Introduction to Radial Basis Function Networks

Mark J. L. Orr[1]

Centre for Cognitive Science,
University of Edinburgh,
2, Buccleuch Place,
Edinburgh EH8 9LW, Scotland

April 1996

Abstract

This document is an introduction to radial basis function (RBF) networks, a type of artificial neural network for application to problems of supervised learning (e.g. regression, classification and time series prediction). It is now only available in PostScript[2] (an older and now unsupported hyper-text version[3] may be available for a while longer).

The document was first published in 1996 along with a package of Matlab functions[4] implementing the methods described. In 1999 a new document, *Recent Advances in Radial Basis Function Networks*, became available[5], with a second and improved version of the Matlab package[6].

[1] mjo@anc.ed.ac.uk
[2] www.anc.ed.ac.uk/∼mjo/papers/intro.ps
[3] www.anc.ed.ac.uk/∼mjo/intro/intro.html
[4] www.anc.ed.ac.uk/∼mjo/software/rbf.zip
[5] www.anc.ed.ac.uk/∼mjo/papers/recad.ps
[6] www.anc.ed.ac.uk/∼mjo/software/rbf2.zip

Contents

1 Introduction

This document is an introduction to linear neural networks, particularly radial basis function (RBF) networks. The approach described places an emphasis on retaining, as much as possible, the linear character of RBF networks[7], despite the fact that for good generalisation there has to be some kind of nonlinear optimisation. The two main advantages of this approach are keeping the mathematics simple (it is just linear algebra) and the computations relatively cheap (there is no optimisation by general purpose gradient descent algorithms).

Linear models have been studied in statistics for about 200 years and the theory is applicable to RBF networks which are just one particular type of linear model. However, the fashion for neural networks, which started in the mid-80's, has given rise to new names for concepts already familiar to statisticians [27]. Table 1 gives some examples. Such terms are used interchangeably in this document.

statistics	*neural networks*
model	network
estimation	learning
regression	supervised learning
interpolation	generalisation
observations	training set
parameters	(synaptic) weights
independent variables	inputs
dependent variables	outputs
ridge regression	weight decay

Table 1: Equivalent terms in statistics and neural networks.

The document is structured as follows. We first outline supervised learning (section 2), the main application area for RBF networks, including the related areas of classification and time series prediction (section 2.2). We then describe linear models (section 3) including RBF networks (section 3.2). Least squares optimisation (section 4), including the effects of ridge regression, is then briefly reviewed followed by model selection (section 5). After that we cover ridge regression (section 6) in more detail and lastly we look at forward selection (section 7) for building networks. Most of the mathematical details are put in an appendix (section A).

[7] For alternative approaches see, for example, the work of Platt [24] and associates [21] and of Fritzke [15].

2 Supervised Learning

A ubiquitous problem in statistics with applications in many areas is to guess or estimate a function from some example input-output pairs with little or no knowledge of the form of the function. So common is the problem that it has different names in different disciplines (e.g. nonparametric regression, function approximation, system identification, inductive learning).

In neural network parlance, the problem is called *supervised learning*. The function is *learned* from the examples which a *teacher* supplies. The set of examples, or *training set*, contains elements which consist of paired values of the independent (input) variable and the dependent (output) variable. For example, the independent variable in the functional relation

$$y \;=\; f(\mathbf{x})$$

is $\mathbf{x}$ (a vector) and the dependent variable is y (a scalar). The value of the variable y depends, through the function f, on each of the components of the vector variable

$$\mathbf{x} \;=\; \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} .$$

Note that we are using bold lower-case letters for vectors and italicised lower-case letters for scalars, including scalar valued functions like f (see appendix A.1 on notational conventions).

The general case is where both the independent and dependent variables are vectors. This adds more mathematics but little extra insight to the special case of univariate output so, for simplicity, we will confine our attention to the latter. Note, however, that multiple outputs can be treated in a special way in order to reduce redundancy [2].

The training set, in which there are p pairs (indexed by i running from 1 up to p), is represented by

$$\mathcal{T} \;=\; \{(\mathbf{x}_i,\, \hat{y}_i)\}_{i=1}^{p} \,. \tag{2.1}$$

The reason for the hat over the letter y (another convention – see appendix A.1– indicating an estimate or uncertain value) is that the output values of the training set are usually assumed to be corrupted by noise. In other words, the correct value to pair with $\mathbf{x}_i$, namely y_i, is unknown. The training set only specifies $\hat{y}_i$ which is equal to y_i plus a small amount of unknown noise.

In real applications the independent variable values in the training set are often also affected by noise. This type of noise is more difficult to model and we shall not attempt it. In any case, taking account of noise in the inputs is approximately equivalent to assuming noiseless inputs but with an increased amount of noise in the outputs.

2.1 Nonparametric Regression

There are two main subdivisions of regression problems in statistics: *parametric* and *nonparametric*. In parametric regression the form of the functional relationship between the dependent and independent variables is known but may contain parameters whose values are unknown and capable of being estimated from the training set. For example, fitting a straight line,

$$f(x) \;=\; a\,x + b\,,$$

to a bunch of points, $\{(x_i,\, \hat{y}_i)\}_{i=1}^{p}$, (see figure 1) is parametric regression because the functional form of the dependence of y on x is given, even though the values of a and b are not. Typically, in any given parametric problem, the free parameters, as well as the dependent and independent variables, have meaningful interpretations, like "initial water level" or "rate of flow".

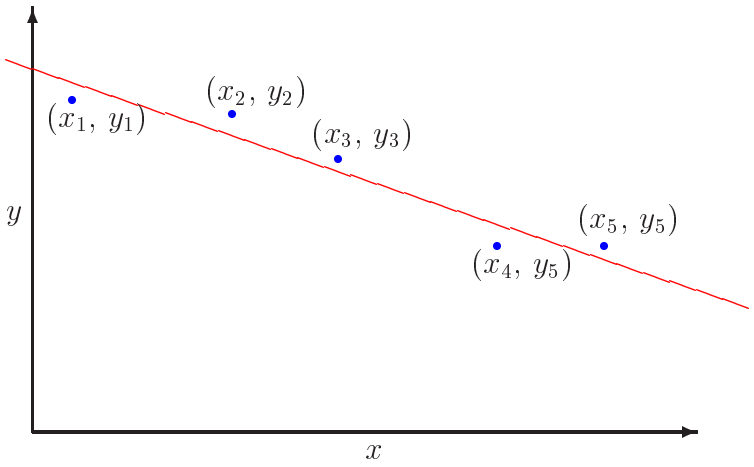

Figure 1: Fitting a straight line to a bunch of points is a kind of parametric regression where the form of the model is known.

The distinguishing feature of nonparametric regression is that there is no (or very little) *a priori* knowledge about the form of the true function which is being estimated. The function is still modelled using an equation containing free parameters but in a way which allows the class of functions which the model can represent to be very broad. Typically this involves using many free parameters which have no physical meaning in relation to the problem. In parametric regression there is typically a small number of parameters and often they have physical interpretations.

Neural networks, including radial basis function networks, are nonparametric models and their weights (and other parameters) have no particular meaning in relation to the problems to which they are applied. Estimating values for the weights of a neural network (or the parameters of any nonparametric model) is never the primary goal in supervised learning. The primary goal is to estimate the underlying function (or at least to estimate its output at certain desired values of the input).

On the other hand, the main goal of parametric regression can be, and often is, the estimation of parameter values because of their intrinsic meaning.

2.2 Classification and Time Series Prediction

In classification problems the goal is to assign previously unseen patterns to their respective classes based on previous examples from each class. Thus the output of the learning algorithm is one of a discrete set of possible classes rather than, as in nonparametric regression (section 2.1), the value of a continuous function. However, classification problems can be made to look like nonparametric regression if the outputs of the estimated function are interpreted as being proportional to the probability that the input belongs to the corresponding class.

The discrete class labels of the training set (equation 2.1) outputs are given numerical values by interpreting the k-th class label as a probability of 1 that the example belongs to the class and a probability of 0 that it belongs to any other class. Thus the training output values are vectors of length equal to the number of classes and containing a single one (and otherwise zeros). After training the network responds to a new pattern with continuous values in each component of the output vector which can be interpreted as being proportional to class probability and used to generate a class assignment. For a comparison of various types of algorithms on different classification problems see [23].

Time series prediction is concerned with estimating the next value and future values of a sequence such as

$$\ldots,\ y_{t-3},\ y_{t-2},\ y_{t-1},\ ?,\ ?,\ \ldots$$

but usually not as a explicit function of time. Normally time series are modeled as *auto-regressive* in nature, i.e. the outputs, suitably delayed, are also the inputs:

$$y_t\ =\ f(y_{t-1},\, y_{t-2},\, y_{t-3},\, \ldots)\,.$$

To create the training set (equation 2.1) from the available historical sequence first requires the choice of how many and which delayed outputs affect the next output. This is one of a number of complications which make time series prediction a more difficult problem than straight regression or classification. Others include regime switching and asynchronous sampling [13].

3 Linear Models

A linear model for a function $y(\mathbf{x})$ takes the form

$$f(\mathbf{x}) \;=\; \sum_{j=1}^{m} w_j \, h_j(\mathbf{x}) \,. \tag{3.1}$$

The model f is expressed as a linear combination of a set of m fixed functions (often called *basis functions* by analogy with the concept of a vector being composed of a linear combination of basis vectors). The choice of the letter 'w' for the coefficients of the linear combinations and the letter 'h' for the basis functions reflects our interest in neural networks which have weights and hidden units.

The flexibility of f, its ability to fit many different functions, derives only from the freedom to choose different values for the weights. The basis functions and any parameters which they might contain are fixed. If this is not the case, if the basis functions can change during the learning process, then the model is nonlinear.

Any set of functions can be used as the basis set although it helps, of course, if they are well behaved (differentiable). However, models containing only basis functions drawn from one particular class have a special interest. Classical statistics abounds with linear models whose basis functions are polynomials ($h_j(x) = x^j$ or variations on the theme). Combinations of sinusoidal waves (Fourier series),

$$h_j(x) \;=\; \sin\left(\frac{2\,\pi\,j\,(x-\theta_j)}{m}\right) \,,$$

are often used in signal processing applications. Logistic functions, of the sort

$$h(\mathbf{x}) = \frac{1}{1+\exp(\mathbf{b}^\top \mathbf{x} - b_0)} \,,$$

are popular in artificial neural networks, particularly in *multi-layer perceptrons* (MLPs) [8].

A familiar example, almost the simplest polynomial, is the straight line

$$f(x) = a\,x + b$$

which is a linear model whose two basis functions are

$$h_1(x) \;=\; 1 \,,$$

$$h_2(x) \;=\; x \,,$$

and whose weights are $w_1 = b$ and $w_2 = a$. This is, of course, a very simple model and is not flexible enough to be used for supervised learning (section2).

Linear models are simpler to analyse mathematically. In particular, if supervised learning problems (section 2) are solved by least squares (section 4) then it is possible to derive and solve a set of equations for the optimal weight values implied by the training set (equation 2.1). The same does not apply for nonlinear models, such as MLPs, which require iterative numerical procedures for their optimisation.

3.1 Radial Functions

Radial functions are a special class of function. Their characteristic feature is that their response decreases (or increases) monotonically with distance from a central point. The centre, the distance scale, and the precise shape of the radial function are parameters of the model, all fixed if it is linear.

A typical radial function is the Gaussian which, in the case of a scalar input, is

$$h(x) \;=\; \exp\left(-\frac{(x-c)^2}{r^2}\right) .$$

Its parameters are its centre c and its radius r. Figure 2 illustrates a Gaussian RBF with centre $c = 0$ and radius $r = 1$.

A Gaussian RBF monotonically decreases with distance from the centre. In contrast, a multiquadric RBF which, in the case of scalar input, is

$$h(x) \;=\; \frac{\sqrt{r^2 + (x-c)^2}}{r} ,$$

monotonically increases with distance from the centre (see Figure 2). Gaussian-like RBFs are local (give a significant response only in a neighbourhood near the centre) and are more commonly used than multiquadric-type RBFs which have a global response. They are also more biologically plausible because their response is finite.

