01.Linear Models for Regression

Supervised Learning

We turn now to a discussion of super□vised learning, starting with 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 vec□tor x of input variables. We have already encountered an example of a regression problem when we considered polynomial curve fitting.
The simplest form of linear regression.
models are also linear functions of the input variables. However, we can obtain a much more useful class of functions by taking linear combinations of a fixed set of nonlinear functions of the *input variables*, known as *basis functions*.
More generally, from a probabilistic perspective, we aim to model the predictive distribution *p(t|x)* because this expresses our uncertainty about the value of *t* for each value of *x*. From this conditional dis□tribution we can make predictions of *t*, for any new value of *x*, in such a way as to minimize the expected value of a suitably chosen loss function.

Linear Basis Function Models

The simplest linear model for regression is one that involves a linear combination of the input variables: $y(x, w) = w_0 + w_1x_1 + \ldots + w_Dx_D$ where $\mathrm{x} = (x_1, \ldots, x_D)^T$
This is often simply known as linear regression.
We therefore extend the class of models by considering linear combinations of fixed nonlinear functions of the input variables, of the form:

$$y(\mathrm{x}, \mathrm{w}) = \omega_0 + \sum_{j=1}^{M-1} \omega_j \phi_j(\mathrm{x}), \text{ where } \phi_j(\mathrm{x}) \text{ are known as basis function}$$

The total number of parameter are M as the maximum index is *M*. The parameter ω_0 is called *bias parameter* as it allows any fixed offset in the data. Is useful to define a dummy basis function $\phi_0(\mathrm{x}) = 1$ so y(*x*, *w*) is:

$$y(\mathrm{x}, \mathrm{w}) = \sum_{j=0}^{M-1} \omega_j \phi_j(\mathrm{x}) = \mathrm{w}^T \phi(\mathrm{x})$$

$$\text{where } \mathrm{w} = (\omega_0, \ldots, \omega_{M-1})^T \ \phi = (\phi_0, \ldots, \phi_{M-1})^T$$

The function y(x,w) can be non-linear if we use nonlinear basis functions BUT the model is still linear because the function is linear in **w**.
One limitation of polynomial basis functions is that they are global functions of the input variable, so that changes in one region of input space affect all other regions. This can be resolved by dividing the input space up into regions and fit a different polynomial in each region, leading to spline functions.

Maximum likelihood and least squares

consider the least squares approach, and its relation to maximum likelihood, in more detail. We assume that the target variable *t* is given by a deterministic func□tion ***y(x, w)*** with additive Gaussian noise so that $t = y(x, w) + \epsilon$ where ϵ is a zero mean Gaussian random variable with precision(inverse variance) β .
Thus we ca write: $p(t|\mathrm{x}, \mathrm{w}) = \aleph(t|y((\mathrm{x}, \mathrm{w}), \beta^{-1})$. Recall that, if we assume a squared loss function, then the optimal prediction, for a new value of *x*, will be given by the conditional mean of the target variable.
For a Gaussian conditional distribution similar to the one above the conditional mean will be simply $\mathbb{E}[t|x] = \int tp(t|x)dt = y(x, w)$. Note that the Gaussian noise assumption implies that the conditional distribution of t given x is unimodal, which may be inappropriate for some applications.
likelihood function, which is a function of the adjustable parameters **w** and β, in the form:

$$p(t|X, w, \beta) = \prod_{n=1}^{N} \aleph(t_n|\mathbf{w}^T\phi(\mathbf{x}_n), \beta^{-1})$$

Note that in supervised learning problems such as regres□sion (and classification), we are not seeking to model the distribution of the input variables. Thus x will always appear in the set of conditioning variables(**x** can be dropped from the expression).
Taking the logarithm of the likelihood function, and making use of the standard form for the univariate Gaussian, we have:

$$ln(p(t|\mathbf{w}, \beta)) = \sum_{n=1}^{N} ln(\aleph(t_n|\mathbf{w}^T\phi(\mathbf{x}_n), \beta^{-1})) = \frac{N}{2}ln\beta - \frac{N}{2}ln(2\pi) - \beta E_D(\mathbf{w})$$

