathbf{x}_n\}$, where $n = 1, \ldots, N$, together with corresponding target values $\{t_n\}$, and the goal is to predict the value of t for a new value of $\mathbf{x}$. To do this, we formulate a function $y(\mathbf{x}, \mathbf{w})$ whose values for new inputs $\mathbf{x}$ constitute the predictions for the corresponding values of t, and where $\mathbf{w}$ represents a vector of parameters that can be learned from the training data.

The simplest model for regression is one that involves a linear combination of the input variables:

$$y(\mathbf{x}, \mathbf{w}) = w_0 + w_1 x_1 + \ldots + w_D x_D \tag{4.1}$$

where $\mathbf{x} = (x_1, \ldots, x_D)^{\mathrm{T}}$. The term *linear regression* sometimes refers specifically to this form of model. The key property of this model is that it is a linear function of the parameters $w_0, \ldots, w_D$. It is also, however, a linear function of the input variables x_i, and this imposes significant limitations on the model.

4.1.1 Basis functions

We can extend the class of models defined by (4.1) by considering linear combinations of fixed nonlinear functions of the input variables, of the form

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x}) \tag{4.2}$$

where $\phi_j(\mathbf{x})$ are known as *basis functions*. By denoting the maximum value of the index j by $M - 1$, the total number of parameters in this model will be M.

The parameter w_0 allows for any fixed offset in the data and is sometimes called a *bias* parameter (not to be confused with bias in a statistical sense). It is often convenient to define an additional dummy basis function $\phi_0(\mathbf{x})$ whose value is fixed at $\phi_0(\mathbf{x}) = 1$ so that (4.2) becomes

Section 4.3

$$y(\mathbf{x}, \mathbf{w}) = \sum_{j=0}^{M-1} w_j \phi_j(\mathbf{x}) = \mathbf{w}^{\mathrm{T}} \boldsymbol{\phi}(\mathbf{x}) \tag{4.3}$$

where $\mathbf{w} = (w_0, \ldots, w_{M-1})^{\mathrm{T}}$ and $\boldsymbol{\phi} = (\phi_0, \ldots, \phi_{M-1})^{\mathrm{T}}$. We can represent the model (4.3) using a neural network diagram, as shown in Figure 4.1.

By using nonlinear basis functions, we allow the function $y(\mathbf{x}, \mathbf{w})$ to be a nonlinear function of the input vector $\mathbf{x}$. Functions of the form (4.2) are called linear models, however, because they are linear in $\mathbf{w}$. It is this linearity in the parameters that will greatly simplify the analysis of this class of models. However, it also leads to some significant limitations.

Section 6.1

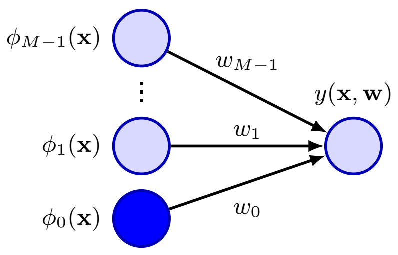

Figure 4.1 The linear regression model (4.3) can be expressed as a simple neural network diagram involving a single layer of parameters. Here each basis function $\phi_j(\mathbf{x})$ is represented by an input node, with the solid node representing the 'bias' basis function ϕ_0, and the function $y(\mathbf{x}, \mathbf{w})$ is represented by an output node. Each of the parameters w_j is shown by a line connecting the corresponding basis function to the output.

Before the advent of deep learning it was common practice in machine learning to use some form of fixed pre-processing of the input variables $\mathbf{x}$, also known as *feature extraction*, expressed in terms of a set of basis functions $\{\phi_j(\mathbf{x})\}$. The goal was to choose a sufficiently powerful set of basis functions that the resulting learning task could be solved using a simple network model. Unfortunately, it is very difficult to hand-craft suitable basis functions for anything but the simplest applications. Deep learning avoids this problem by learning the required nonlinear transformations of the data from the data set itself.

Chapter 1

We have already encountered an example of a regression problem when we discussed curve fitting using polynomials. The polynomial function (1.1) can be expressed in the form (4.3) if we consider a single input variable x and if we choose basis functions defined by $\phi_j(x) = x^j$. There are many other possible choices for the basis functions, for example

$$\phi_j(x) = \exp\left\{-\frac{(x-\mu_j)^2}{2s^2}\right\} \tag{4.4}$$

where the μ_j govern the locations of the basis functions in input space, and the parameter s governs their spatial scale. These are usually referred to as 'Gaussian' basis functions, although it should be noted that they are not required to have a probabilistic interpretation. In particular the normalization coefficient is unimportant because these basis functions will be multiplied by learnable parameters w_j.

Another possibility is the sigmoidal basis function of the form

$$\phi_j(x) = \sigma\left(\frac{x-\mu_j}{s}\right) \tag{4.5}$$

where $\sigma(a)$ is the logistic sigmoid function defined by

$$\sigma(a) = \frac{1}{1+\exp(-a)}. \tag{4.6}$$

Exercise 4.3

Equivalently, we can use the $\tanh$ function because this is related to the logistic sigmoid by $\tanh(a) = 2\sigma(2a) - 1$, and so a general linear combination of logistic sigmoid functions is equivalent to a general linear combination of $\tanh$ functions in the sense that they can represent the same class of input–output functions. These various choices of basis function are illustrated in Figure 4.2.

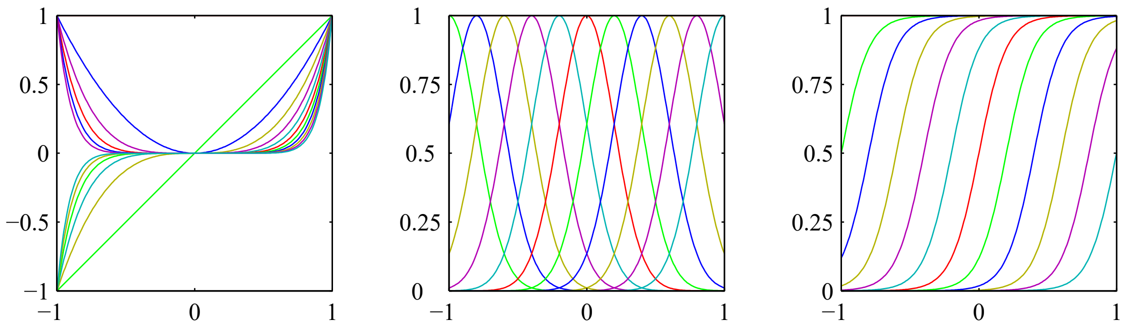

Figure 4.2 Examples of basis functions, showing polynomials on the left, Gaussians of the form (4.4) in the centre, and sigmoidal basis functions of the form (4.5) on the right.