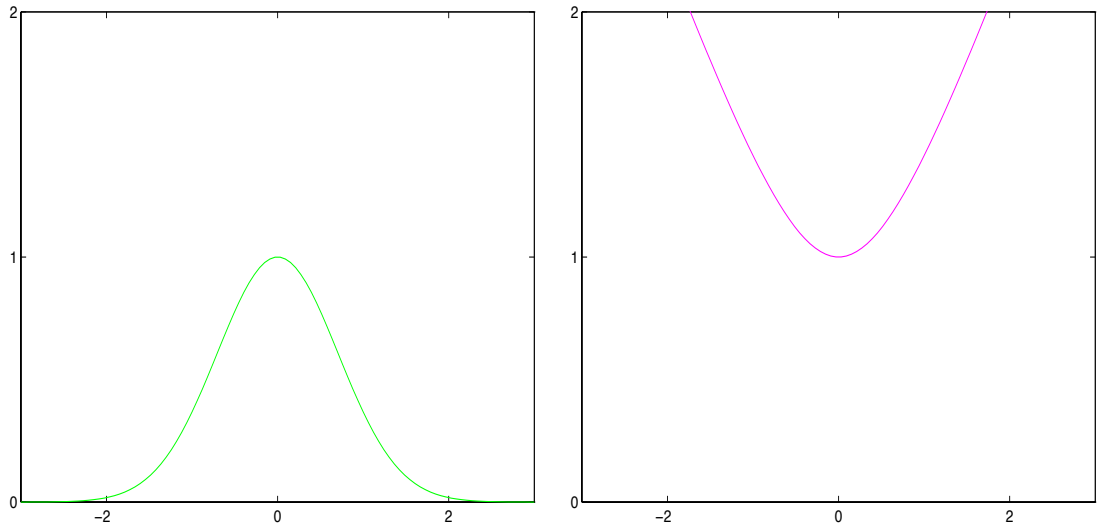

Figure 2: Gaussian (left) and multiquadric RBFs.

Formulae for other types of RBF functions and for multiple inputs are given in appendix A.3.

3.2 Radial Basis Function Networks

Radial functions are simply a class of functions. In principle, they could be employed in any sort of model (linear or nonlinear) and any sort of network (single-layer or

multi-layer). However, since Broomhead and Lowe's 1988 seminal paper [3], radial basis function networks (RBF networks) have traditionally been associated with radial functions in a single-layer network such as shown in figure 3.

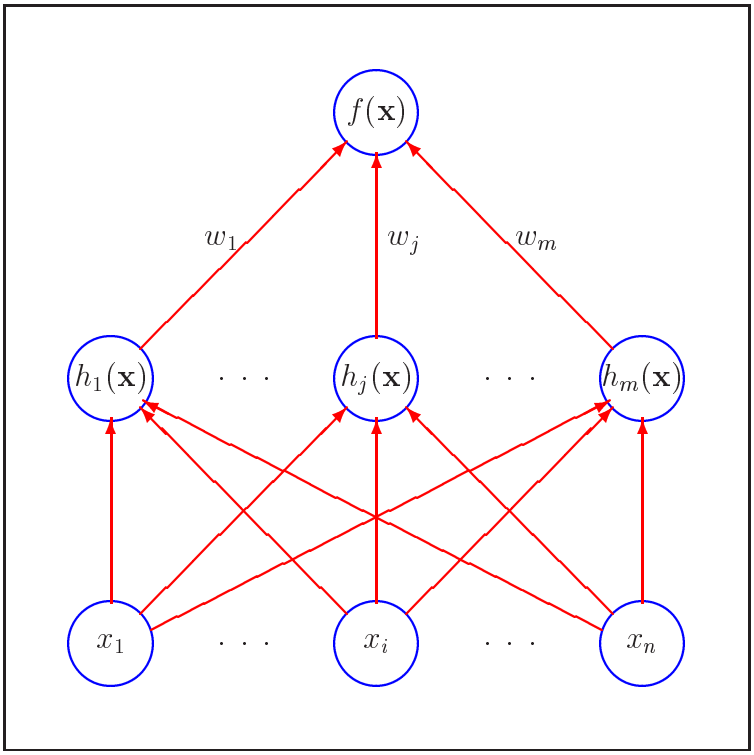

Figure 3: The traditional radial basis function network. Each of n components of the input vector $\mathbf{x}$ feeds forward to m basis functions whose outputs are linearly combined with weights $\{w_j\}_{j=1}^{m}$ into the network output $f(\mathbf{x})$.

An RBF network is nonlinear if the basis functions can move or change size or if there is more than one hidden layer. Here we focus on single-layer networks with functions which are fixed in position and size. We do use nonlinear optimisation but only for the regularisation parameter(s) in ridge regression (section 6) and the optimal subset of basis functions in forward selection (section 7). We also avoid the kind of expensive nonlinear gradient descent algorithms (such as the *conjugate gradient* and *variable metric* methods [25]) that are employed in explicitly nonlinear networks. Keeping one foot firmly planted in the world of linear algebra makes analysis easier and computations quicker.

4 Least Squares

When applied to supervised learning (section 2) with linear models (section 3) the least squares principle leads to a particularly easy optimisation problem. If the model is

$$f(\mathbf{x}) \;=\; \sum_{j=1}^{m} w_j \, h_j(\mathbf{x})$$

and the training set (section 2) is $\{(\mathbf{x}_i,\, \hat{y}_i)\}_{i=1}^{p}$, then the least squares recipe is to minimise the *sum-squared-error*

$$S \;=\; \sum_{i=1}^{p} \left(\hat{y}_i - f(\mathbf{x}_i)\right)^2 \,, \tag{4.1}$$

with respect to the weights of the model. If a weight penalty term is added to the sum-squared-error, as is the case with ridge regression (section 6), then the following *cost function* is minimised

$$C \;=\; \sum_{i=1}^{p} \left(\hat{y}_i - f(\mathbf{x}_i)\right)^2 + \sum_{j=1}^{m} \lambda_j \, w_j^2 \,, \tag{4.2}$$

where the $\{\lambda_j\}_{j=1}^{m}$ are regularisation parameters.

4.1 The Optimal Weight Vector

We show in appendix A.4 that minimisation of the cost function (equation 4.2) leads to a set of m simultaneous linear equations in the m unknown weights and how the linear equations can be written more conveniently as the matrix equation

$$\mathbf{A}\,\hat{\mathbf{w}} \;=\; \mathbf{H}^{\top}\hat{\mathbf{y}} \,,$$

where $\mathbf{H}$, the *design matrix*, is

$$\mathbf{H} \;=\; \begin{bmatrix} h_1(\mathbf{x}_1) & h_2(\mathbf{x}_1) & \dots & h_m(\mathbf{x}_1) \\ h_1(\mathbf{x}_2) & h_2(\mathbf{x}_2) & \dots & h_m(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ h_1(\mathbf{x}_p) & h_2(\mathbf{x}_p) & \dots & h_m(\mathbf{x}_p) \end{bmatrix} \,, \tag{4.3}$$

$\mathbf{A}^{-1}$, the *variance matrix*, is

$$\mathbf{A}^{-1} \;=\; (\mathbf{H}^\top \mathbf{H} + \mathbf{\Lambda})^{-1}\,, \tag{4.4}$$

the elements of the matrix $\mathbf{\Lambda}$ are all zero except for the regularisation parameters along its diagonal and $\hat{\mathbf{y}} = [\hat{y}_1\ \hat{y}_2\ \ldots\ \hat{y}_p]^\top$ is the vector of training set (equation 2.1) outputs. The solution is the so-called *normal equation* [26],

$$\hat{\mathbf{w}} \;=\; \mathbf{A}^{-1}\mathbf{H}^\top \hat{\mathbf{y}}\,, \tag{4.5}$$

and $\hat{\mathbf{w}} = [\hat{w}_1\ \hat{w}_2\ \ldots\ \hat{w}_m]^\top$ is the vector of weights which minimises the cost function (equation 4.2). The use of these equations is illustrated with a simple example (section 4.5).

To save repeating the analysis with and without a weight penalty, the appendices and the rest of this section includes its effects. However, any equation involving weight penalties can easily be converted back to ordinary least squares (without any penalty) simply by setting all the regularisation parameters to zero.

4.2 The Projection Matrix

A useful matrix which often crops up in the analysis of linear networks is the projection matrix (appendix A.6)

$$\mathbf{P} \;=\; \mathbf{I}_p - \mathbf{H}\,\mathbf{A}^{-1}\mathbf{H}^\top\,. \tag{4.6}$$

This square matrix projects vectors in p-dimensional space perpendicular to the m-dimensional subspace spanned by the model, though only in the case of ordinary least squares (no weight penalty). The training set output vector, $\hat{\mathbf{y}}$, lives in p-dimensional space, since there are p patterns (see figure 4). However, the model, being linear and possessing only m degrees of freedom (the m weights), can only reach points in an m-dimensional hyperplane, a subspace of p-dimensional space (assuming $m \leq p$). For example, if $p = 3$ and $m = 2$ the model can only reach points lying in some fixed plane and can never exactly match $\hat{\mathbf{y}}$ if it lies somewhere outside this plane. The least squares principle implies that the optimal model is the one with the minimum distance from $\hat{\mathbf{y}}$, i.e. the projection of $\hat{\mathbf{y}}$ onto the m-dimensional hyperplane. The mismatch or error between $\hat{\mathbf{y}}$ and the least squares model is the projection of $\hat{\mathbf{y}}$ perpendicular to the subspace and this turns out to be $\mathbf{P}\,\hat{\mathbf{y}}$.

The importance of the matrix $\mathbf{P}$ is perhaps evident from the following two equations, proven in appendix A.6. At the optimal weight (equation 4.5) the value of the cost function (equation 4.2) is

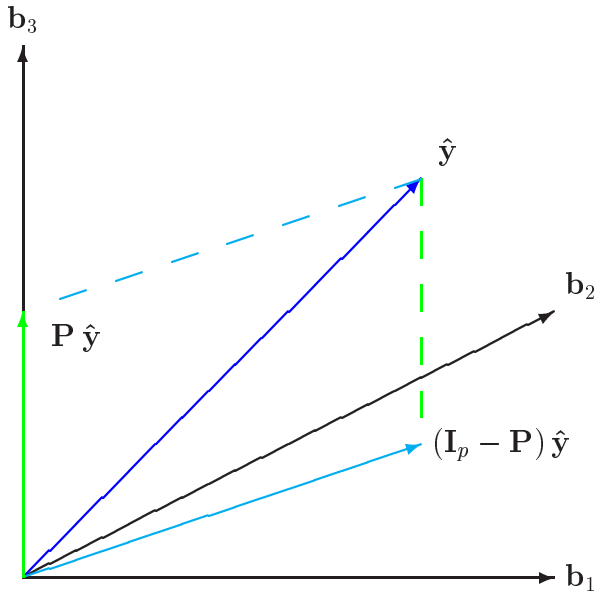

Figure 4: The model spans a 2-dimensional plane (with basis vectors $\mathbf{b}_1$ and $\mathbf{b}_2$) in 3-dimensional space (with an extra basis vector $\mathbf{b}_3$) and cannot match $\hat{\mathbf{y}}$ exactly. The least squares model is the projection of $\hat{\mathbf{y}}$ onto this plane and the error vector is $\mathbf{P}\,\hat{\mathbf{y}}$.

$$\hat{C} \;=\; \hat{\mathbf{y}}^\top \mathbf{P}\,\hat{\mathbf{y}}\,, \tag{4.7}$$

and the sum-squared-error (equation 4.1) is

$$\hat{S} \;=\; \hat{\mathbf{y}}^\top \mathbf{P}^2\,\hat{\mathbf{y}}\,. \tag{4.8}$$

These two equations, the latter in particular, will be useful later when we have to choose the best regularisation parameter(s) in ridge regression (section 6) or the best subset in forward selection (section 7).

4.3 Incremental Operations

A common problem in supervised learning is to determine the effect of adding a new basis function or taking one away, often as part of a process of trying to discover the optimal set of basis functions (see forward selection (section 7)). Alternatively,

one might also be interested in the effect of removing one of the training examples (perhaps in an attempt to clean outliers out of the data) or of including a new one (on-line learning).

These are incremental (or decremental) operations with respect to either the basis functions or the training set. Their effects can, of course, be calculated by retraining the network from scratch – the only route available when using nonlinear networks. However, in the case of linear networks, such as radial basis function networks, simple analytical formulae can be derived to cover each situation. These are based on a couple of basic formulae (appendix A.7) for handling matrix inverses. Their use can produce significant savings in compute time compared to retraining from scratch (see table 3). The details are all in appendix A.7. Formulae are given for

- adding a new basis function (appendix A.7.1),
- removing an old basis function (appendix A.7.2),
- adding a new training pattern (appendix A.7.3),
- removing an old training pattern (appendix A.7.4).