where the sum-of-square error function is defined by $E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^T\phi(\mathbf{x}_n)\}^2$
We can use maximum likelihood to determine **w** and β.
From the gradient of the log likelihood function equation setted to zero and solved in **w** we obtain $\mathbf{w}_{ML} = (\mathit{\Phi}^T\mathit{\Phi})^{-1}\mathit{\Phi}^T\mathbf{t}$ which are known as the normal equations for the least squares problem. Here $\mathit{\Phi}$ is an N×M matrix, called the *design matrix*, whose elements are given by $\mathit{\Phi}_{nj} = \mathit{\Phi}_j(\mathbf{x}_n)$, so that

$$\mathit{\Phi} = \begin{pmatrix} nbsp; & nbsp; & nbsp; \mathit{\Phi}_0(\mathbf{x}_1) & \mathit{\Phi}_1(\mathbf{x}_1) & \dots & \mathit{\Phi}_{M-1}(\mathbf{x}_1) \\ nbsp; & nbsp; & nbsp; \mathit{\Phi}_0(\mathbf{x}_2) & \mathit{\Phi}_1(\mathbf{x}_2) & \dots & \mathit{\Phi}_{M-1}(\mathbf{x}_2) \\ nbsp; & nbsp; & nbsp; \dots & \dots & \dots & \dots \\ nbsp; & nbsp; & nbsp; \mathit{\Phi}_0(\mathbf{x}_N) & \mathit{\Phi}_1(\mathbf{x}_N) & \dots & \mathit{\Phi}_{M-1}(\mathbf{x}_N) \end{pmatrix}$$

At this point, we can gain some insight into the role of the bias parameter ω_0. If we make the bias parameter explicit, then the error function become:

$$E_D(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \omega_0 - \mathbf{w}^T\phi(\mathbf{x}_n)\}^2$$

Setting the derivative with respect to ω_0 equal to zero, and solving for ω_0, we obtain:

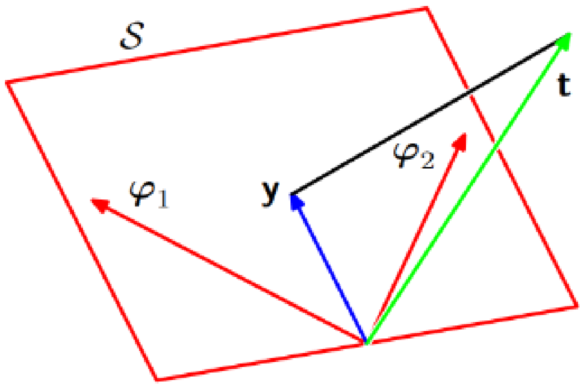
$$\omega_0 = \frac{1}{N}\sum_{n=1}^{N} t_n - \sum_{j=1}^{M-1} \omega_j \frac{1}{N}\sum_{n=1}^{N} \phi_j(\mathbf{x}_n) = \bar{t} - \sum_{j=1}^{M-1} \omega_j \bar{\phi}_j$$

Thus the bias ω_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 with respect to the noise precision parameter β, giving:

$$\frac{1}{\beta_M L} = \frac{1}{N}\sum_{n-1}^{N}\{t_n - \mathbf{w}_{ML}^T \phi(\mathbf{x}_n)\}^2$$

and so we see that the inverse of the noise precision is given by the residual variance of the target values around the regression function.