Yet another possible choice of basis function is the Fourier basis, which leads to an expansion in sinusoidal functions. Each basis function represents a specific frequency and has infinite spatial extent. By contrast, basis functions that are localized to finite regions of input space necessarily comprise a spectrum of different spatial frequencies. In signal processing applications, it is often of interest to consider basis functions that are localized in both space and frequency, leading to a class of functions known as *wavelets* (Ogden, 1997; Mallat, 1999; Vidakovic, 1999). These are also defined to be mutually orthogonal, to simplify their application. Wavelets are most applicable when the input values live on a regular lattice, such as the successive time points in a temporal sequence or the pixels in an image.

Most of the discussion in this chapter, however, is independent of the choice of basis function set, and so we will not specify the particular form of the basis functions, except for numerical illustration. Furthermore, to keep the notation simple, we will focus on the case of a single target variable t, although we will briefly outline the modifications needed to deal with multiple target variables. *Section 4.1.7*

4.1.2 Likelihood function

Section 1.2

We solved the problem of fitting a polynomial function to data by minimizing a sum-of-squares error function, and we also showed that this error function could be motivated as the maximum likelihood solution under an assumed Gaussian noise model. We now return to this discussion and consider the least-squares approach, and its relation to maximum likelihood, in more detail.

As before, we assume that the target variable t is given by a deterministic function $y(\mathbf{x}, \mathbf{w})$ with additive Gaussian noise so that

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon \tag{4.7}$$

where ϵ is a zero-mean Gaussian random variable with variance σ^2. Thus, we can write

$$p(t|\mathbf{x}, \mathbf{w}, \sigma^2) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \sigma^2). \tag{4.8}$$

Now consider a data set of inputs $\mathbf{X} = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with corresponding target values $t_1, \ldots, t_N$. We group the target variables $\{t_n\}$ into a column vector that we denote by t where the typeface is chosen to distinguish it from a single observation of a multivariate target, which would be denoted $\mathbf{t}$. Making the assumption that these data points are drawn independently from the distribution (4.8), we obtain an expression for the likelihood function, which is a function of the adjustable parameters $\mathbf{w}$ and σ^2:

$$p(\mathsf{t}|\mathbf{X}, \mathbf{w}, \sigma^2) = \prod_{n=1}^{N} \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \sigma^2) \tag{4.9}$$

where we have used (4.3). Taking the logarithm of the likelihood function and making use of the standard form (2.49) for the univariate Gaussian, we have

$$\begin{aligned}\ln p(\mathsf{t}|\mathbf{X}, \mathbf{w}, \sigma^2) &= \sum_{n=1}^{N} \ln \mathcal{N}(t_n|\mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \sigma^2) \\ &= -\frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln(2\pi) - \frac{1}{\sigma^2}E_D(\mathbf{w}) \end{aligned} \tag{4.10}$$

where the sum-of-squares error function is defined by

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2. \tag{4.11}$$

Section 2.3.4

The first two terms in (4.10) can be treated as constants when determining $\mathbf{w}$ because they are independent of $\mathbf{w}$. Therefore, as we saw previously, maximizing the likelihood function under a Gaussian noise distribution is equivalent to minimizing the sum-of-squares error function (4.11).

4.1.3 Maximum likelihood

Having written down the likelihood function, we can use maximum likelihood to determine $\mathbf{w}$ and σ^2. Consider first the maximization with respect to $\mathbf{w}$. The gradient of the log likelihood function (4.10) with respect to $\mathbf{w}$ takes the form

$$\nabla_{\mathbf{w}} \ln p(\mathsf{t}|\mathbf{X}, \mathbf{w}, \sigma^2) = \frac{1}{\sigma^2}\sum_{n=1}^{N}\left\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\right\}\boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}. \tag{4.12}$$

Setting this gradient to zero gives

$$0 = \sum_{n=1}^{N} t_n\boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}} - \mathbf{w}^{\mathrm{T}}\left(\sum_{n=1}^{N}\boldsymbol{\phi}(\mathbf{x}_n)\boldsymbol{\phi}(\mathbf{x}_n)^{\mathrm{T}}\right). \tag{4.13}$$

Solving for $\mathbf{w}$ we obtain

$$\mathbf{w}_{\mathrm{ML}} = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathsf{t}, \tag{4.14}$$

which are known as the *normal equations* for the least-squares problem. Here $\boldsymbol{\Phi}$ is an $N \times M$ matrix, called the *design matrix*, whose elements are given by $\Phi_{nj} = \phi_j(\mathbf{x}_n)$, so that

$$\boldsymbol{\Phi} = \begin{pmatrix} \phi_0(\mathbf{x}_1) & \phi_1(\mathbf{x}_1) & \cdots & \phi_{M-1}(\mathbf{x}_1) \\ \phi_0(\mathbf{x}_2) & \phi_1(\mathbf{x}_2) & \cdots & \phi_{M-1}(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ \phi_0(\mathbf{x}_N) & \phi_1(\mathbf{x}_N) & \cdots & \phi_{M-1}(\mathbf{x}_N) \end{pmatrix}. \tag{4.15}$$

The quantity

$$\boldsymbol{\Phi}^{\dagger} \equiv \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1} \boldsymbol{\Phi}^{\mathrm{T}} \tag{4.16}$$

is known as the *Moore–Penrose pseudo-inverse* of the matrix $\boldsymbol{\Phi}$ (Rao and Mitra, 1971; Golub and Van Loan, 1996). It can be regarded as a generalization of the notion of a matrix inverse to non-square matrices. Indeed, if $\boldsymbol{\Phi}$ is square and invertible, then using the property $(\mathbf{AB})^{-1} = \mathbf{B}^{-1}\mathbf{A}^{-1}$ we see that $\boldsymbol{\Phi}^{\dagger} \equiv \boldsymbol{\Phi}^{-1}$.

At this point, we can gain some insight into the role of the bias parameter w_0. If we make the bias parameter explicit, then the error function (4.11) becomes

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - w_0 - \sum_{j=1}^{M-1} w_j\phi_j(\mathbf{x}_n)\}^2. \tag{4.17}$$

Setting the derivative with respect to w_0 equal to zero and solving for w_0, we obtain

$$w_0 = \bar{t} - \sum_{j=1}^{M-1} w_j\overline{\phi_j} \tag{4.18}$$

where we have defined

$$\bar{t} = \frac{1}{N}\sum_{n=1}^{N} t_n, \qquad \overline{\phi_j} = \frac{1}{N}\sum_{n=1}^{N}\phi_j(\mathbf{x}_n). \tag{4.19}$$

Thus, the bias w_0 compensates for the difference between the averages (over the training set) of the target values and the weighted sum of the averages of the basis function values.