Equation A.6 for the change in the projection matrix (equation 4.6) caused by removing a basis function is particularly useful as we shall see in later sections on ridge regression (6) and forward selection (7).

4.4 The Effective Number of Parameters

Suppose you are given a set of numbers $\{x_i\}_{i=1}^{p}$ randomly drawn from a Gaussian distribution and you are asked to estimate its variance, without being told its mean. The first thing you would do is to estimate the mean,

$$\bar{x} \;=\; \frac{1}{p}\sum_{i=1}^{p} x_i \,,$$

and then use this estimate in the calculation of the variance,

$$\hat{\sigma}^2 \;=\; \frac{1}{p-1}\sum_{i=1}^{p} (x_i - \bar{x})^2 \,.$$

But where has the factor $p-1$ in this familiar formula come from? Why not divide by p, the number of samples? When a parameter such as $\bar{x}$ is estimated, as above, it is unavoidable that it fits some of the noise in the data. The apparent variance (which you would have got by dividing by p) will thus be an underestimate and, since one degree of freedom has been used up in the calculation of the mean, the balance is restored by reducing the remaining degrees of freedom by one (i.e. by dividing by $p-1$).

The same applies in supervised learning (section 2). It would be a mistake to divide the sum-squared-training-error by the number of patterns in order to estimate the noise variance since some degrees of freedom will have been used up in fitting the model. In a linear model (section 3) there are m weights; if there are p patterns in the training set that leaves $p - m$ degrees of freedom.

The variance estimate is thus

$$\hat{\sigma}^2 \;=\; \frac{\hat{S}}{p-m}\,, \tag{4.9}$$

where $\hat{S}$ is the sum-squared-error over the training set at the optimal weight vector. $\hat{\sigma}^2$ is called the unbiased estimate of variance (equation 5.3).

However, things are not as simple when ridge regression is used. Although there are still m weights in the model, what John Moody calls the *effective number of parameters* [18] (and David MacKay calls the *number of good parameter measurements* [17]) is less than m and depends on the size of the regularisation parameter(s). The simplest formula for this number, γ, is

$$\gamma \;=\; p - \mathrm{trace}\,(\mathbf{P})\;, \tag{4.10}$$

which is consistent with both Moody's and MacKay's formulae, as shown in appendix A.8. The unbiased estimate of variance becomes

$$\hat{\sigma}^2 \;=\; \frac{\hat{S}}{p-\gamma}\,.$$

In the absence of regularisation $\mathrm{trace}\,(\mathbf{P}) = p - m$ and $\gamma = m$, as we would expect.

4.5 Example

Almost the simplest linear model for scalar inputs is the straight line given by

$$f(x) \;=\; w_1\, h_1(x) + w_2\, h_2(x)\,,$$

where the two basis functions are

$$h_1(x) \;=\; 1\,,$$

$$h_2(x) \;=\; x\,.$$

We stress this is unlikely ever to be used in a nonparametric context (section 2.1) as it obviously lacks the flexibility to fit a large range of different functions, but its simplicity lends itself to illustrating a point or two.

Assume we sample points from the curve $y = x$ at three points, $x_1 = 1$, $x_2 = 2$ and $x_3 = 3$, generating the training set

$$\{(1,\, 1.1),\, (2,\, 1.8),\, (3,\, 3.1)\}\,.$$

Clearly our sampling is noisy since the sampled output values do not correspond exactly with the expected values. Suppose, however, that the actual line from which we have sampled is unknown to us and that the only information we have is the three noisy samples. We estimate its coefficients (intercept w_1 and slope w_2) by solving the normal equation (section 4.1).

To do this we first set up the design matrix (equation 4.3) which is

$$\mathbf{H} \;=\; \begin{bmatrix} h_1(x_1) & h_2(x_1) \\ h_1(x_2) & h_2(x_2) \\ h_1(x_3) & h_2(x_3) \end{bmatrix} \;=\; \begin{bmatrix} 1 & 1 \\ 1 & 2 \\ 1 & 3 \end{bmatrix},$$

and the vector of training set (section 2) outputs which is

$$\hat{\mathbf{y}} \;=\; \begin{bmatrix} 1.1 \\ 1.8 \\ 3.1 \end{bmatrix}.$$

Next we compute the product of $\mathbf{H}$ with its own transpose, which is

$$\mathbf{H}^\top\mathbf{H} \;=\; \begin{bmatrix} 3 & 6 \\ 6 & 14 \end{bmatrix}.$$

and then the inverse of this square matrix to obtain

$$\mathbf{A}^{-1} \;=\; \left(\mathbf{H}^\top\mathbf{H}\right)^{-1} \;=\; \begin{bmatrix} 7/3 & -1 \\ -1 & 1/2 \end{bmatrix}.$$

You can check that this is correct by multiplying out $\mathbf{H}^\top\mathbf{H}$ and its inverse to obtain the identity matrix. Using this in equation 4.5 for the optimal weight vector yields

$$\hat{\mathbf{w}} \;=\; \mathbf{A}^{-1}\mathbf{H}^\top\mathbf{y} \;=\; \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

This means that the line with the least sum-squared-error with respect to the training set has slope 1 and intercept 0 (see figure 5).

Fortuitously, we have arrived at exactly the right answer since the actual line sampled did indeed have slope 1 and intercept 0. The projection matrix (appendix A.6) is

$$\mathbf{P} \;=\; \mathbf{I}_3 - \mathbf{H}\,\mathbf{A}^{-1}\mathbf{H} \;=\; \frac{1}{6}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix},$$

and consequently the sum-squared-error (equation 4.8) at the optimal weight is

$$\hat{\mathbf{y}}^\top\mathbf{P}^2\,\hat{\mathbf{y}} \;=\; 0.06\,.$$

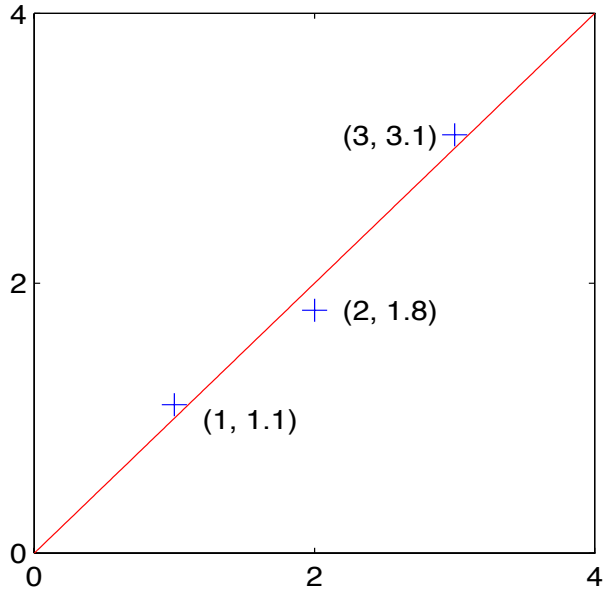

Figure 5: The least squares line fit to the three input-output pairs.

Another way of computing the same figure is

$$(\mathbf{H}\,\hat{\mathbf{w}} - \hat{\mathbf{y}})^\top (\mathbf{H}\,\hat{\mathbf{w}} - \hat{\mathbf{y}}) \;=\; 0.06\,.$$

Suppose, in ignorance of the true function, we look at figure 5 and decide that the model is wrong – that it should have an extra term, x^2. The new model is then

$$f(x) \;=\; w_1\,h_1(x) + w_2\,h_2(x) + w_3\,h_3(x)\,,$$

where

$$h_3(x) \;=\; x^2\,.$$

The effect of the extra basis function is to add an extra column,

$$\mathbf{h}_3 \;=\; \begin{bmatrix} h_3(x_1) \\ h_3(x_2) \\ h_3(x_3) \end{bmatrix} \;=\; \begin{bmatrix} x_1^2 \\ x_2^2 \\ x_3^2 \end{bmatrix} \;=\; \begin{bmatrix} 1 \\ 4 \\ 9 \end{bmatrix}\,,$$

to the design matrix. We could retrain the network from scratch, adding the extra column to $\mathbf{H}$, recomputing $\mathbf{H}^\top\mathbf{H}$, its inverse $\mathbf{A}^{-1}$ and the new projection matrix. The cost, in floating point operations, is given in table 2. Alternatively, we could retrain as an incremental operation, calculating the new variance matrix from equation A.4 and the new projection matrix from equation A.5. This results in a reduction in the amount of computation (see table 2), a reduction which for larger values of p and m can be quite significant (see table 3 in appendix A.7).

Figure 6 shows the fit made by the model with the extra basis function. The curve passes through each training point and there is consequently no training set

	retrain from scratch	incremental operation
$\mathbf{A}^{-1}$	156	71
$\mathbf{P}$	117	30

Table 2: The cost in FLOPS of retraining from scratch and retraining by using an incremental operation after adding the extra x^2 term to the model in the example.

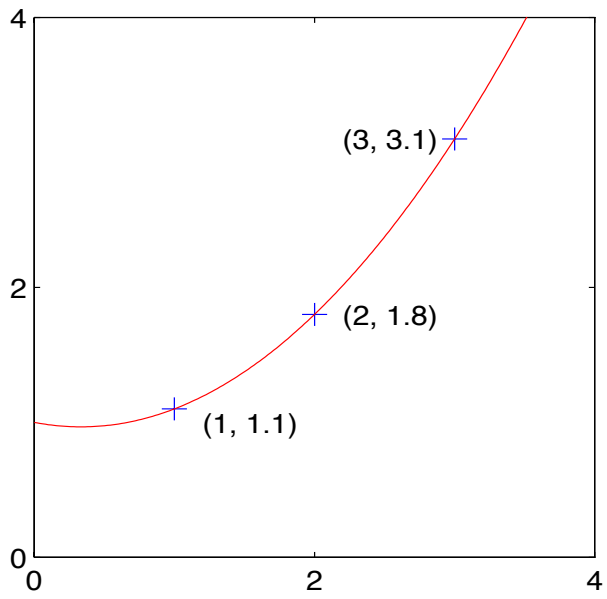

Figure 6: The least squares quadratic fit to the three input-output pairs.

error. The extra basis function has made the model flexible enough to absorb all the noise. If we did not already know that the target was a straight line we might now be fooled into thinking that the new model was better than the previous one. This is a simple demonstration of the dilemma of supervised learning (section 2) – if the model is too flexible it will fit the noise, if it is too inflexible it will miss the target. Somehow we have to tread a path between these two pitfalls and this is the reason for employing model selection criteria (section 5) which we look at next.

5 Model Selection Criteria

The model selection criteria we shall cover are all estimates of prediction error, that is, estimates of how well the trained model will perform on future (unknown) inputs. The best model is the one whose estimated prediction error is least. Cross-validation (section 5.1) is the standard tool for measuring prediction error but there are several others, exemplified by generalised cross-validation (section 5.2), which all involve various adjustments to the sum-squared-error (equation 4.8) over the training set.

The key elements in these criteria are the projection matrix and the effective number of parameters in the network. In ridge regression (section 6) the projection matrix, although we continue to call it that, is not exactly a projection and the effective number of parameters is not equal to m, the number of weights. To be as general as possible we shall use ridge versions for both the projection matrix (equation 4.6) and the effective number of parameters (equation 4.10). However, the ordinary least squares versions of the selection criteria can always be obtained simply by setting all the regularisation parameters to zero and remembering that the projection matrix is idempotent ($\mathbf{P}^2 = \mathbf{P}$).

For reviews of model selection see the two articles by Hocking [9, 10] and chapter 17 of Efron and Tisbshirani's book [14].

5.1 Cross-Validation

If data is not scarce then the set of available input-output measurements can be divided into two parts – one part for training and one part for testing. In this way several different models, all trained on the training set, can be compared on the test set. This is the basic form of cross-validation.

A better method, which is intended to avoid the possible bias introduced by relying on any one particular division into test and train components, is to partition the original set in several different ways and to compute an average score over the different partitions. An extreme variant of this is to split the p patterns into a training set of size $p-1$ and a test of size 1 and average the squared error on the left-out pattern over the p possible ways of obtaining such a partition [1]. This is called leave-one-out (LOO) cross-validation. The advantage is that all the data can be used for training – none has to be held back in a separate test set.

The beauty of LOO for linear models (equation 3.1) such as RBF networks (section 3.2) is that it can be calculated analytically (appendix A.9) as

$$\hat{\sigma}^2_{\mathrm{LOO}} \;=\; \frac{\hat{\mathbf{y}}^\top \mathbf{P}\,(\mathrm{diag}\,(\mathbf{P}))^{-2}\,\mathbf{P}\,\hat{\mathbf{y}}}{p}\,. \tag{5.1}$$

In contrast, LOO is intractable to calculate for all but the smallest nonlinear models as there is no alternative to training and testing p times.