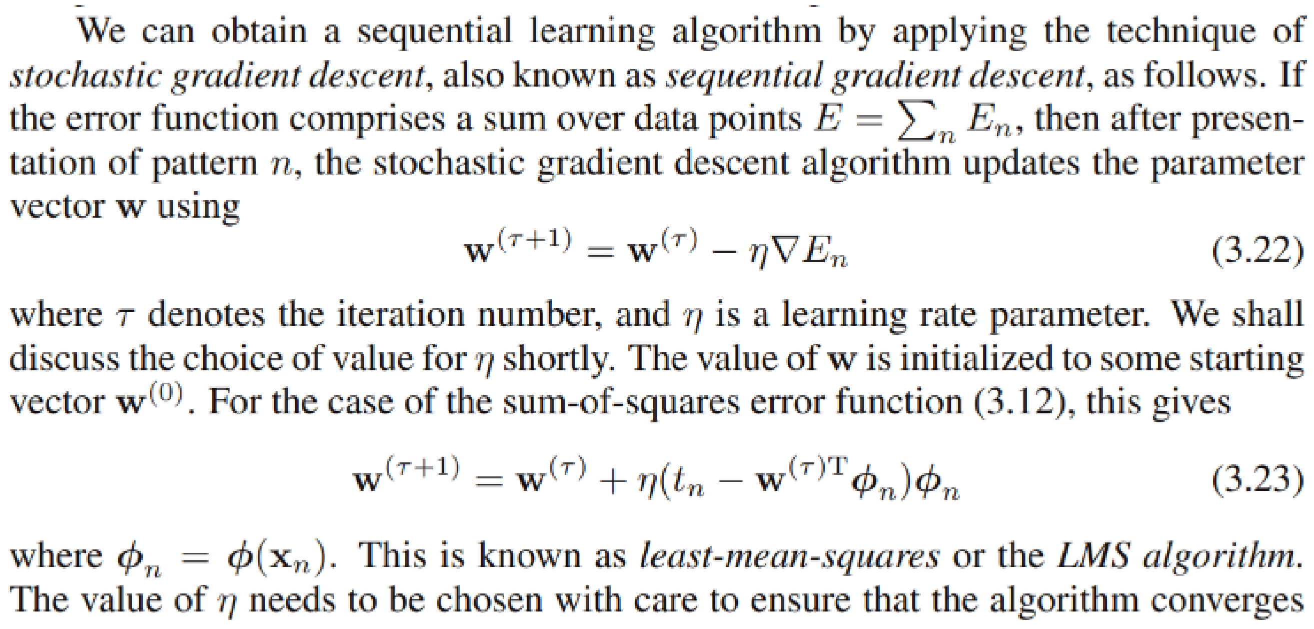
Geometry of least squares



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 φ_j of length N with elements $\phi_j(\mathbf{x}_n)$.

Sequential learning

Batch techniques, such as the maximum likelihood solution, which in□volve processing the entire training set in one go, can be computationally costly for large data sets. If the data set is sufficiently large, it may be worthwhile to use sequential algorithms, also known as on-line algorithms, in which the data points are considered one at a time, and the model parameters up□dated after each such presentation. Sequential learning is also appropriate for real□time applications in which the data observations are arriving in a continuous stream, and predictions must be made before all of the data points are seen.



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 pattern n, the stochastic gradient descent algorithm updates the parameter vector $\mathbf{w}$ using

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} - \eta \nabla E_n \tag{3.22}$$

where τ denotes the iteration number, and η is a learning rate parameter. We shall discuss the choice of value for η shortly. The value of $\mathbf{w}$ is initialized to some starting vector $\mathbf{w}^{(0)}$. For the case of the sum-of-squares error function (3.12), this gives

$$\mathbf{w}^{(\tau+1)} = \mathbf{w}^{(\tau)} + \eta(t_n - \mathbf{w}^{(\tau)\mathrm{T}}\boldsymbol{\phi}_n)\boldsymbol{\phi}_n \tag{3.23}$$

where $\boldsymbol{\phi}_n = \boldsymbol{\phi}(\mathbf{x}_n)$. This is known as *least-mean-squares* or the *LMS algorithm*. The value of η needs to be chosen with care to ensure that the algorithm converges

Regularized least squares

We introduced the idea of adding a regularization term to an error function in order to control over-fitting, so that the total error function to be minimized takes the form $E_D(\mathbf{w}) + \lambda E_W(\mathbf{w})$ 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 sim□plest forms of regularizer is given by the sum-of-squares of the weight vector ele□ments $E_w(\mathbf{w}) = \frac{1}{2}\mathbf{w}^T\mathbf{w}$. If we also consider the sum-of-squares error function given by $E(\mathbf{w}) = \frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{W}^T\phi(\mathbf{x}_n)\}^2$ then the total error function becomes

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

This particular choice of regularizer is known as weight decay because in sequential learning algorithms, it encourages weight values to decay towards zero, unless supported by the data. In statistics, it provides an ex□ample 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 w, and so its exact minimizer can be found in closed form. Specifically we obtain $\mathbf{w} = (\lambda\mathbf{I} + \mathit{\Phi}^T\mathit{\Phi})^{-1}\mathit{\Phi}^T\mathbf{t}$, that represent a simple extension of the the least-square solution.