We can also maximize the log likelihood function (4.10) with respect to the variance σ^2, giving

$$\sigma^2_{\mathrm{ML}} = \frac{1}{N}\sum_{n=1}^{N}\{t_n - \mathbf{w}_{\mathrm{ML}}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2, \tag{4.20}$$

and so we see that the maximum likelihood value of the variance parameter is given by the residual variance of the target values around the regression function.

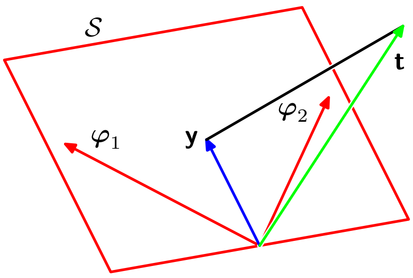

Figure 4.3 Geometrical interpretation of the least-squares solution in an N-dimensional space whose axes are the values of $t_1, \ldots, t_N$. The least-squares regression function is obtained by finding the orthogonal projection of the data vector $\mathbf{t}$ onto the subspace spanned by the basis functions $\phi_j(\mathbf{x})$ in which each basis function is viewed as a vector $\boldsymbol{\varphi}_j$ of length N with elements $\phi_j(\mathbf{x}_n)$.

4.1.4 Geometry of least squares

At this point, it is instructive to consider the geometrical interpretation of the least-squares solution. To do this, we consider an N-dimensional space whose axes are given by the t_n, so that $\mathbf{t} = (t_1, \ldots, t_N)^{\mathrm{T}}$ is a vector in this space. Each basis function $\phi_j(\mathbf{x}_n)$, evaluated at the N data points, can also be represented as a vector in the same space, denoted by $\boldsymbol{\varphi}_j$, as illustrated in Figure 4.3. Note that $\boldsymbol{\varphi}_j$ corresponds to the jth column of $\boldsymbol{\Phi}$, whereas $\boldsymbol{\phi}(\mathbf{x}_n)$ corresponds to the transpose of the nth row of $\boldsymbol{\Phi}$. If the number M of basis functions is smaller than the number N of data points, then the M vectors $\phi_j(\mathbf{x}_n)$ will span a linear subspace $\mathcal{S}$ of dimensionality M. We define $\mathbf{y}$ to be an N-dimensional vector whose nth element is given by $y(\mathbf{x}_n, \mathbf{w})$, where $n = 1, \ldots, N$. Because $\mathbf{y}$ is an arbitrary linear combination of the vectors $\boldsymbol{\varphi}_j$, it can live anywhere in the M-dimensional subspace. The sum-of-squares error (4.11) is then equal (up to a factor of $1/2$) to the squared Euclidean distance between $\mathbf{y}$ and $\mathbf{t}$. Thus, the least-squares solution for $\mathbf{w}$ corresponds to that choice of $\mathbf{y}$ that lies in subspace $\mathcal{S}$ and is closest to $\mathbf{t}$. Intuitively, from Figure 4.3, we anticipate that this solution corresponds to the orthogonal projection of $\mathbf{t}$ onto the subspace $\mathcal{S}$. This is indeed the case, as can easily be verified by noting that the solution for $\mathbf{y}$ is given by $\boldsymbol{\Phi}\mathbf{w}_{\mathrm{ML}}$ and then confirming that this takes the form of an orthogonal projection. *Exercise 4.4*

In practice, a direct solution of the normal equations can lead to numerical difficulties when $\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}$ is close to singular. In particular, when two or more of the basis vectors $\boldsymbol{\varphi}_j$ are co-linear, or nearly so, the resulting parameter values can have large magnitudes. Such near degeneracies will not be uncommon when dealing with real data sets. The resulting numerical difficulties can be addressed using the technique of *singular value decomposition*, or *SVD* (Deisenroth, Faisal, and Ong, 2020). Note that the addition of a regularization term ensures that the matrix is non-singular, even in the presence of degeneracies.

4.1.5 Sequential learning

The maximum likelihood solution (4.14) involves processing the entire training set in one go and is known as a *batch* method. This can become computationally costly for large data sets. If the data set is sufficiently large, it may be worthwhile to use *sequential* algorithms, also known as *online* algorithms, in which the data points are considered one at a time and the model parameters updated after each such presentation. Sequential learning is also appropriate for real-time applications in which the data observations arrive in a continuous stream and predictions must be made before all the data points are seen.

Chapter 7

We can obtain a sequential learning algorithm by applying the technique of *stochastic gradient descent*, also known as *sequential gradient descent*, as follows. If the error function comprises a sum over data points $E = \sum_n E_n$, then after presentation of data point n, the stochastic gradient descent algorithm updates the parameter vector $\mathbf{w}$ using

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n \tag{4.21}$$

where τ denotes the iteration number, and η is a suitably chosen learning rate parameter. The value of $\mathbf{w}$ is initialized to some starting vector $\mathbf{w}^{(0)}$. For the sum-of-squares error function (4.11), this gives

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \eta(t_n - \mathbf{w}^{(\tau)\mathrm{T}}\boldsymbol{\phi}_n)\boldsymbol{\phi}_n \tag{4.22}$$

where $\boldsymbol{\phi}_n = \boldsymbol{\phi}(\mathbf{x}_n)$. This is known as the *least-mean-squares* or the *LMS algorithm*.

4.1.6 Regularized least squares

Section 1.2

We have previously introduced the idea of adding a regularization term to an error function to control over-fitting, so that the total error function to be minimized takes the form

$$E_D(\mathbf{w}) + \lambda E_W(\mathbf{w}) \tag{4.23}$$

where λ is the regularization coefficient that controls the relative importance of the data-dependent error $E_D(\mathbf{w})$ and the regularization term $E_W(\mathbf{w})$. One of the simplest forms of regularizer is given by the sum of the squares of the weight vector elements:

$$E_W(\mathbf{w}) = \frac{1}{2}\sum_j w_j^2 = \frac{1}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}. \tag{4.24}$$

If we also consider the sum-of-squares error function given by

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2, \tag{4.25}$$

then the total error function becomes

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\mathbf{w}^{\mathrm{T}}\mathbf{w}. \tag{4.26}$$

In statistics, this regularizer provides an example of a *parameter shrinkage* method because it shrinks parameter values towards zero. It has the advantage that the error function remains a quadratic function of $\mathbf{w}$, and so its exact minimizer can be found in closed form. Specifically, setting the gradient of (4.26) with respect to $\mathbf{w}$ to zero and solving for $\mathbf{w}$ as before, we obtain

Exercise 4.6

$$\mathbf{w} = \left(\lambda\mathbf{I} + \boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}. \tag{4.27}$$

This represents a simple extension of the least-squares solution (4.14).