5.2 Generalised Cross-Validation

The matrix $\mathrm{diag}\,(\mathbf{P})$ makes LOO slightly awkward to handle mathematically. Its cousin, *generalised cross-validation* (GCV) [6], is more convenient and is

$$\hat{\sigma}^2_{\mathrm{GCV}} \;=\; \frac{p\,\hat{\mathbf{y}}^\top \mathbf{P}^2\,\hat{\mathbf{y}}}{\left(\mathrm{trace}\,(\mathbf{P})\right)^2}\,. \tag{5.2}$$

The similarity with leave-one-out cross-validation (equation 5.1) is apparent. Just replace $\mathrm{diag}\,(\mathbf{P})$ in the equation for LOO with the average diagonal element times the identity matrix, $(\mathrm{trace}\,(\mathbf{P})\,/p)\,\mathbf{I}_p$.

GCV is one of a number of criteria which all involve an adjustment to the average mean-squared-error over the training set (equation 4.8). There are several other criteria which share this form [14]. The *unbiased estimate of variance* (UEV), which we met in a previous section (4.4), is

$$\hat{\sigma}^2_{\mathrm{UEV}} \;=\; \frac{\hat{\mathbf{y}}^\top \mathbf{P}^2\,\hat{\mathbf{y}}}{p-\gamma}\,, \tag{5.3}$$

where γ is the effective number of parameters (equation 4.10). A version of Mallow's C_p [22] and also a special case of Akaike's information criterion is *final prediction error* (FPE)

$$\begin{aligned}\hat{\sigma}^2_{\mathrm{FPE}} \;&=\; \frac{1}{p}\left(\hat{\mathbf{y}}^\top \mathbf{P}^2\,\hat{\mathbf{y}} + 2\,\gamma\,\hat{\sigma}^2_{\mathrm{UEV}}\right)\\ &=\; \frac{p+\gamma}{p-\gamma}\,\frac{\hat{\mathbf{y}}^\top \mathbf{P}^2\,\hat{\mathbf{y}}}{p}\,.\end{aligned} \tag{5.4}$$

Schwarz's *Bayesian information criterion* (BIC) [28] is

$$\begin{aligned}\hat{\sigma}^2_{\mathrm{BIC}} \;&=\; \frac{1}{p}\left(\hat{\mathbf{y}}^\top \mathbf{P}^2\,\hat{\mathbf{y}} + \ln(p)\,\gamma\,\hat{\sigma}^2_{\mathrm{UEV}}\right)\\ &=\; \frac{p+(\ln(p)-1)\,\gamma}{p-\gamma}\,\frac{\hat{\mathbf{y}}^\top \mathbf{P}^2\,\hat{\mathbf{y}}}{p}\,.\end{aligned} \tag{5.5}$$

Generalised-cross validation can also be written in terms of γ instead of $\mathrm{trace}\,(\mathbf{P})$, using the equation for the effective number of parameters (4.10).

$$\hat{\sigma}^2_{\mathrm{GCV}} \;=\; \frac{p\,\hat{\mathbf{y}}^\top \mathbf{P}^2\,\hat{\mathbf{y}}}{(p-\gamma)^2}\,. \tag{5.6}$$

UEV, FPE, GCV and BIC are all in the form $\hat{\sigma}^2_{\mathrm{XYZ}} = \Gamma_{\mathrm{XYZ}}\, \hat{\mathbf{y}}^\top \mathbf{P}^2\, \hat{\mathbf{y}}/p$ and have a natural ordering,

$$\Gamma_{\mathrm{UEV}} \le \Gamma_{\mathrm{FPE}} \le \Gamma_{\mathrm{GCV}} \le \Gamma_{\mathrm{BIC}}\,,$$

as is plain if the Γ factors are expanded out in Taylor series.

$$\begin{aligned}
\frac{p}{p-\gamma} &= \Gamma_{\mathrm{UEV}} = 1 + \frac{\gamma}{p} + \frac{\gamma^2}{p} + \frac{\gamma^3}{p} + \cdots \\
\frac{p+\gamma}{p-\gamma} &= \Gamma_{\mathrm{FPE}} = 1 + \frac{2\,\gamma}{p} + \frac{2\,\gamma^2}{p^2} + \frac{2\,\gamma^3}{p^3} + \cdots \\
\frac{p^2}{(p-\gamma)^2} &= \Gamma_{\mathrm{GCV}} = 1 + \frac{2\,\gamma}{p} + \frac{3\,\gamma^2}{p^2} + \frac{4\,\gamma^3}{p^3} + \cdots \\
\frac{p + (\ln(p) - 1)\,\gamma}{p-\gamma} &= \Gamma_{\mathrm{BIC}} = 1 + \ln(p)\left(\frac{\gamma}{p} + \frac{\gamma^2}{p^2} + \frac{\gamma^3}{p^3} + \cdots\right)
\end{aligned}$$

5.3 Example

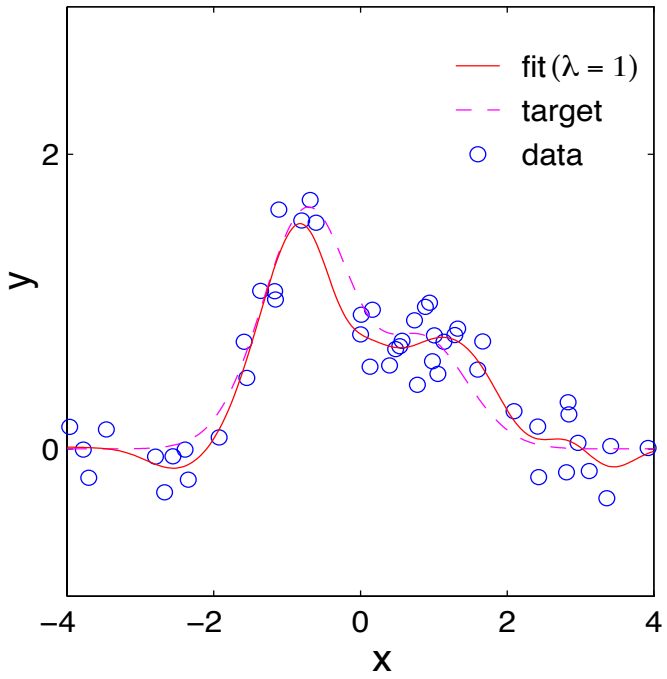

Figure 7: Training set, target function and fit for the example.

Figure 7 shows a set of $p = 50$ training input-output points sampled from the function

$$y(x) = (1 + x - 2\,x^2)\, e^{-x^2}$$

with added Gaussian noise of standard deviation $\sigma = 0.2$. We shall fit this data with an RBF network (section 3.2) with $m = 50$ Gaussian centres (section 3.1) coincident with the input points in the training set and of width $r = 0.5$. To avoid overfit we shall use standard ridge regression (section 6.1) controlled by a single regularisation parameter λ. The error predictions, as a function of λ, made by the various model selection criteria are shown in figure 8. Also shown is the mean-squared-error (MSE) between the fit and the target over an independent test set.

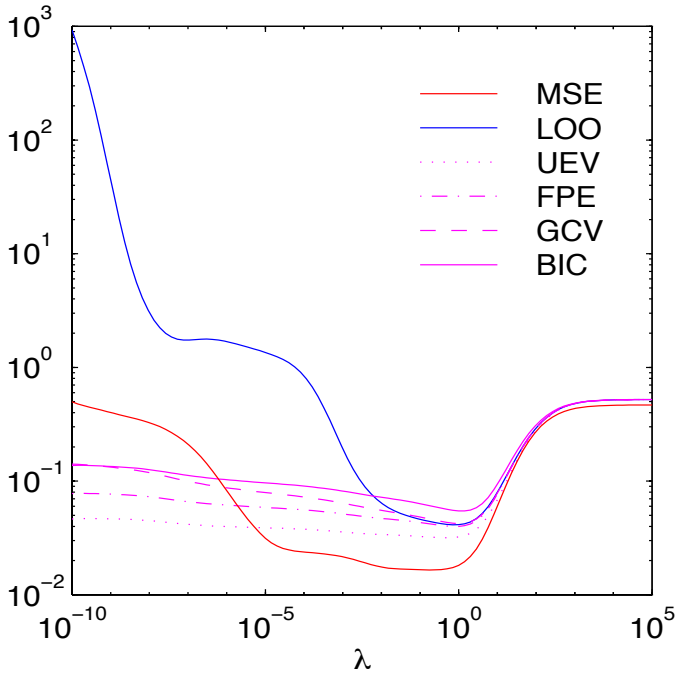

Figure 8: The various model selection criteria, plus mean-squared-error over a test set, as a function of the regularisation parameter.

A feature of the figure is that all the criteria have a minimum which is close to the minimum MSE, near $\lambda = 1$. This is why the model selection criteria are useful. When we do not have access to the true MSE, as in any practical problem, they are often able to select the best model (in this example, the best λ) by picking out the one with the lowest predicted error.

Two words of warning: they don't always work as effectively as in this example and UEV is inferior to GCV as a selection criteria [20] (and probably to the others as well).

6 Ridge Regression

Around the middle of the 20th century the Russian theoretician Andre Tikhonov was working on the solution of *ill-posed problems*. These are mathematical problems for which no unique solution exists because, in effect, there is not enough information specified in the problem. It is necessary to supply extra information (or assumptions) and the mathematical technique Tikhonov developed for this is known as regularisation.

Tikhonov's work only became widely known in the West after the publication in 1977 of his book [29]. Meanwhile, two American statisticians, Arthur Hoerl and Robert Kennard, published a paper in 1970 [11] on ridge regression, a method for solving badly conditioned linear regression problems. Bad conditioning means numerical difficulties in performing the matrix inverse necessary to obtain the variance matrix (equation 4.4). It is also a symptom of an ill-posed regression problem in Tikhonov's sense and Hoerl & Kennard's method was in fact a crude form of regularisation, known now as zero-order regularisation [25].

In the 1980's, when neural networks became popular, weight decay was one of a number of techniques 'invented' to help prune unimportant network connections. However, it was soon recognised [8] that weight decay involves adding the same penalty term to the sum-squared-error as in ridge regression. Weight-decay and ridge regression are equivalent.

While it is admittedly crude, I like ridge regression because it is mathematically and computationally convenient and consequently other forms of regularisation are rather ignored here. If the reader is interested in higher-order regularisation I suggest looking at [25] for a general overview and [16] for a specific example (second-order regularisation in RBF networks).

We next describe ridge regression from the perspective of bias and variance (section 6.1) and how it affects the equations for the optimal weight vector (appendix A.4), the variance matrix (appendix A.5) and the projection matrix (appendix A.6). A method to select a good value for the regularisation parameter, based on a re-estimation formula (section 6.2), is then presented. Next comes a generalisation of ridge regression which, if radial basis functions (section 3.1) are used, can be justly called local ridge regression (section 6.3). It involves multiple regularisation parameters and we describe a method (section 6.4) for their optimisation. Finally, we illustrate with a simple example (section 6.5).

6.1 Bias and Variance

When the input is $\mathbf{x}$ the trained model predicts the output as $f(\mathbf{x})$. If we had many training sets (which we never do but just suppose) and if we knew the true output, $y(\mathbf{x})$, we could calculate the mean-squared-error as

$$\text{MSE} \;=\; \left\langle (y(\mathbf{x}) - f(\mathbf{x}))^2 \right\rangle \;,$$

where the expectation (averaging) indicated by $\langle \ldots \rangle$ is taken over the training sets. This score, which tells us how good the average prediction is, can be broken down into two components [5], namely

$$\text{MSE} \;=\; \left(y(\mathbf{x}) - \langle f(\mathbf{x}) \rangle\right)^2 + \left\langle \left(f(\mathbf{x}) - \langle f(\mathbf{x}) \rangle\right)^2 \right\rangle \,.$$

The first part is the *bias* and the second part is the *variance*.

If $\langle f(\mathbf{x}) \rangle = y(\mathbf{x})$ for all $\mathbf{x}$ then the model is unbiased (the bias is zero). However, an unbiased model may still have a large mean-squared-error if it has a large variance. This will be the case if $f(\mathbf{x})$ is highly sensitive to the peculiarities (such as noise and the choice of sample points) of each particular training set and it is this sensitivity which causes regression problems to be ill-posed in the Tikhonov [29] sense. Often, however, the variance can be significantly reduced by deliberately introducing a small amount of bias so that the net effect is a reduction in mean-squared-error.