A more general regularizer is sometimes used, for which the regularized error takes the form

$$\frac{1}{2}\sum_{n=1}^{N}\{t_n - \mathbf{w}^{\mathrm{T}}\boldsymbol{\phi}(\mathbf{x}_n)\}^2 + \frac{\lambda}{2}\sum_{j=1}^{M}|w_j|^q \tag{3.29}$$

where $q = 2$ corresponds to the quadratic regularizer (3.27). Figure 3.3 shows contours of the regularization function for different values of q.

The case of $q = 1$ is know as the *lasso* in the statistics literature (Tibshirani, 1996). It has the property that if λ is sufficiently large, some of the coefficients w_j are driven to zero, leading to a *sparse* model in which the corresponding basis functions play no role. To see this, we first note that minimizing (3.29) is equivalent to minimizing the unregularized sum-of-squares error (3.12) subject to the constraint

$$\sum_{j=1}^{M}|w_j|^q \leqslant \eta \tag{3.30}$$

for an appropriate value of the parameter η, where the two approaches can be related using Lagrange multipliers. The origin of the sparsity can be seen from Figure 3.4, which shows that the minimum of the error function, subject to the constraint (3.30). As λ is increased, so an increasing number of parameters are driven to zero.

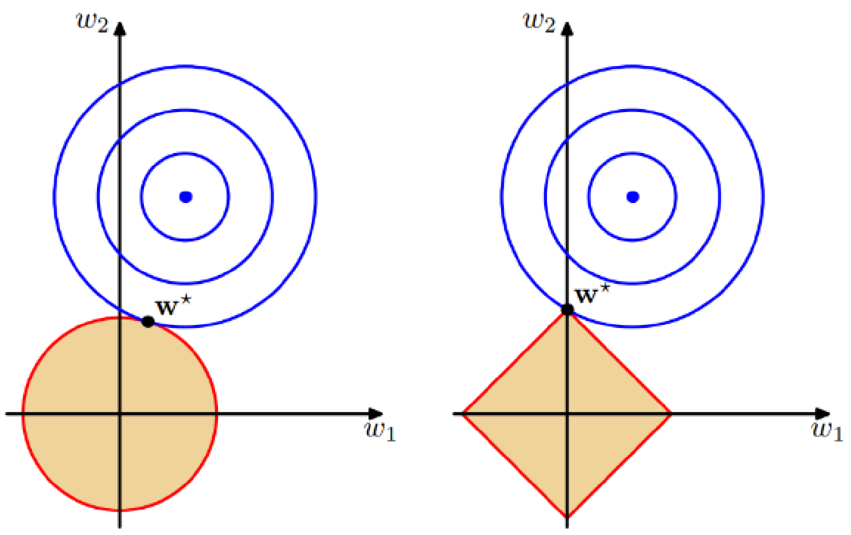


Figure 3.4 Plot of the contours of the unregularized error function (blue) along with the constraint region (3.30) for the quadratic regularizer $q = 2$ on the left and the lasso regularizer $q = 1$ on the right, in which the optimum value for the parameter vector $\mathbf{w}$ is denoted by $\mathbf{w}^\star$. The lasso gives a sparse solution in which $w_1^\star = 0$.

Multiple outputs

In some applica□tions, we may wish to predict K > 1 target variables, which we denote collectively by the target vector $\mathbf{t}$. This could be done by introducing a different set of basis func□tions for each component of $\mathbf{t}$, leading to multiple, independent regression problems. However, a more interesting, and more common, approach is to use the same set of basis functions to model all of the components of the target vector so that $y(\mathbf{x}, \mathbf{w}) = W^T\phi(\mathbf{x})$ where y is a K-dimensional column vector, $\mathbf{W}$ is an M × K matrix of parameters, and $\Phi(\mathbf{x})$ is an M-dimensional column vector with elements $\Phi_j(\mathbf{x})$, with $\Phi_0(\mathbf{x}) = 1$ as before. Suppose we take the conditional distribution of the target vector to be an isotropic Gaussian of the form $p(\Phi t|\mathbf{t}, \mathbf{W}, \beta) = \aleph(\mathbf{t}|\mathbf{W}^T\Phi(\mathbf{x}), \beta^{-1}\mathbf{I})$. 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 × K such that the n^{th} row is given by $\mathbf{t}_n^T$ 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:

$$ln(p(\mathbf{T}|\mathbf{X}, \mathbf{W}, \beta)) = \sum_{n=1}^{N} ln\aleph(\mathbf{t}_n|\mathbf{W}^T\phi(\mathbf{x}), \beta^{-1}\mathbf{I}) = \frac{NK}{2}ln(\frac{\beta}{2\pi}) - \frac{\beta}{2}\sum_{n=1}^{N}\|\mathbf{t}_n - \mathbf{W}^T\phi(\mathbf{x}_n)\|^2$$

As before, we can maximize this function with respect to $\mathbf{W}$, giving $\mathbf{W}_{ML} = (\Phi^T\Phi)^{-1}\Phi^T\mathbf{T}$. If we examine this result for each target variable t_k, we have $\mathbf{w}_k = (\Phi^T\Phi)^{-1}\Phi^T\mathbf{t}_k = \Phi^\dagger\mathbf{t}_k$ where t_k is an N-dimensional column vector with components t_nk for n = 1,...,N. Thus the solution to the regression problem decouples between the different target variables, and we need only compute a single pseudo-inverse matrix $\Phi^\dagger$, which is shared by all of the vectors $\mathbf{w}_k$.

Bayesian Linear Regression

We turn to a Bayesian treatment of linear regression, which will avoid the over-fitting problem of maximum likelihood, and which will also lead to automatic methods of determining model complexity using the training data alone.

Parameter distribution

We will introduce a prior probability distribution over the model parameter $\mathbf{w}$ where we will treat the noise precision parameter β as a known constant. The conjugate prior of the likelihood function $p(\mathbf{t}|\mathbf{w})$ is given by the Gaussian distribution of the form $p(\mathbf{w}) = \aleph(\mathbf{w}|\mathbf{m}_0, \mathbf{S}_0)$ having mean m_0 and covariance $\mathbf{S}_0$.
Next we compute the posterior distribution(which is still Gaussian), which is proportional to the product of the likelihood function and the prior. The posterior distribution has the form of $p(\mathbf{w}|\mathbf{t}) = \aleph(\mathbf{w}|\mathbf{m}_N, \mathbf{S}_N)$ where $\mathbf{m}_N = \mathbf{S}_N(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta\Phi^T\mathbf{t})$ and $\mathbf{S}_N^{-1} = \mathbf{S}_0^{-1} + \beta\Phi^T\Phi$
Note that because the posterior distribution is Gaussian, its mode coincides with its mean. Thus the maximum posterior weight vector is simply given by $\mathbf{w}_{MAP} = \mathbf{m}_N$.

Predictive distribution

In practice we want to make predictions of *t* for new values of $\mathbf{x}$. This requires that we evaluate the *predictive distribution* defined by

$$p(t|\mathbf{t}, \alpha, \beta) = \int p(t|\mathbf{w}, \beta)p(\mathbf{w}, \mathbf{t}, \alpha, \beta)d\mathbf{w}$$

in which *t* is the vector of target values from the training set. The predictive distribution takes the form $p(t|\mathbf{x}, \mathbf{t}, \alpha, \beta) = \aleph(t|\mathbf{m}_N^T\Phi(\mathbf{x}), \sigma_N^2(\mathbf{x}))$ where the variance $\sigma_N^2(\mathbf{x})$ of the predictive function is given by $\sigma_N^2(\mathbf{x}) = \frac{1}{\beta} + \Phi(\mathbf{x})^T\mathbf{S}_N\Phi(\mathbf{x})$. The first term in $\sigma_N^2(\mathbf{x})$ represents the noise on the data whereas the second term reflects the uncertainty associated with the parameters w. Because the noise process and the distribution of w are independent Gaussians, their variances are additive, for $N \to \infty$ the second term goes to zero and the variance of the predictive distribution arises solely from the additive noise governed by the parameter β. It can also be proven that $\sigma^2_{N+1}(\mathbf{x}) \leq \sigma^2_N(\mathbf{x})$.