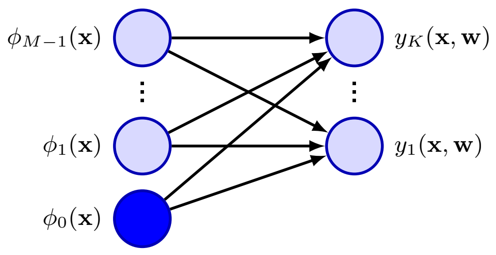

Figure 4.4 Representation of a linear regression model as a neural network having a single layer of connections. Each basis function is represented by a node, with the solid node representing the 'bias' basis function ϕ_0. Likewise each output $y_1, \ldots, y_K$ is represented by a node. The links between the nodes represent the corresponding weight and bias parameters.

4.1.7 Multiple outputs

So far, we have considered situations with a single target variable t. In some applications, we may wish to predict $K > 1$ target variables, which we denote collectively by the target vector $\mathbf{t} = (t_1, \ldots, t_K)^{\mathrm{T}}$. This could be done by introducing a different set of basis functions for each component of $\mathbf{t}$, leading to multiple, independent regression problems. However, a more common approach is to use the same set of basis functions to model all of the components the target vector so that

$$\mathbf{y}(\mathbf{x}, \mathbf{w}) = \mathbf{W}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}) \tag{4.28}$$

where $\mathbf{y}$ is a K-dimensional column vector, $\mathbf{W}$ is an $M \times K$ matrix of parameters, and $\boldsymbol{\phi}(\mathbf{x})$ is an M-dimensional column vector with elements $\phi_j(\mathbf{x})$ with $\phi_0(\mathbf{x}) = 1$ as before. Again, this can be represented as a neural network having a single layer of parameters, as shown in Figure 4.4.

Suppose we take the conditional distribution of the target vector to be an isotropic Gaussian of the form

$$p(\mathbf{t}|\mathbf{x}, \mathbf{W}, \sigma^2) = \mathcal{N}(\mathbf{t}|\mathbf{W}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}), \sigma^2\mathbf{I}). \tag{4.29}$$

If we have a set of observations $\mathbf{t}_1, \ldots, \mathbf{t}_N$, we can combine these into a matrix $\mathbf{T}$ of size $N \times K$ such that the nth row is given by $\mathbf{t}_n^{\mathrm{T}}$. Similarly, we can combine the input vectors $\mathbf{x}_1, \ldots, \mathbf{x}_N$ into a matrix $\mathbf{X}$. The log likelihood function is then given by

$$\begin{aligned}\ln p(\mathbf{T}|\mathbf{X}, \mathbf{W}, \sigma^2) &= \sum_{n=1}^{N} \ln \mathcal{N}(\mathbf{t}_n|\mathbf{W}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n), \sigma^2\mathbf{I}) \\ &= -\frac{NK}{2}\ln\left(2\pi\sigma^2\right) - \frac{1}{2\sigma^2}\sum_{n=1}^{N}\left\|\mathbf{t}_n - \mathbf{W}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\right\|^2. \end{aligned} \tag{4.30}$$

As before, we can maximize this function with respect to $\mathbf{W}$, giving

$$\mathbf{W}_{\mathrm{ML}} = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{T} \tag{4.31}$$

where we have combined the input feature vectors $\boldsymbol{\phi}(\mathbf{x}_1), \ldots, \boldsymbol{\phi}(\mathbf{x}_N)$ into a matrix $\boldsymbol{\Phi}$. If we examine this result for each target variable t_k, we have

$$\mathbf{w}_k = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}_k = \boldsymbol{\Phi}^{\dagger}\mathbf{t}_k \tag{4.32}$$

where $\mathbf{t}_k$ is an N-dimensional column vector with components t_{nk} for $n = 1, \ldots N$. Thus, the solution to the regression problem decouples between the different target variables, and we need compute only a single pseudo-inverse matrix $\boldsymbol{\Phi}^{\dagger}$, which is shared by all the vectors $\mathbf{w}_k$.

Exercise 4.7

The extension to general Gaussian noise distributions having arbitrary covariance matrices is straightforward. Again, this leads to a decoupling into K independent regression problems. This result is unsurprising because the parameters $\mathbf{W}$ define only the mean of the Gaussian noise distribution, and we know that the maximum likelihood solution for the mean of a multivariate Gaussian is independent of the covariance. From now on, we will therefore consider a single target variable t for simplicity.

Section 3.2.7

4.2. Decision theory

We have formulated the regression task as one of modelling a conditional probability distribution $p(t|\mathbf{x})$, and we have chosen a specific form for the conditional probability, namely a Gaussian (4.8) with an $\mathbf{x}$-dependent mean $y(\mathbf{x}, \mathbf{w})$ governed by parameters $\mathbf{w}$ and with variance given by the parameter σ^2. Both $\mathbf{w}$ and σ^2 can be learned from data using maximum likelihood. The result is a *predictive distribution* given by

$$p(t|\mathbf{x}, \mathbf{w}_{\mathrm{ML}}, \sigma^2_{\mathrm{ML}}) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}_{\mathrm{ML}}), \sigma^2_{\mathrm{ML}}). \tag{4.33}$$

The predictive distribution expresses our uncertainty over the value of t for some new input $\mathbf{x}$. However, for many practical applications we need to predict a specific value for t rather than returning an entire distribution, particularly where we must take a specific action. For example, if our goal is to determine the optimal level of radiation to use for treating a tumour and our model predicts a probability distribution over radiation dose, then we must use that distribution to decide the specific dose to be administered. Our task therefore breaks down into two stages. In the first stage, called the *inference* stage, we use the training data to determine a predictive distribution $p(t|\mathbf{x})$. In the second stage, known as the *decision* stage, we use this predictive distribution to determine a specific value $f(\mathbf{x})$, which will be dependent on the input vector $\mathbf{x}$, that is optimal according to some criterion. We can do this by minimizing a *loss function* that depends on both the predictive distribution $p(t|\mathbf{x})$ and on f.

Intuitively we might choose the mean of the conditional distribution, so that we would use $f(\mathbf{x}) = y(\mathbf{x}, \mathbf{w}_{\mathrm{ML}})$. In some cases this intuition will be correct, but in other situations it can give very poor results. It is therefore useful to formalize this so that we can understand when it applies and under what assumptions, and the framework for doing this is called *decision theory*.