Introducing bias is equivalent to restricting the range of functions for which a model can account. Typically this is achieved by removing degrees of freedom. Examples would be lowering the order of a polynomial or reducing the number of weights in a neural network. Ridge regression does not explicitly remove degrees of freedom but instead reduces the effective number of parameters (section 4.4). The resulting loss of flexibility makes the model less sensitive.

A convenient, if somewhat arbitrary, method of restricting the flexibility of linear models (section 3) is to augment the sum-squared-error (equation 4.1) with a term which penalises large weights,

$$C \;=\; \sum_{i=1}^{p} \left(\hat{y}_i - f(\mathbf{x}_i)\right)^2 + \lambda \sum_{j=1}^{m} w_j^2 \,. \tag{6.1}$$

This is ridge regression (weight decay) and the regularisation parameter $\lambda > 0$ controls the balance between fitting the data and avoiding the penalty. A small value for λ means the data can be fit tightly without causing a large penalty; a large value for λ means a tight fit has to be sacrificed if it requires large weights. The bias introduced favours solutions involving small weights and the effect is to smooth the output function since large weights are usually required to produce a highly variable (rough) output function.

The optimal weight vector for the above cost function (equation 6.1) has already been dealt with (appendix A.4), as have the variance matrix (appendix A.5) and the projection matrix (appendix A.6). In summary,

$$\mathbf{A} \;=\; \mathbf{H}^\top \mathbf{H} + \lambda \, \mathbf{I}_m \,, \tag{6.2}$$

$$\hat{\mathbf{w}} \;=\; \mathbf{A}^{-1} \mathbf{H}^\top \hat{\mathbf{y}} \,, \tag{6.3}$$

$$\mathbf{P} \;=\; \mathbf{I}_p - \mathbf{H} \, \mathbf{A}^{-1} \mathbf{H}^\top \,. \tag{6.4}$$

6.2 Optimising the Regularisation Parameter

Some sort of model selection (section 5) must be used to choose a value for the regularisation parameter λ. The value chosen is the one associated with the lowest prediction error. But which method should be used to predict the error and how is the optimal value found?

The answer to the first question is that nobody knows for sure. The popular choices are leave-one-out cross-validation, generalised cross-validation, final prediction error and Bayesian information criterion. Then there are also bootstrap methods [14]. Our approach will be to use the most convenient method, as long as there are no serious objections to it, and the most convenient method is generalised cross validation (GCV) [6]. It leads to the simplest optimisation formulae, especially in local optimisation (section 6.4).

Since all the model selection criteria depend nonlinearly on λ we need a method of nonlinear optimisation. We could use any of the standard techniques for this, such as the Newton method, and in fact that has been done [7]. Alternatively [20], we can exploit the fact that when the derivative of the GCV error prediction is set to zero, the resulting equation can be manipulated so that only $\hat{\lambda}$ appears on the left hand side (see appendix A.10),

$$\hat{\lambda} \;=\; \frac{\hat{\mathbf{y}}^\top \mathbf{P}^2\, \hat{\mathbf{y}}\, \mathrm{trace}\left(\mathbf{A}^{-1} - \hat{\lambda}\, \mathbf{A}^{-2}\right)}{\hat{\mathbf{w}}^\top \mathbf{A}^{-1}\, \hat{\mathbf{w}}\, \mathrm{trace}\left(\mathbf{P}\right)} \,. \tag{6.5}$$

This is not a solution, it is a re-estimation formula because the right hand side depends on $\hat{\lambda}$ (explicitly as well as implicitly through $\mathbf{A}^{-1}$ and $\mathbf{P}$). To use it, an initial value of $\hat{\lambda}$ is chosen and used to calculate a value for the right hand side. This leads to a new estimate and the process can be repeated until convergence.

6.3 Local Ridge Regression

Instead of treating all weights equally with the penalty term $\lambda \sum_{j=1}^{m} w_j^2$ we can treat them all separately and have a regularisation parameter associated with each basis function by using $\sum_{j=1}^{m} \lambda_j\, w_j^2$. The new cost function is

$$C \;=\; \sum_{i=1}^{p} \left(\hat{y}_i - f(\mathbf{x}_i)\right)^2 + \sum_{j=1}^{m} \lambda_j\, w_j^2 \,, \tag{6.6}$$

and is identical to the the standard form (equation 6.1) if the regularisation parameters are all equal ($\lambda_j = \lambda$). Then the variance matrix (appendix A.5) is

$$\mathbf{A}^{-1} \;=\; \left(\mathbf{H}^\top \mathbf{H} + \mathbf{\Lambda}\right)^{-1} \,, \tag{6.7}$$

where $\mathbf{\Lambda}$ is a diagonal matrix with the regularisation parameters, $\{\lambda\}_{j=1}^{m}$, along its diagonal. As usual, the optimal weight vector is

$$\hat{\mathbf{w}} \;=\; \mathbf{A}^{-1}\mathbf{H}^{\top}\hat{\mathbf{y}}\,. \tag{6.8}$$

and the projection matrix is

$$\mathbf{P} \;=\; \mathbf{I}_p - \mathbf{H}\,\mathbf{A}^{-1}\mathbf{H}^{\top}\,. \tag{6.9}$$

These formulae are identical to standard ridge regression (section 6.1) except for the variance matrix where $\lambda\mathbf{I}_m$ has been replaced by $\mathbf{\Lambda}$.

In general there is nothing local about this form of weight decay. However, if we confine ourselves to local basis functions such as radial functions (section 3.1) (but not the multiquadric type which are seldom used in practice) then the smoothness produced by this form of ridge regression is controlled in a local fashion by the individual regularisation parameters. That is why we call it local ridge regression and it provides a mechanism to adapt smoothness to local conditions. Standard ridge regression, with just one parameter, λ, to control the bias/variance trade-off, has difficulty with functions which have significantly different smoothness in different parts of the input space.

6.4 Optimising the Regularisation Parameters

To take advantage of the adaptive smoothing capability provided by local ridge regression requires the use of model selection criteria (section 5) to optimise the regularisation parameters. Information about how much smoothing to apply in different parts of the input space may be present in the data and the model selection criteria can help to extract it.

The selection criteria depend mainly on the projection matrix (equation 6.9) $\mathbf{P}$ and therefore we need to deduce its dependence on the individual regularisation parameters. The relevant relationship (equation A.6) is one of the incremental operations (section 4.3). Adapting the notation somewhat, it is

$$\mathbf{P} \;=\; \mathbf{P}_j - \frac{\mathbf{P}_j\,\mathbf{h}_j\,\mathbf{h}_j^{\top}\mathbf{P}_j}{\lambda_j + \mathbf{h}_j^{\top}\mathbf{P}_j\,\mathbf{h}_j}\,, \tag{6.10}$$

where $\mathbf{P}_j$ is the projection matrix after the j-th basis function has been removed and $\mathbf{h}_j$ is the j-th column of the design matrix (equation 4.3).

In contrast to the case of standard ridge regression (section 6.2), there is an analytic solution for the optimal value of λ_j based on GCV minimisation [19] – no re-estimation is necessary (see appendix A.11). The trouble is that there are $m-1$ other parameters to optimise and each time one λ_j is optimised it changes

the optimal value of each of the others. Optimising all the parameters together has to be done as a kind of re-estimation, doing one at a time and then repeating until they all converge [19].

When $\lambda_j = \infty$ the two projection matrices, $\mathbf{P}$ and $\mathbf{P}_j$ are equal. This means that if the optimal value of λ_j is ∞ then the j-th basis function can be removed from the network. In practice, especially if the network is initially very flexible (high variance, low bias – see section 6.1) infinite optimal values are very common and local ridge regression can be used as a method of pruning unnecessary hidden units.

The algorithm can get stuck in local minima, like any other nonlinear optimisation, depending on the initial settings. For this reason it is best in practice to give the algorithm a head start by using the results from other methods rather than starting with random parameters. For example, standard ridge regression (section 6.1) can be used to find the best global parameter, $\hat{\lambda}$, and the local algorithm can then start from

$$\hat{\lambda}_j = \hat{\lambda}\,, \quad 1 \le j \le m\,.$$

Alternatively, forward selection (section 7) can be used to choose a subset, $\mathcal{S}$, of the original m basis functions in which case the local algorithm can start from

$$\hat{\lambda}_j \;=\; \begin{cases} 0 & \text{if } j \in \mathcal{S} \\ \infty & \text{otherwise} \end{cases} \quad .$$

6.5 Example

Figure 9 shows four different fits (the red curves) to a training set of $p = 50$ patterns randomly sampled from the the sine wave

$$y = \sin(12\,x)$$

between $x = 0$ and $x = 1$ with Gaussian noise of standard deviation $\sigma = 0.1$ added. The training set input-output pairs are shown by blue circles and the true target by dashed magenta curves. The model used is a radial basis function network (section 3.2) with $m = 50$ Gaussian functions of width $r = 0.05$ whose positions coincide with the training set input points. Each fit uses standard ridge regression (section 6.1) but with four different values of the regularisation parameter λ.

The first fit (top left) is for $\lambda = 1 \times 10^{-10}$ and is too rough – high weights have not been penalised enough. The last fit (bottom right) is for $\lambda = 1 \times 10^{5}$ and is too smooth – high weights have been penalised too much. The other two fits are for $\lambda = 1 \times 10^{-5}$ (top right) and $\lambda = 1$ (bottom left) which are just about right, there is not much to choose between them.

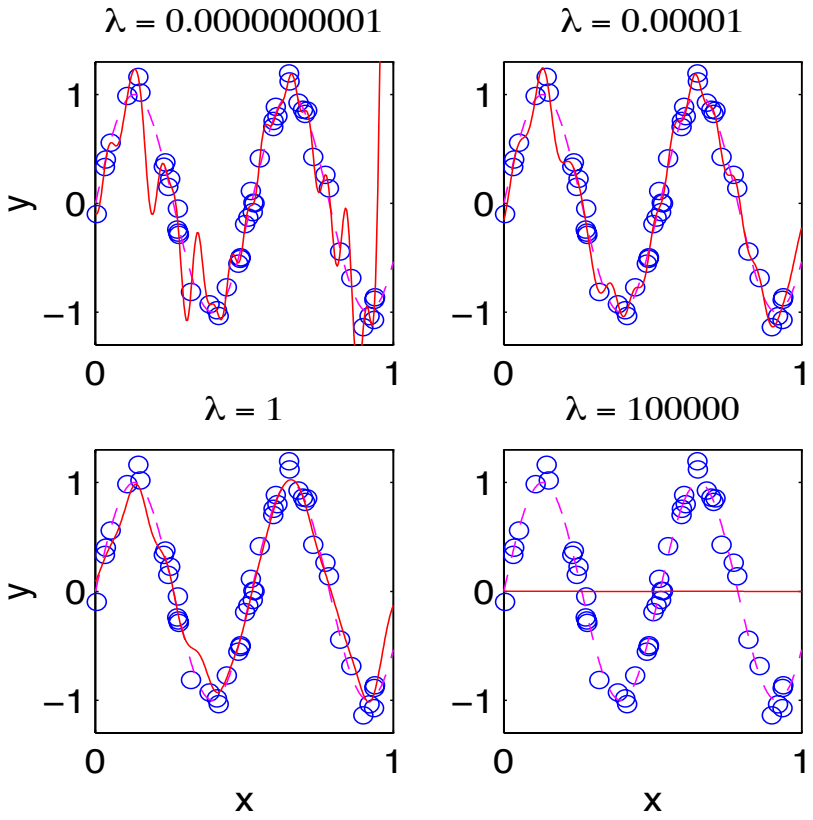

Figure 9: Four different RBF fits (solid red curves) to data (blue crosses) sampled from a sine wave (dashed magenta curve).