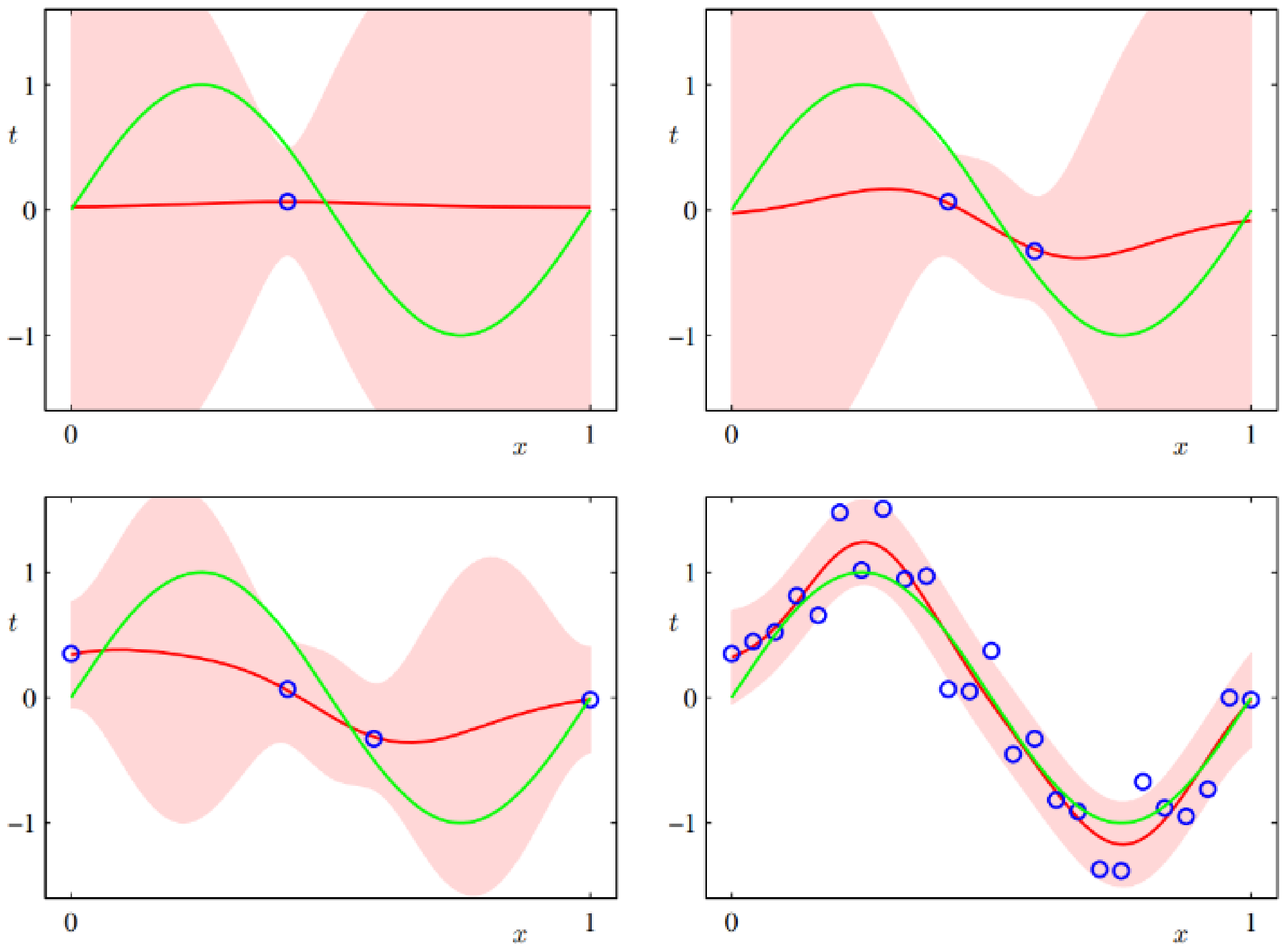


Figure 3.8 Examples of the predictive distribution (3.58) for a model consisting of 9 Gaussian basis functions of the form (3.4) using the synthetic sinusoidal data set of Section 1.1. See the text for a detailed discussion.