Suppose that we choose a value $f(\mathbf{x})$ for our prediction when the true value is t. In doing so, we incur some form of penalty or cost. This is determined by a *loss*, which we denote $L(t, f(\mathbf{x}))$. Of course, we do not know the true value of t, so instead of minimizing L itself, we minimize the average, or expected, loss which is

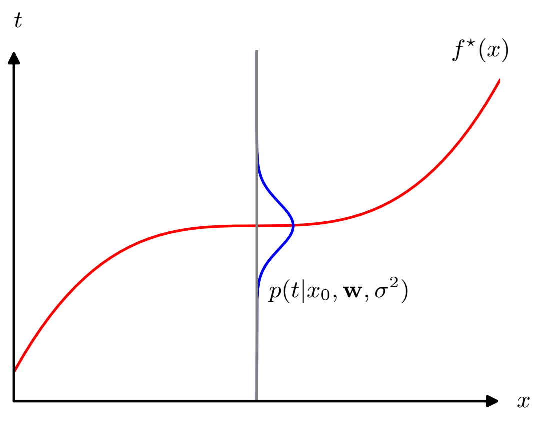

Figure 4.5 The regression function $f^\star(x)$, which minimizes the expected squared loss, is given by the mean of the conditional distribution $p(t|x)$.

given by

$$\mathbb{E}[L] = \iint L(t, f(\mathbf{x}))p(\mathbf{x}, t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t \tag{4.34}$$

where we are averaging over the distribution of both input and target variables, weighted by their joint distribution $p(\mathbf{x}, t)$. A common choice of loss function in regression problems is the squared loss given by $L(t, f(\mathbf{x})) = \{f(\mathbf{x}) - t\}^2$. In this case, the expected loss can be written

$$\mathbb{E}[L] = \iint \{f(\mathbf{x}) - t\}^2 p(\mathbf{x}, t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t. \tag{4.35}$$

It is important not to confuse the squared-loss function with the sum-of-squares error function introduced earlier. The error function is used to set the parameters during training in order to determine the conditional probability distribution $p(t|\mathbf{x})$, whereas the loss function governs how the conditional distribution is used to arrive at a predictive function $f(\mathbf{x})$ specifying a prediction for each value of $\mathbf{x}$.

Our goal is to choose $f(\mathbf{x})$ so as to minimize $\mathbb{E}[L]$. If we assume a completely flexible function $f(\mathbf{x})$, we can do this formally using the calculus of variations to give

Appendix B

$$\frac{\delta \mathbb{E}[L]}{\delta f(\mathbf{x})} = 2\int \{f(\mathbf{x}) - t\}p(\mathbf{x}, t)\,\mathrm{d}t = 0. \tag{4.36}$$

Solving for $f(\mathbf{x})$ and using the sum and product rules of probability, we obtain

$$f^\star(\mathbf{x}) = \frac{1}{p(\mathbf{x})}\int tp(\mathbf{x}, t)\,\mathrm{d}t = \int tp(t|\mathbf{x})\,\mathrm{d}t = \mathbb{E}_t[t|\mathbf{x}], \tag{4.37}$$

which is the conditional average of t conditioned on $\mathbf{x}$ and is known as the *regression function*. This result is illustrated in Figure 4.5. It can readily be extended to multiple target variables represented by the vector $\mathbf{t}$, in which case the optimal solution is the conditional average $\mathbf{f}^\star(\mathbf{x}) = \mathbb{E}_t[\mathbf{t}|\mathbf{x}]$. For a Gaussian conditional distribution of the

Exercise 4.8

form (4.8), the conditional mean will be simply

$$\mathbb{E}[t|\mathbf{x}] = \int tp(t|\mathbf{x})\,\mathrm{d}t = y(\mathbf{x}, \mathbf{w}). \tag{4.38}$$

The use of calculus of variations to derive (4.37) implies that we are optimizing over all possible functions $f(\mathbf{x})$. Although any parametric model that we can implement in practice is limited in the range of functions that it can represent, the framework of deep neural networks, discussed extensively in later chapters, provides a highly flexible class of functions that, for many practical purposes, can approximate any desired function to high accuracy.

We can derive this result in a slightly different way, which will also shed light on the nature of the regression problem. Armed with the knowledge that the optimal solution is the conditional expectation, we can expand the square term as follows

$$\begin{aligned}\{f(\mathbf{x}) - t\}^2 &= \{f(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}] + \mathbb{E}[t|\mathbf{x}] - t\}^2 \\ &= \{f(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2 + 2\{f(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}\{\mathbb{E}[t|\mathbf{x}] - t\} + \{\mathbb{E}[t|\mathbf{x}] - t\}^2\end{aligned}$$

where, to keep the notation uncluttered, we use $\mathbb{E}[t|\mathbf{x}]$ to denote $\mathbb{E}_t[t|\mathbf{x}]$. Substituting into the loss function (4.35) and performing the integral over t, we see that the cross-term vanishes and we obtain an expression for the loss function in the form

$$\mathbb{E}[L] = \int \{f(\mathbf{x}) - \mathbb{E}[t|\mathbf{x}]\}^2\, p(\mathbf{x})\,\mathrm{d}\mathbf{x} + \int \mathrm{var}\,[t|\mathbf{x}]\, p(\mathbf{x})\,\mathrm{d}\mathbf{x}. \tag{4.39}$$

The function $f(\mathbf{x})$ we seek to determine appears only in the first term, which will be minimized when $f(\mathbf{x})$ is equal to $\mathbb{E}[t|\mathbf{x}]$, in which case this term will vanish. This is simply the result that we derived previously, and shows that the optimal least-squares predictor is given by the conditional mean. The second term is the variance of the distribution of t, averaged over $\mathbf{x}$, and represents the intrinsic variability of the target data and can be regarded as noise. Because it is independent of $f(\mathbf{x})$, it represents the irreducible minimum value of the loss function.

The squared loss is not the only possible choice of loss function for regression. Here we consider briefly one simple generalization of the squared loss, called the *Minkowski* loss, whose expectation is given by

$$\mathbb{E}[L_q] = \iint |f(\mathbf{x}) - t|^q p(\mathbf{x}, t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t, \tag{4.40}$$

which reduces to the expected squared loss for $q = 2$. The function $|f - t|^q$ is plotted against $f - t$ for various values of q in Figure 4.6. The minimum of $\mathbb{E}[L_q]$ is given by the conditional mean for $q = 2$, the conditional median for $q = 1$, and the conditional mode for $q \to 0$. *Exercise 4.12*

Note that the Gaussian noise assumption implies that the conditional distribution of t given $\mathbf{x}$ is unimodal, which may be inappropriate for some applications. In this case a squared loss can lead to very poor results and we need to develop more sophisticated approaches. For example, we can extend this model by using mixtures