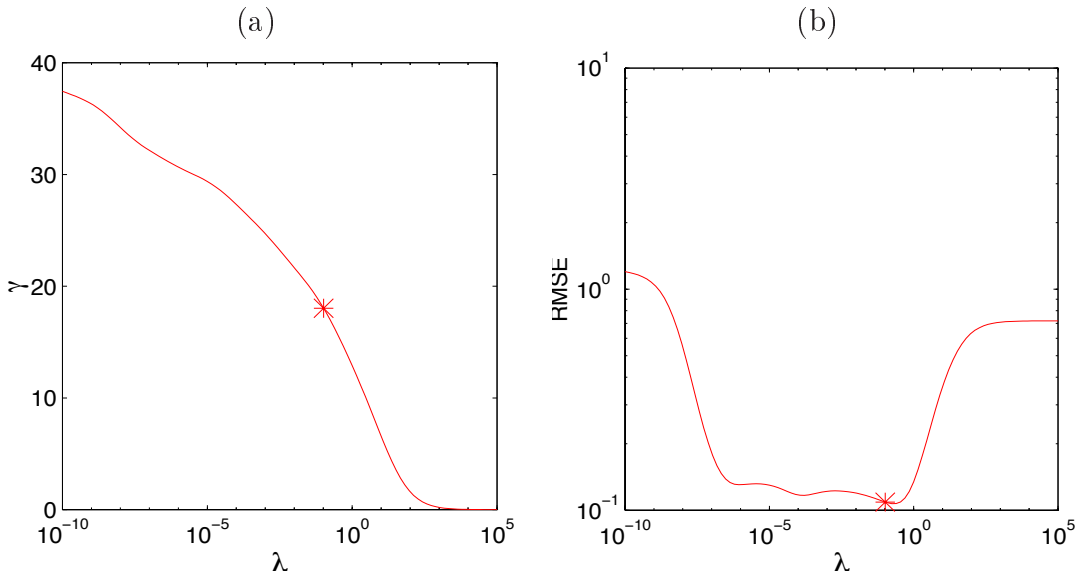

Figure 10: (a) γ and (b) RMSE as functions of λ. The optimal value is shown with a star.

The variation of the effective number of parameters (section 4.10), γ, as a function of λ is shown in figure 10(a). Clearly γ decreases monotonically as λ increases and the RBF network loses flexibility. Figure 10(b) shows root-mean-squared-error (RMSE) as a function of λ. RMSE was calculated using an array of 250 noiseless samples of the target between $x = 0$ and $x = 1$. The figure suggests $\lambda \approx 0.1$ as the best value (minimum RMSE) for the regularisation parameter.

In real applications where the target is, of course, unknown we do not, unfortunately, have access to RMSE. Then we must use one of the model selection criteria (section 5) to find parameters like λ. The solid red in figure 11 shows the variation of GCV over a range of λ values. The re-estimation formula (equation 6.5) based on GCV gives $\hat{\lambda} = 0.10$ starting from the initial value of $\hat{\lambda} = 0.01$. Note, however, that had we started the re-estimation at $\hat{\lambda} = 10^{-5}$ then the local minima, at $\hat{\lambda} = 2.1 \times 10^{-4}$, would have been the final resting place. The values of γ and RMSE at the optimum, $\hat{\lambda} = 0.1$, are marked with stars in figure 10.

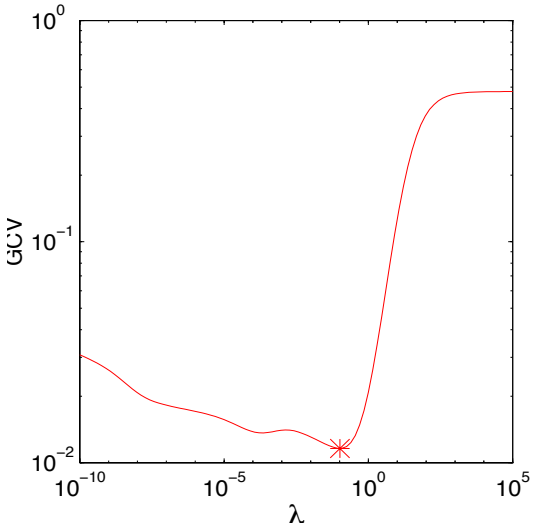

Figure 11: GCV as functions of λ with the optimal value shown with a star.

The values $\hat{\lambda}_j = 0.1$ were used to initialise a local ridge regression algorithm which continually sweeps through the regularisation parameters optimising each in turn until the GCV error prediction converges. This algorithm reduced the error prediction from an initial value of $\hat{\sigma}^2 = 0.016$ at the optimal global parameter value to $\hat{\sigma}^2 = 0.009$. At the same time 18 regularisation parameters ended up with an optimal value of ∞ enabling the 18 corresponding hidden units to be pruned from the network.

7 Forward Selection

We previously looked at ridge regression (section 6) as a means of controlling the balance between bias and variance (section 6.1) by varying the effective number of parameters (section 4.4) in a linear model (section 3) of fixed size.

An alternative strategy is to compare models made up of different subsets of basis functions drawn from the same fixed set of candidates. This is called *subset selection* in statistics [26]. To find the best subset is usually intractable — there are $2^M - 1$ subsets in a set of size M — so heuristics must be used to search a small but hopefully interesting fraction of the space of all subsets. One of these heuristics is called *forward selection* which starts with an empty subset to which is added one basis function at a time — the one which most reduces the sum-squared-error (equation 4.1) — until some chosen criterion, such as GCV (section 5.2), stops decreasing. Another is *backward elimination* which starts with the full subset from which is removed one basis function at a time — the one which least increases the sum-squared-error — until, once again, the chosen criterion stops decreasing.

It is interesting to compare subset selection with the standard way of optimising neural networks. The latter involves the optimisation, by gradient descent, of a nonlinear sum-squared-error surface in a high-dimensional space defined by the network parameters [8]. In RBF networks (section 3.1) the network parameters are the centres, sizes and hidden-to-output weights. In subset selection the optimisation algorithm searches in a discrete space of subsets of a set of hidden units with fixed centres and sizes and tries to find the subset with the lowest prediction error (section 5). The hidden-to-output weights are not selected, they are slaved to the centres and sizes of the chosen subset. Forward selection is also, of course, a nonlinear type of algorithm but it has the following advantages:

- There is no need to fix the number of hidden units in advance.
- The model selection criteria are tractable.
- The computational requirements are relatively low.

In forward selection each step involves growing the network by one basis function. Adding a new basis function is one of the incremental operations (section 4.3). The key equation (A.5) is

$$\mathbf{P}_{m+1} \;=\; \mathbf{P}_m - \frac{\mathbf{P}_m \mathbf{f}_J \mathbf{f}_J^\top \mathbf{P}_m}{\mathbf{f}_J^\top \mathbf{P}_m \mathbf{f}_J}\,, \tag{7.1}$$

which expresses the relationship between $\mathbf{P}_m$, the projection matrix (equation 4.6) for the m hidden units in the current subset, and $\mathbf{P}_{m+1}$, the succeeding projection matrix if the J-th member of the full set is added. The vectors $\{\mathbf{f}_J\}_{J=1}^M$ are the

columns of the design matrix (equation 4.3) for the entire set of candidate basis functions,

$$\mathbf{F} \;=\; [\mathbf{f}_1\ \mathbf{f}_2\ \ldots\ \mathbf{f}_M]\,,$$

of which there are M (where $M \gg m$).

If the J-th basis function is chosen then $\mathbf{f}_J$ is appended to the last column of $\mathbf{H}_m$, the design matrix of the current subset. This column is renamed $\mathbf{h}_{m+1}$ and the new design matrix is $\mathbf{H}_{m+1}$. The choice of basis function can be based on finding the greatest decrease in sum-squared-error (equation 4.1). From the update rule (equation 7.1) for the projection matrix and equation 4.8 for sum-squared-error

$$\hat{S}_m - \hat{S}_{m+1} \;=\; \frac{(\hat{\mathbf{y}}^\top \mathbf{P}_m \mathbf{f}_J)^2}{\mathbf{f}_J^\top \mathbf{P}_m \mathbf{f}_J}\,. \tag{7.2}$$

(see appendix A.12). The maximum (over $1 \le J \le M$) of this difference is used to find the best basis function to add to the current network.

To decide when to stop adding further basis functions one of the model selection criteria (section 5), such as GCV (equation 5.2), is monitored. Although the training error, $\hat{S}$, will never increase as extra functions are added, GCV will eventually stop decreasing and start to increase as overfit (section 6.1) sets in. That is the point at which to cease adding to the network. See section 7.4 for an illustration.

7.1 Orthogonal Least Squares

Forward selection is a relatively fast algorithm but it can be speeded up even further using a technique called *orthogonal least squares* [4]. This is a Gram-Schmidt orthogonalisation process [12] which ensures that each new column added to the design matrix of the growing subset is orthogonal to all previous columns. This simplifies the equation for the change in sum-squared-error and results in a more efficient algorithm.

Any matrix can be factored into the product of a matrix with orthogonal columns and a matrix which is upper triangular. In particular, the design matrix (equation 4.3), $\mathbf{H}_m \in \mathcal{R}^{p\times m}$, can be factored into

$$\mathbf{H}_m \;=\; \tilde{\mathbf{H}}_m \mathbf{U}_m\,, \tag{7.3}$$

where

$$\tilde{\mathbf{H}}_m \;=\; [\tilde{\mathbf{h}}_1\ \tilde{\mathbf{h}}_2\ \ldots\ \tilde{\mathbf{h}}_m] \;\in\; \mathcal{R}^{p\times m}$$

has orthogonal columns ($\tilde{\mathbf{h}}_i^\top \tilde{\mathbf{h}}_j = 0,\ i \ne j$) and $\mathbf{U}_m \in \mathcal{R}^{m\times m}$ is upper triangular.

When considering whether to add the basis function corresponding to the J-th column, $\mathbf{f}_J$, of the full design matrix the projection of $\mathbf{f}_J$ in the space already spanned by the m columns of the current design matrix is irrelevant. Only its projection perpendicular to this space, namely

$$\tilde{\mathbf{f}}_J \;=\; \mathbf{f}_J - \sum_{j=1}^{m} \frac{\mathbf{f}_J^\top \tilde{\mathbf{h}}_j}{\tilde{\mathbf{h}}_j^\top \tilde{\mathbf{h}}_j}\, \tilde{\mathbf{h}}_j \,,$$

can contribute to a further reduction in the training error, and this reduction is

$$\hat{S}_m - \hat{S}_{m+1} \;=\; \frac{(\hat{\mathbf{y}}^\top \tilde{\mathbf{f}}_J)^2}{\tilde{\mathbf{f}}_J^\top \tilde{\mathbf{f}}_J} \,, \tag{7.4}$$

as shown in appendix A.13. Computing this change in sum-squared-error requires of order p floating point operations, compared with p^2 for the unnormalised version (equation 7.2). This is the basis of the increased efficiency of orthogonal least squares.

A small overhead is necessary to maintain the columns of the full design matrix orthogonal to the space spanned by the columns of the growing design matrix and to update the upper triangular matrix. After $\tilde{\mathbf{f}}_J$ is selected the new orthogonalised full design matrix is

$$\tilde{\mathbf{F}}_{m+1} \;=\; \tilde{\mathbf{F}}_m - \frac{\tilde{\mathbf{f}}_J \, \tilde{\mathbf{f}}_J^\top \tilde{\mathbf{F}}_m}{\tilde{\mathbf{f}}_J^\top \tilde{\mathbf{f}}_J} \,,$$

and the upper triangular matrix is updated to

$$\mathbf{U}_m \;=\; \begin{bmatrix} \mathbf{U}_{m-1} & (\tilde{\mathbf{H}}_{m-1}^\top \tilde{\mathbf{H}}_{m-1})^{-1} \tilde{\mathbf{H}}_{m-1}^\top \mathbf{f}_J \\ \mathbf{0}_{m-1}^\top & 1 \end{bmatrix} .$$

Initially $\mathbf{U}_1 = 1$ and $\tilde{\mathbf{F}}_0 = \mathbf{F}$. The orthogonalised optimal weight vector,

$$\tilde{\mathbf{w}}_m \;=\; \left(\tilde{\mathbf{H}}_m^\top \tilde{\mathbf{H}}_m\right)^{-1} \mathbf{H}_m^\top \hat{\mathbf{y}} \,,$$

and the unorthogonalised optimal weight (equation 4.5) are then related by

$$\hat{\mathbf{w}}_m \;=\; \mathbf{U}_m^{-1}\, \tilde{\mathbf{w}}_m \,,$$

(see appendix A.13).