The green curves correspond to the function $sin(2\pi x)$ from which the data points were generated. Data sets of size N = 1, N = 2, N = 4, and N = 25 are shown in the four plots by the blue circles. For each plot, the red curve shows the mean of the corresponding Gaussian predictive distribution, and the red shaded region spans one standard deviation either side of the mean. Note that the predictive uncertainty depends on x and is smallest in the neighbourhood of the data points. Also note that the level of uncertainty decreases as more data points are observed.

Equivalent kernel

The mean of the predictive distribution at a point $\mathbf{x}$ is given by a linear combination of the training set target variables t_n, so that we can write

$$y(\mathbf{x}, \mathbf{m}_N) = \mathbf{m}_N^T \phi(\mathbf{x}) = \beta \phi(\mathbf{x})^T \mathbf{S}_N \Phi^T \mathbf{t} = \sum_{n=1}^{N} \beta \phi(\mathbf{x})^T \mathbf{S}_N \phi(\mathbf{x}_n) t_n = \sum_{n=1}^{N} k(\mathbf{x}, \mathbf{x}_n) t_n$$

where the function $k(\mathbf{x}, \mathbf{x}') = \beta \phi(\mathbf{x})^T S_N \phi(\mathbf{x}')$ is known as the smoother matrix or the equivalent kernel. Note that the equivalent kernel depends on the input values x_n from the data set because these appear in the definition of S_N . We should weight local evidence more strongly

than distant evidence. Note that this localization property holds not only for the localized Gaussian basis functions but also for the nonlocal polynomial and sigmoidal basis functions.
The formulation of linear regression in terms of a kernel function suggests an alternative approach to regression as follows. Instead of introducing a set of basis functions, which implicitly determines an equivalent kernel, we can instead define a localized kernel directly and use this to make predictions for new input vectors $\mathbf{x}$, given the observed training set. This leads to a practical framework for regression (and classification) called Gaussian processes.
We have seen that the effective kernel defines the weights by which the training set target values are combined in order to make a prediction at a new value of $\mathbf{x}$, and it can be shown that these weights sum to one, in other words $\sum_{n=1}^{N} k(\mathbf{x}, \mathbf{x}_n) = 1$ for all values of $\mathbf{x}$
Note that the kernel function can be negative as well as positive, so although it satisfies a summation constraint, the corresponding predictions are not necessarily convex combinations of the training set target variables.
we note that the equivalent kernel $k(\mathbf{x}, \mathbf{x}') = \beta\phi(\mathbf{x})^T S_N \phi(\mathbf{x}')$ satisfies an important property shared by kernel functions in general, namely that it can be expressed in the form an inner product with respect to a vector ψ(x) of nonlinear functions, so that
$k(\mathbf{x}, \mathbf{z}) = \psi(\mathbf{x})^T \psi(\mathbf{z})$ where $\psi(\mathbf{x}) = \beta^{1/2} S_N^{1/2} \phi(\mathbf{x})$.

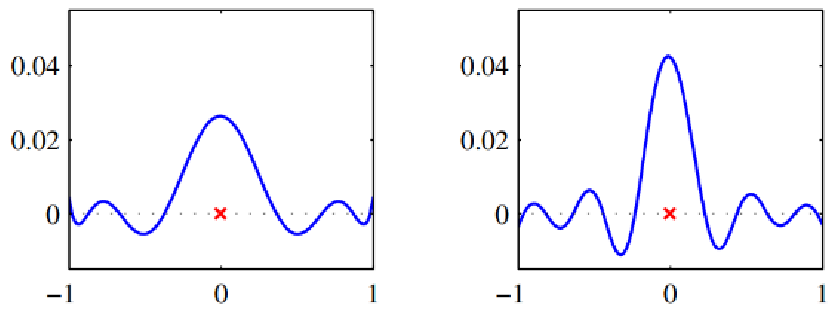


Figure 3.11 Examples of equivalent kernels $k(x, x')$ for $x = 0$ plotted as a function of x', corresponding (left) to the polynomial basis functions and (right) to the sigmoidal basis functions shown in Figure 3.1. Note that these are localized functions of x' even though the corresponding basis functions are nonlocal.