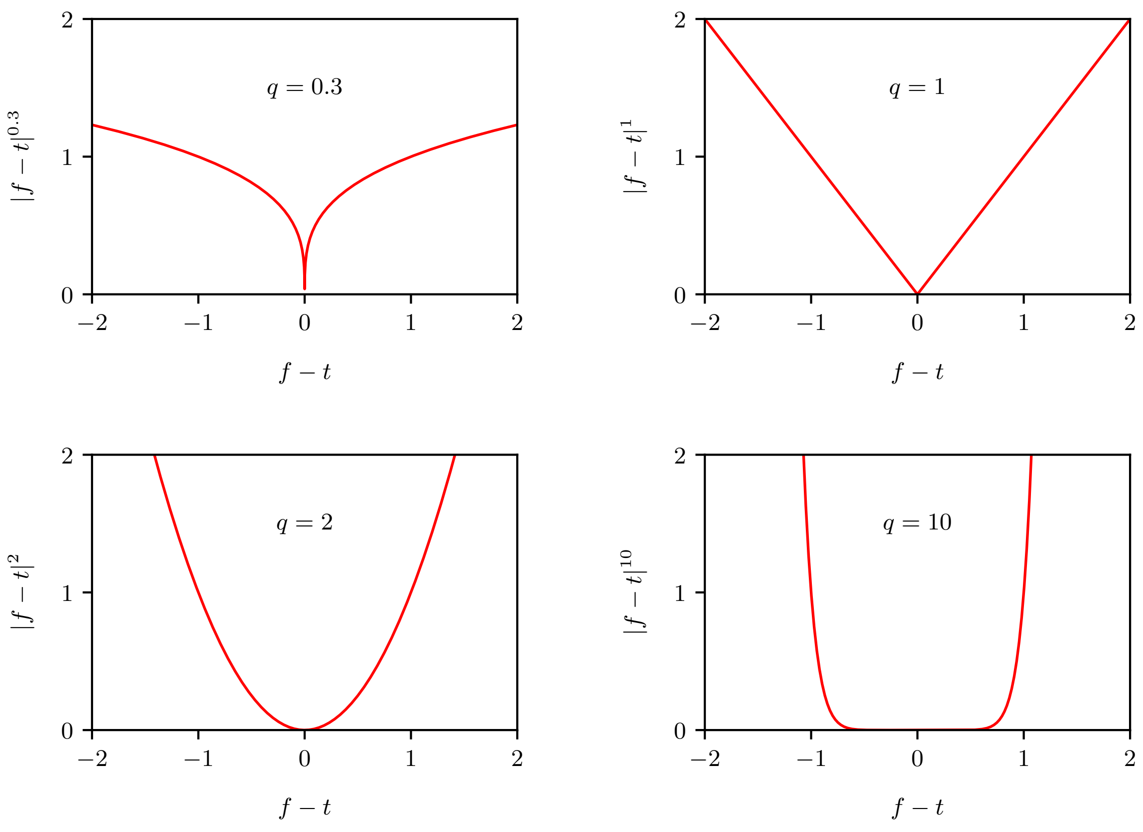

Figure 4.6 Plots of the quantity $L_q = |f - t|^q$ for various values of q.

Section 6.5

Section 5.2

of Gaussians to give multimodal conditional distributions, which often arise in the solution of *inverse problems*. Our focus in this section has been on decision theory for regression problems, and in the next chapter we shall develop analogous concepts for classification tasks.

4.3. The Bias–Variance Trade-off

Section 1.2

So far in our discussion of linear models for regression, we have assumed that the form and number of basis functions are both given. We have also seen that the use of maximum likelihood can lead to severe over-fitting if complex models are trained using data sets of limited size. However, limiting the number of basis functions to avoid over-fitting has the side effect of limiting the flexibility of the model to capture interesting and important trends in the data. Although a regularization term can control over-fitting for models with many parameters, this raises the question of how to determine a suitable value for the regularization coefficient λ. Seeking the solution that minimizes the regularized error function with respect to both the weight vector $\mathbf{w}$ and the regularization coefficient λ is clearly not the right approach, since this leads to the unregularized solution with $\lambda = 0$.

It is instructive to consider a frequentist viewpoint of the model complexity issue, known as the *bias–variance* trade-off. Although we will introduce this concept in the context of linear basis function models, where it is easy to illustrate the ideas using simple examples, the discussion has very general applicability. Note, however, that over-fitting is really an unfortunate property of maximum likelihood and does not arise when we marginalize over parameters in a Bayesian setting (Bishop, 2006).

Section 4.2

When we discussed decision theory for regression problems, we considered various loss functions, each of which leads to a corresponding optimal prediction once we are given the conditional distribution $p(t|\mathbf{x})$. A popular choice is the squared-loss function, for which the optimal prediction is given by the conditional expectation, which we denote by $h(\mathbf{x})$ and is given by

$$h(\mathbf{x}) = \mathbb{E}[t|\mathbf{x}] = \int t p(t|\mathbf{x})\,\mathrm{d}t. \tag{4.41}$$

We have also seen that the expected squared loss can be written in the form

$$\mathbb{E}[L] = \int \{f(\mathbf{x}) - h(\mathbf{x})\}^2\, p(\mathbf{x})\,\mathrm{d}\mathbf{x} + \iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x}, t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t. \tag{4.42}$$

Recall that the second term, which is independent of $f(\mathbf{x})$, arises from the intrinsic noise on the data and represents the minimum achievable value of the expected loss. The first term depends on our choice for the function $f(\mathbf{x})$, and we will seek a solution for $f(\mathbf{x})$ that makes this term a minimum. Because it is non-negative, the smallest value that we can hope to achieve for this term is zero. If we had an unlimited supply of data (and unlimited computational resources), we could in principle find the regression function $h(\mathbf{x})$ to any desired degree of accuracy, and this would represent the optimal choice for $f(\mathbf{x})$. However, in practice we have a data set $\mathcal{D}$ containing only a finite number N of data points, and consequently, we cannot know the regression function $h(\mathbf{x})$ exactly.

If we were to model $h(\mathbf{x})$ using a function governed by a parameter vector $\mathbf{w}$, then from a Bayesian perspective, the uncertainty in our model would be expressed through a posterior distribution over $\mathbf{w}$. A frequentist treatment, however, involves making a point estimate of $\mathbf{w}$ based on the data set $\mathcal{D}$ and tries instead to interpret the uncertainty of this estimate through the following thought experiment. Suppose we had a large number of data sets each of size N and each drawn independently from the distribution $p(t, \mathbf{x})$. For any given data set $\mathcal{D}$, we can run our learning algorithm and obtain a prediction function $f(\mathbf{x}; \mathcal{D})$. Different data sets from the ensemble will give different functions and consequently different values of the squared loss. The performance of a particular learning algorithm is then assessed by taking the average over this ensemble of data sets.