7.2 Regularised Forward Selection

Forward selection and standard ridge regression (section 6.1) can be combined, leading to a modest improvement in performance [20]. In this case the key equation (7.5) involves the regularisation parameter λ.

$$\mathbf{P}_{m+1} \;=\; \mathbf{P}_m - \frac{\mathbf{P}_m \mathbf{f}_J \mathbf{f}_J^\top \mathbf{P}_m}{\lambda + \mathbf{f}_J^\top \mathbf{P}_m \mathbf{f}_J}\,, \tag{7.5}$$

The search for the maximum decrease in the sum-squared-error (equation 4.1) is then based on

$$\hat{S}_m - \hat{S}_{m+1} \;=\; \frac{2\,\hat{\mathbf{y}}^\top \mathbf{P}_m^2 \mathbf{f}_J\, \hat{\mathbf{y}}^\top \mathbf{P}_m \mathbf{f}_J}{\lambda + \mathbf{f}_J^\top \mathbf{P}_m \mathbf{f}_J} - \frac{(\hat{\mathbf{y}}^\top \mathbf{P}_m \mathbf{f}_J)^2\, \mathbf{f}_J^\top \mathbf{P}_m^2 \mathbf{f}_J}{(\lambda + \mathbf{f}_J^\top \mathbf{P}_m \mathbf{f}_J)^2} \tag{7.6}$$

(see appendix A.14). Alternatively, it is possible to search for the maximum decrease in the cost function (equation 6.1)

$$\hat{C}_m - \hat{C}_{m+1} \;=\; \frac{(\hat{\mathbf{y}}^\top \mathbf{P}_m \mathbf{f}_J)^2}{\lambda + \mathbf{f}_J^\top \mathbf{P}_m \mathbf{f}_J}\,. \tag{7.7}$$

(see appendix A.14).

The advantage of ridge regression is that the regularisation parameter can be optimised (section 6.2) in between the addition of new basis functions. Since new additions will cause a change in the optimal value anyway, there is little point in running the re-estimation formula (equation 6.5) to convergence. Good practical results have been obtained by performing only a single iteration of the re-estimation formula after each new selection [20].

7.3 Regularised Orthogonal Least Squares

Orthogonal least squares (section7.1) works because after factoring $\mathbf{H}_m$ the orthogonalised variance matrix (equation 4.4) is diagonal.

$$\begin{aligned}
\mathbf{A}_m^{-1} &= \left(\mathbf{H}_m^\top \mathbf{H}_m\right)^{-1} \\
&= \mathbf{U}_m^{-1} \left(\tilde{\mathbf{H}}_m^\top \tilde{\mathbf{H}}_m\right)^{-1} \left(\mathbf{U}_m^\top\right)^{-1} \\
&= \mathbf{U}_m^{-1} \begin{bmatrix} \frac{1}{\tilde{\mathbf{h}}_1^\top \tilde{\mathbf{h}}_1} & 0 & \dots & 0 \\ 0 & \frac{1}{\tilde{\mathbf{h}}_2^\top \tilde{\mathbf{h}}_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{1}{\tilde{\mathbf{h}}_m^\top \tilde{\mathbf{h}}_m} \end{bmatrix} \left(\mathbf{U}_m^\top\right)^{-1} \\
&= \mathbf{U}_m^{-1}\, \tilde{\mathbf{A}}^{-1}\, \left(\mathbf{U}_m^\top\right)^{-1}\,.
\end{aligned}$$

$\tilde{\mathbf{A}}^{-1}$ is simple to compute while the inverse of the upper triangular matrix is not required to calculate the change in sum-squared-error. However, when standard ridge regression (section 6.1) is used

$$\begin{aligned}\mathbf{A}_m^{-1} &= \left(\mathbf{H}_m^\top \mathbf{H}_m + \lambda\,\mathbf{I}_m\right)^{-1} \\ &= \left(\mathbf{U}_m^\top \tilde{\mathbf{H}}_m^\top \tilde{\mathbf{H}}_m\,\mathbf{U}_m + \lambda\,\mathbf{I}_m\right)^{-1},\end{aligned}$$

and this expression can not be simplified any further. A solution to this problem is to slightly alter the nature of the regularisation so that

$$\begin{aligned}\mathbf{A}_m^{-1} &= \left(\mathbf{H}_m^\top \mathbf{H}_m + \lambda\,\mathbf{U}_m^\top \mathbf{U}_m\right)^{-1} \\ &= \mathbf{U}_m^{-1}\left(\tilde{\mathbf{H}}_m^\top \tilde{\mathbf{H}}_m + \lambda\,\mathbf{I}_m\right)^{-1}\left(\mathbf{U}_m^\top\right)^{-1} \\ &= \mathbf{U}_m^{-1}\begin{bmatrix} \frac{1}{\lambda+\tilde{\mathbf{h}}_1^\top\tilde{\mathbf{h}}_1} & 0 & \dots & 0 \\ 0 & \frac{1}{\lambda+\tilde{\mathbf{h}}_2^\top\tilde{\mathbf{h}}_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{1}{\lambda+\tilde{\mathbf{h}}_m^\top\tilde{\mathbf{h}}_m} \end{bmatrix}\left(\mathbf{U}_m^\top\right)^{-1} \\ &= \mathbf{U}_m^{-1}\,\tilde{\mathbf{A}}^{-1}\left(\mathbf{U}_m^\top\right)^{-1}.\end{aligned}$$

This means that instead of the normal cost function (equation 6.1) for standard ridge regression (section 6.1) the cost is actually

$$C_m = (\hat{\mathbf{y}} - \mathbf{H}_m\,\mathbf{w})^\top(\hat{\mathbf{y}} - \mathbf{H}_m\,\mathbf{w}) + \lambda\,\mathbf{w}_m^\top \mathbf{U}_m^\top \mathbf{U}_m\,\mathbf{w}_m. \tag{7.8}$$

Then the change in sum-squared-error is

$$\hat{S}_m - \hat{S}_{m+1} = \frac{(2\,\lambda + \tilde{\mathbf{f}}_J^\top\tilde{\mathbf{f}}_J)\,(\hat{\mathbf{y}}^\top\tilde{\mathbf{f}}_J)^2}{(\lambda + \tilde{\mathbf{f}}_J^\top\tilde{\mathbf{f}}_J)^2}. \tag{7.9}$$

Alternatively, searching for the best basis function to add could be made on the basis of the change in cost, which is simply

$$\hat{C}_m - \hat{C}_{m+1} = \frac{(\hat{\mathbf{y}}^\top\tilde{\mathbf{f}}_J)^2}{\lambda + \tilde{\mathbf{f}}_J^\top\tilde{\mathbf{f}}_J}. \tag{7.10}$$

For details see appendix A.15.

7.4 Example

Figure 12 shows a set of $p = 50$ training examples sampled with Gaussian noise of standard deviation $\sigma = 0.1$ from the logistic function

$$y(x) \;=\; \frac{1 - e^{-x}}{1 + e^{-x}}\,.$$

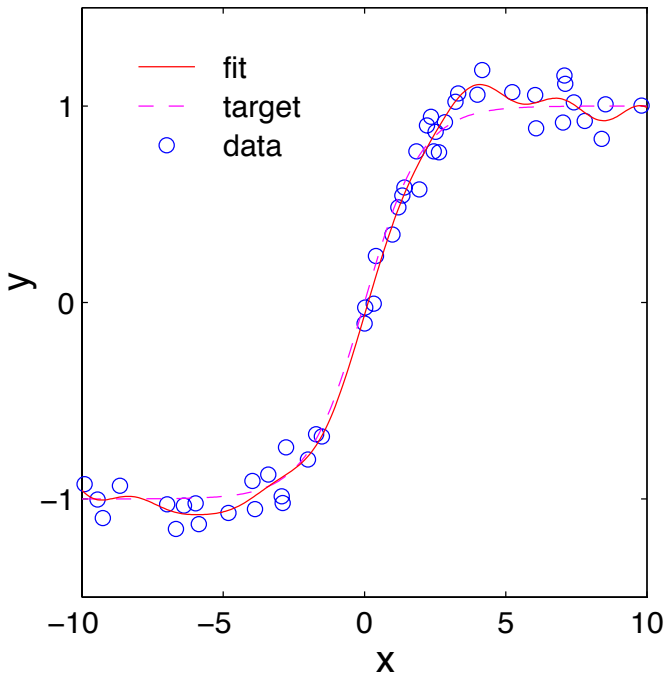

Figure 12: Training set, target function and forward selection fit.

This data is to be modeled by forward selection from a set of $M = 50$ Gaussian radial functions (section 3.1) coincident with the training set inputs and of width $r = 2$. Figure 13 shows the sum-squared-error (equation 4.1), generalised cross-validation (equation 5.2) and the mean-squared-error between the fit and target (based on an independent test set) as new basis functions are added one at a time.

The training set error monotonically decreases as more basis functions are added but the GCV error prediction eventually starts to increase (at the point marked with a star). This is the signal to stop adding basis functions (which in this example occurs after the RBF network has acquired 13 basis functions) and happily coincides with the minimum test set error. The fit with these 13 basis functions is shown in figure 12. Further additions only serve to make GCV and MSE worse. Eventually, after about 19 additions, the variance matrix (equation 4.4) becomes ill-conditioned when numerical calculations are unreliable.

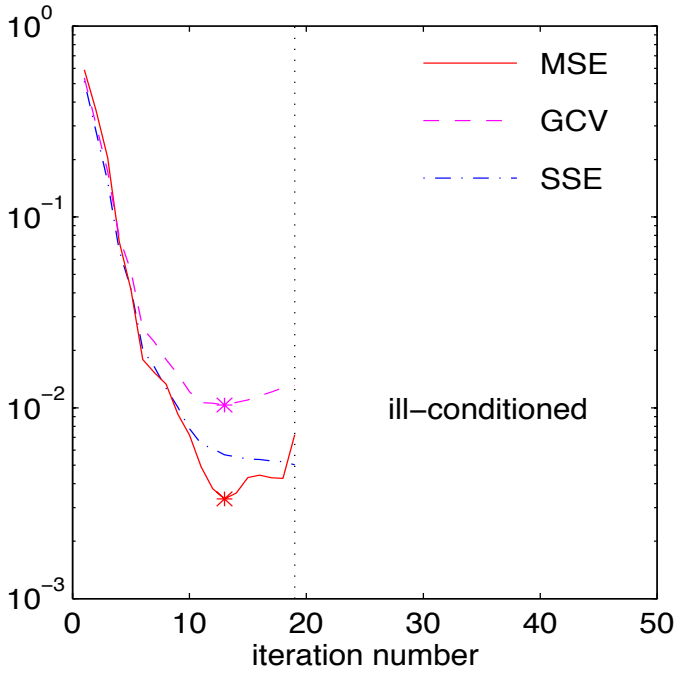

Figure 13: The progress of SSE, GCV and MSE as new basis functions are added in plain vanilla forward selection. The minimum GCV, and corresponding MSE, are marked with stars. After about 19 selections the variance matrix becomes badly conditioned.

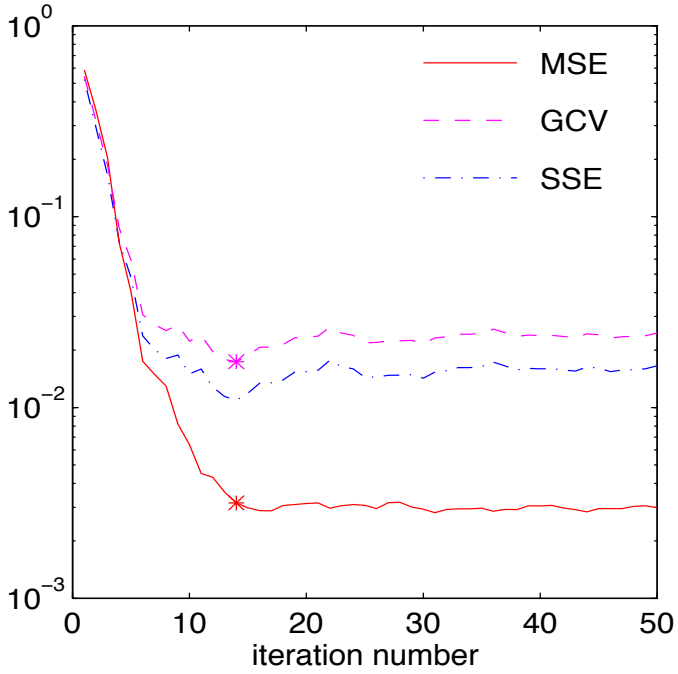

Figure 14: The progress of SSE, GCV and MSE as new basis functions are added in regularised forward selection.

The results of using regularised forward selection (section 7.2) are shown in figure 14. The sum-squared-error (equation 4.1) no longer decreases monotonically because $\hat{\lambda}$ is re-estimated after each selection. Also, regularisation prevents the variance matrix (equation 4.4) becoming ill-conditioned, even after all the candidate basis functions are in the subset.