Consider the integrand of the first term in (4.42), which for a particular data set $\mathcal{D}$ takes the form

$$\{f(\mathbf{x}; \mathcal{D}) - h(\mathbf{x})\}^2. \tag{4.43}$$

Because this quantity will be dependent on the particular data set $\mathcal{D}$, we take its average over the ensemble of data sets. If we add and subtract the quantity $\mathbb{E}_{\mathcal{D}}[f(\mathbf{x};\mathcal{D})]$ inside the braces, and then expand, we obtain

$$\begin{aligned}&\{f(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[f(\mathbf{x};\mathcal{D})] + \mathbb{E}_{\mathcal{D}}[f(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2 \\ &= \{f(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[f(\mathbf{x};\mathcal{D})]\}^2 + \{\mathbb{E}_{\mathcal{D}}[f(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2 \\ &+ 2\{f(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[f(\mathbf{x};\mathcal{D})]\}\{\mathbb{E}_{\mathcal{D}}[f(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}.\end{aligned} \tag{4.44}$$

We now take the expectation of this expression with respect to $\mathcal{D}$ and note that the final term will vanish, giving

$$\begin{aligned}&\mathbb{E}_{\mathcal{D}}\left[\{f(\mathbf{x};\mathcal{D}) - h(\mathbf{x})\}^2\right] \\ &= \underbrace{\{\mathbb{E}_{\mathcal{D}}[f(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2}_{(\text{bias})^2} + \underbrace{\mathbb{E}_{\mathcal{D}}\left[\{f(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[f(\mathbf{x};\mathcal{D})]\}^2\right]}_{\text{variance}}.\end{aligned} \tag{4.45}$$

We see that the expected squared difference between $f(\mathbf{x};\mathcal{D})$ and the regression function $h(\mathbf{x})$ can be expressed as the sum of two terms. The first term, called the squared *bias*, represents the extent to which the average prediction over all data sets differs from the desired regression function. The second term, called the *variance*, measures the extent to which the solutions for individual data sets vary around their average, and hence, this measures the extent to which the function $f(\mathbf{x};\mathcal{D})$ is sensitive to the particular choice of data set. We will provide some intuition to support these definitions shortly when we consider a simple example.

So far, we have considered a single input value $\mathbf{x}$. If we substitute this expansion back into (4.42), we obtain the following decomposition of the expected squared loss:

$$\text{expected loss} = (\text{bias})^2 + \text{variance} + \text{noise} \tag{4.46}$$

where

$$(\text{bias})^2 = \int \{\mathbb{E}_{\mathcal{D}}[f(\mathbf{x};\mathcal{D})] - h(\mathbf{x})\}^2 p(\mathbf{x})\,\mathrm{d}\mathbf{x} \tag{4.47}$$

$$\text{variance} = \int \mathbb{E}_{\mathcal{D}}\left[\{f(\mathbf{x};\mathcal{D}) - \mathbb{E}_{\mathcal{D}}[f(\mathbf{x};\mathcal{D})]\}^2\right] p(\mathbf{x})\,\mathrm{d}\mathbf{x} \tag{4.48}$$

$$\text{noise} = \iint \{h(\mathbf{x}) - t\}^2 p(\mathbf{x},t)\,\mathrm{d}\mathbf{x}\,\mathrm{d}t \tag{4.49}$$

and the bias and variance terms now refer to integrated quantities.

Our goal is to minimize the expected loss, which we have decomposed into the sum of a (squared) bias, a variance, and a constant noise term. As we will see, there is a trade-off between bias and variance, with very flexible models having low bias and high variance, and relatively rigid models having high bias and low variance. The model with the optimal predictive capability is the one that leads to the best balance between bias and variance. This is illustrated by considering the sinusoidal data set introduced earlier. Here we independently generate 100 data sets, each containing *Section 1.2*

$N = 25$ data points, from the sinusoidal curve $h(x) = \sin(2\pi x)$. The data sets are indexed by $l = 1, \ldots, L$, where $L = 100$. For each data set $\mathcal{D}^{(l)}$, we fit a model with $M = 24$ Gaussian basis functions along with a constant 'bias' basis function to give a total of 25 parameters. By minimizing the regularized error function (4.26), we obtain a prediction function $f^{(l)}(x)$, as shown in Figure 4.7.

The top row corresponds to a large value of the regularization coefficient λ that gives low variance (because the red curves in the left plot look similar) but high bias (because the two curves in the right plot are very different). Conversely on the bottom row, for which λ is small, there is large variance (shown by the high variability between the red curves in the left plot) but low bias (shown by the good fit between the average model fit and the original sinusoidal function). Note that the result of averaging many solutions for the complex model with $M = 25$ is a very good fit to the regression function, which suggests that averaging may be a beneficial procedure. Indeed, a weighted averaging of multiple solutions lies at the heart of a Bayesian approach, although the averaging is with respect to the posterior distribution of parameters, not with respect to multiple data sets.

We can also examine the bias–variance trade-off quantitatively for this example. The average prediction is estimated from

$$\overline{f}(x) = \frac{1}{L}\sum_{l=1}^{L} f^{(l)}(x), \tag{4.50}$$

and the integrated squared bias and integrated variance are then given by

$$(\text{bias})^2 = \frac{1}{N}\sum_{n=1}^{N}\left\{\overline{f}(x_n) - h(x_n)\right\}^2 \tag{4.51}$$

$$\text{variance} = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{L}\sum_{l=1}^{L}\left\{f^{(l)}(x_n) - \overline{f}(x_n)\right\}^2 \tag{4.52}$$

where the integral over x, weighted by the distribution $p(x)$, is approximated by a finite sum over data points drawn from that distribution. These quantities, along with their sum, are plotted as a function of $\ln\lambda$ in Figure 4.8. We see that small values of λ allow the model to become finely tuned to the noise on each individual data set leading to large variance. Conversely, a large value of λ pulls the weight parameters towards zero leading to large bias.

Note that the bias–variance decomposition is of limited practical value because it is based on averages with respect to ensembles of data sets, whereas in practice we have only the single observed data set. If we had a large number of independent training sets of a given size, we would be better off combining them into a single larger training set, which of course would reduce the level of over-fitting for a given model complexity. Nevertheless, the bias–variance decomposition often provides useful insights into the model complexity issue, and although we have introduced it in this chapter from the perspective of regression problems, the underlying intuition has broad applicability.

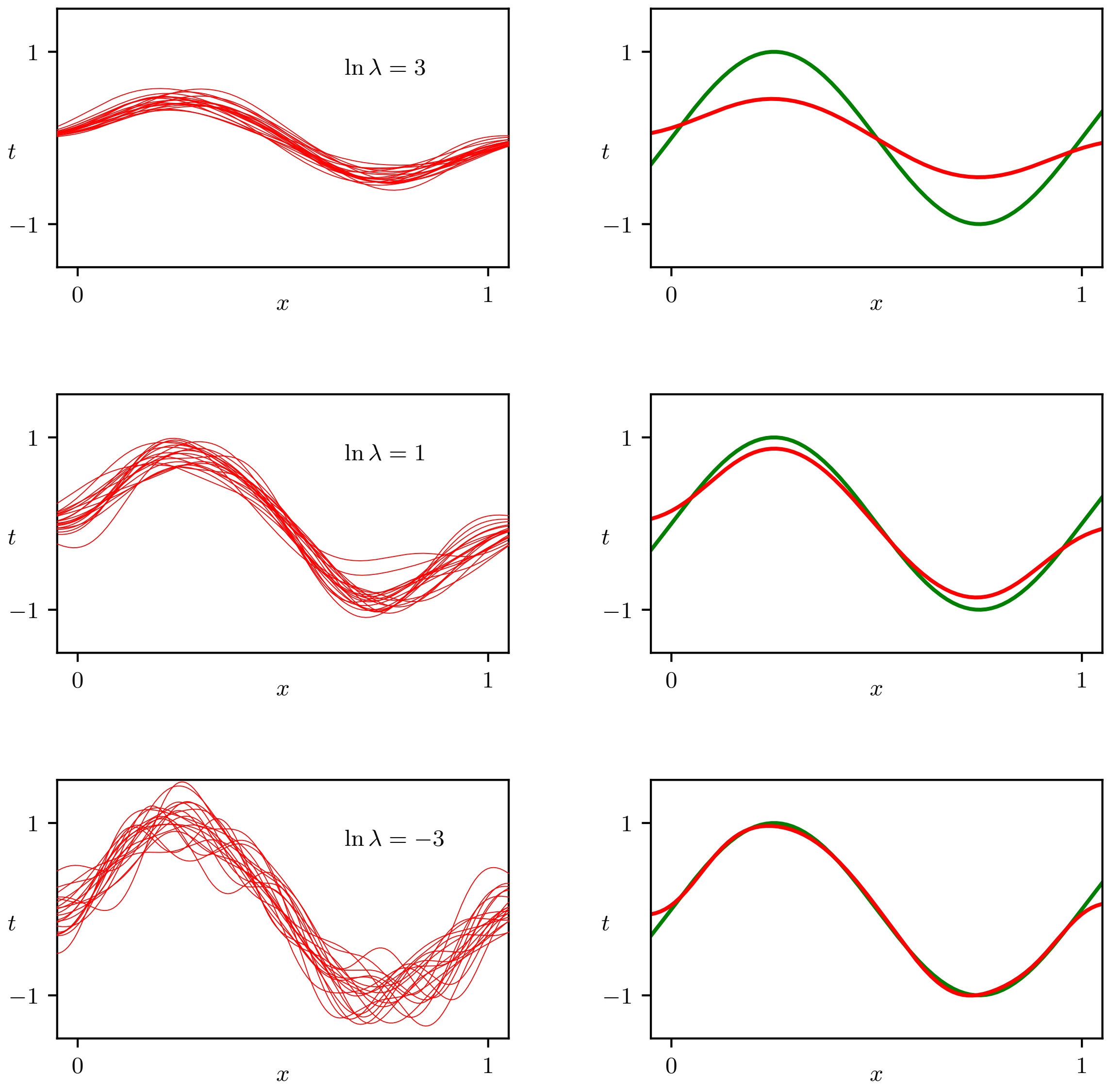

Figure 4.7 Illustration of the dependence of bias and variance on model complexity governed by a regularization parameter λ, using the sinusoidal data from Chapter 1. There are $L = 100$ data sets, each having $N = 25$ data points, and there are 24 Gaussian basis functions in the model so that the total number of parameters is $M = 25$ including the bias parameter. The left column shows the result of fitting the model to the data sets for various values of $\ln \lambda$ (for clarity, only 20 of the 100 fits are shown). The right column shows the corresponding average of the 100 fits (red) along with the sinusoidal function from which the data sets were generated (green).

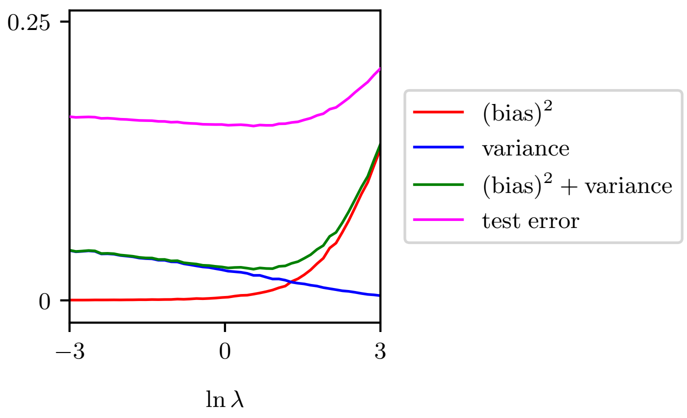

Figure 4.8 Plot of squared bias and variance, together with their sum, corresponding to the results shown in Figure 4.7. Also shown is the average test set error for a test data set size of 1,000 points. The minimum value of $(\text{bias})^2 + \text{variance}$ occurs around $\ln\lambda = 0.43$, which is close to the value that gives the minimum error on the test data.

Exercises

4.1 ($\star$) Consider the sum-of-squares error function given by (1.2) in which the function $y(x, \mathbf{w})$ is given by the polynomial (1.1). Show that the coefficients $\mathbf{w} = \{w_i\}$ that minimize this error function are given by the solution to the following set of linear equations:

$$\sum_{j=0}^{M} A_{ij} w_j = T_i \tag{4.53}$$

where

$$A_{ij} = \sum_{n=1}^{N} (x_n)^{i+j}, \qquad T_i = \sum_{n=1}^{N} (x_n)^i t_n. \tag{4.54}$$

Here a suffix i or j denotes the index of a component, whereas $(x)^i$ denotes x raised to the power of i.

4.2 ($\star$) Write down the set of coupled linear equations, analogous to (4.53), satisfied by the coefficients w_i that minimize the regularized sum-of-squares error function given by (1.4).

4.3 ($\star$) Show that the tanh function defined by

$$\tanh(a) = \frac{e^a - e^{-a}}{e^a + e^{-a}} \tag{4.55}$$

and the logistic sigmoid function defined by (4.6) are related by

$$\tanh(a) = 2\sigma(2a) - 1. \tag{4.56}$$

Hence, show that a general linear combination of logistic sigmoid functions of the form

$$y(x, \mathbf{w}) = w_0 + \sum_{j=1}^{M} w_j \sigma\left(\frac{x - \mu_j}{s}\right) \tag{4.57}$$