In this example the two methods chose similar (but not identical) subsets and ended with very close test set errors. However, on average, over a large number of similar training sets, the regularised version performs slightly better than the plain vanilla version [20].

A Appendices

A.1 Notational Conventions

Scalars are represented by italicised letters such as y or λ. Vectors are represented by bold lower case letters such as $\mathbf{x}$ and $\boldsymbol{\lambda}$. The first component of $\mathbf{x}$ (a vector) is x_1 (a scalar). Vectors are single-column matrices, so, for example, if $\mathbf{x}$ has n components then

$$\mathbf{x} \;=\; \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} .$$

Matrices, such as $\mathbf{H}$, are represented by bold capital letters. If the rows of $\mathbf{H}$ are indexed by i and the columns by j then the entry in the i-th row and j-th column of $\mathbf{H}$ is H_{ij}. If $\mathbf{H}$ has p rows and m columns ($\mathbf{H} \in \mathcal{R}^{p \times m}$) then

$$\mathbf{H} \;=\; \begin{bmatrix} H_{11} & H_{12} & \dots & H_{1m} \\ H_{21} & H_{22} & \dots & H_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ H_{p1} & H_{p2} & \dots & H_{pm} \end{bmatrix} .$$

The transpose of a matrix – the rows and columns swapped – is represented by $\mathbf{H}^\top$. So, for example, if $\mathbf{G} \;=\; \mathbf{H}^\top$ then

$$G_{ji} = H_{ij} \, .$$

The transpose of a vector is a single-row matrix: $\mathbf{x}^\top = [x_1 \; x_2 \; \dots \; x_n]$. It follows that $\mathbf{x} = [x_1 \; x_2 \; \dots \; x_n]^\top$ which is a useful way of writing vectors on a single line. Also, the common operation of vector dot-product can be written as the transpose of one vector multiplied by the other vector. So instead of writing $\mathbf{x} \cdot \mathbf{y}$ we can just write $\mathbf{x}^\top \mathbf{y}$.

The identity matrix, written $\mathbf{I}$, is a square matrix (one with equal numbers of rows and columns) with diagonal entries of 1 and zero elsewhere. The dimension is written as a subscript, as in $\mathbf{I}_m$, which is the identity matrix of size m (m rows and m columns).

The inverse of a square matrix $\mathbf{A}$ is written $\mathbf{A}^{-1}$. If $\mathbf{A}$ has m rows and m columns then

$$\mathbf{A}^{-1}\,\mathbf{A} \;=\; \mathbf{A}\,\mathbf{A}^{-1} \;=\; \mathbf{I}_m \, ,$$

which defines the inverse operation.

Estimated or uncertain values are distinguished by the use of the hat symbol. For example, $\hat{\lambda}$ is an estimated value for λ, and $\hat{\mathbf{w}}$ is an estimate of $\mathbf{w}$.

A.2 Useful Properties of Matrices

Some useful definitions and properties of matrices are given below and illustrated with some of the matrices (and vectors) commonly appearing in the main text.

The elements of a *diagonal* matrix are zero off the diagonal. An example is the matrix of regularisation parameters appearing in equation 6.8 for the optimal weight in local ridge regression

$$\mathbf{\Lambda} \;=\; \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_m \end{bmatrix} .$$

Non-square matrices can also be diagonal. If $\mathbf{M}$ is any matrix then it is diagonal if $M_{ij} = 0,\ i \neq j$.

A *symmetric* matrix is identical to its own transpose. Example are the variance matrix (equation 4.4) $\mathbf{A}^{-1} \in \mathcal{R}^{m \times m}$ and the projection matrix (equation 4.6) $\mathbf{P} \in \mathcal{R}^{p \times p}$. Any square diagonal matrix, such as the identity matrix, is necessarily symmetric.

The inverse of an *orthogonal* matrix is its own transpose. If $\mathbf{V}$ is orthogonal then

$$\mathbf{V}^\top \mathbf{V} \;=\; \mathbf{V}\,\mathbf{V}^\top \;=\; \mathbf{I}_m \,.$$

Any matrix can be decomposed into the product of two orthogonal matrices and a diagonal one. This is called *singular value decomposition* (SVD) [12]. For example, the design matrix (equation 4.3) $\mathbf{H} \in \mathcal{R}^{p \times m}$ decomposes into

$$\mathbf{H} \;=\; \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^\top ,$$

where $\mathbf{U} \in \mathcal{R}^{p \times p}$ and $\mathbf{V} \in \mathcal{R}^{m \times m}$ are orthogonal and $\mathbf{\Sigma} \in \mathcal{R}^{p \times m}$ is diagonal.

The *trace* of a square matrix is the sum of its diagonal elements. The trace of the product of a sequence of matrices is unaffected by rotation of the order. For example,

$$\text{trace}\left(\mathbf{H}\,\mathbf{A}^{-1}\,\mathbf{H}^\top\right) \;=\; \text{trace}\left(\mathbf{A}^{-1}\,\mathbf{H}^\top\mathbf{H}\right) .$$

The transpose of a product is equal to the product of the individual transposes in reverse order. For example,

$$\left(\mathbf{A}^{-1}\,\mathbf{H}^\top\hat{\mathbf{y}}\right)^\top \;=\; \hat{\mathbf{y}}^\top\mathbf{H}\,\mathbf{A}^{-1} .$$

$\mathbf{A}^{-1}$ is symmetric so $(\mathbf{A}^{-1})^\top = \mathbf{A}^{-1}$.

The inverse of a product of square matrices is equal to the product of the individual inverses in reverse order. For example,

$$\left(\mathbf{V}\,\mathbf{\Sigma}^\top\mathbf{\Sigma}\,\mathbf{V}^\top\right)^{-1} \;=\; \mathbf{V}\,\left(\mathbf{\Sigma}^\top\mathbf{\Sigma}\right)^{-1}\mathbf{V}^\top .$$

$\mathbf{V}$ is orthogonal so $\mathbf{V}^{-1} = \mathbf{V}^\top$ and $(\mathbf{V}^\top)^{-1} = \mathbf{V}$.

A.3 Radial Basis Functions

The most general formula for any radial basis function (RBF) is

$$h(\mathbf{x}) \;=\; \phi\left((\mathbf{x}-\mathbf{c})^\top \mathbf{R}^{-1}(\mathbf{x}-\mathbf{c})\right) \;,$$

where ϕ is the function used (Gaussian, multiquadric, etc...), $\mathbf{c}$ is the centre and $\mathbf{R}$ is the metric. The term $(\mathbf{x}-\mathbf{c})^\top \mathbf{R}^{-1}(\mathbf{x}-\mathbf{c})$ is the distance between the input $\mathbf{x}$ and the centre $\mathbf{c}$ in the metric defined by $\mathbf{R}$. There are several common types of functions used, for example, the Gaussian, $\phi(z) = e^{-z}$, the multiquadric, $\phi(z) = (1+z)^{\frac{1}{2}}$, the inverse multiquadric, $\phi(z) = (1+z)^{-\frac{1}{2}}$ and the Cauchy $\phi(z) = (1+z)^{-1}$.

Often the metric is Euclidean. In this case $\mathbf{R} = r^2\,\mathbf{I}$ for some scalar radius r and the above equation simplifies to

$$h(\mathbf{x}) \;=\; \phi\left(\frac{(\mathbf{x}-\mathbf{c})^\top(\mathbf{x}-\mathbf{c})}{r^2}\right) \;.$$

A further simplification is a 1-dimensional input space in which case we have

$$h(\mathbf{x}) \;=\; \phi\left(\frac{(x-c)^2}{r^2}\right) \;.$$

This function is illustrated for $c = 0$ and $r = 1$ in figure 15 for the above RBF types.

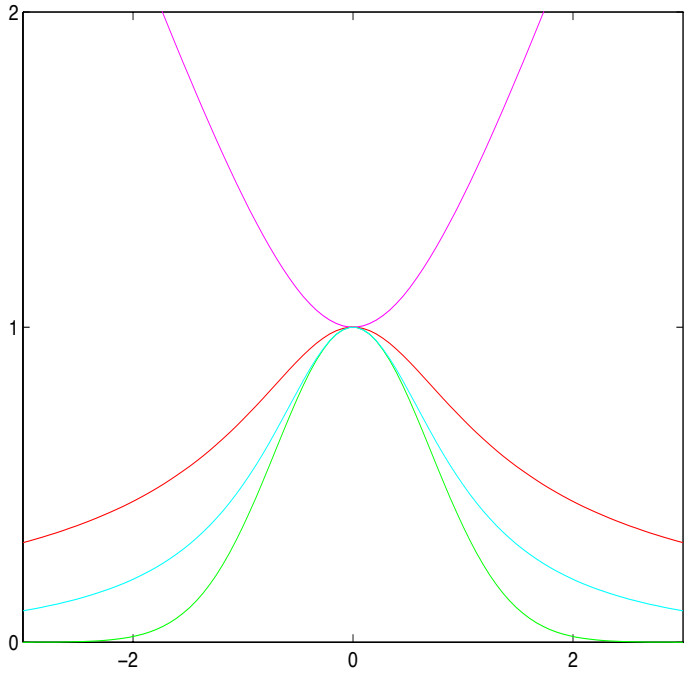

Figure 15: Gaussian (green), multiquadric (magenta), inverse-multiquadric (red) and Cauchy (cyan) RBFs all of unit radius and all centred at the origin.

A.4 The Optimal Weight Vector

As is well known from elementary calculus, to find an extremum of a function you

1. differentiate the function with respect to the free variable(s),
2. equate the result(s) with zero, and
3. solve the resulting equation(s).

In the case of least squares (section 4) applied to supervised learning (section 2) with a linear model (section 3) the function to be minimised is the sum-squared-error

$$S \;=\; \sum_{i=1}^{p} \left(\hat{y}_i - f(\mathbf{x}_i)\right)^2 \;,$$

where

$$f(\mathbf{x}) \;=\; \sum_{j=1}^{m} w_j \, h_j(\mathbf{x}) \,,$$

and the free variables are the weights $\{w_j\}_{j=1}^{m}$.

To avoid repeating two very similar analyses we shall find the minimum not of S but of the cost function

$$C \;=\; \sum_{i=1}^{p} \left(\hat{y}_i - f(\mathbf{x}_i)\right)^2 + \sum_{j=1}^{m} \lambda_j \, w_j^2 \,,$$

used in ridge regression (section 6). This includes an additional weight penalty term controlled by the values of the non-negative regularisation parameters, $\{\lambda\}_{j=1}^{m}$. To get back to ordinary least squares (without any weight penalty) is simply a matter of setting all the regularisation parameters to zero.

So, let us carry out this optimisation for the j-th weight. First, differentiating the cost function

$$\frac{\partial C}{\partial w_j} \;=\; 2 \sum_{i=1}^{p} \left(f(\mathbf{x}_i) - \hat{y}_i\right) \frac{\partial f}{\partial w_j}(\mathbf{x}_i) + 2 \, \lambda_j \, w_j \,.$$

We now need the derivative of the model which, because the model is linear, is particularly simple.

$$\frac{\partial f}{\partial w_j}(\mathbf{x}_i) \;=\; h_j(\mathbf{x}_i) \,.$$

Substituting this into the derivative of the cost function and equating the result to zero leads to the equation

$$\sum_{i=1}^{p} f(\mathbf{x}_i) \, h_j(\mathbf{x}_i) + \lambda_j \, \hat{w}_j \;=\; \sum_{i=1}^{p} \hat{y}_i \, h_j(\mathbf{x}_i) \,.$$

There are m such equations, for $1 \leq j \leq m$, each representing one constraint on the solution. Since there are exactly as many constraints as there are unknowns the system of equations has, except under certain pathological conditions, a unique solution.

To find that unique solution we employ the language of matrices and vectors: linear algebra. These are invaluable for the representation and analysis of systems of linear equations like the one above which can be rewritten in vector notation (see appendix A.1) as follows.

$$\mathbf{h}_j^\top \mathbf{f} + \lambda_j \, \hat{w}_j \;=\; \mathbf{h}_j^\top \hat{\mathbf{y}} \,,$$

where

$$\mathbf{h}_j \;=\; \begin{bmatrix} h_j(\mathbf{x}_1) \\ h_j(\mathbf{x}_2) \\ \vdots \\ h_j(\mathbf{x}_p) \end{bmatrix}, \quad \mathbf{f} \;=\; \begin{bmatrix} f(\mathbf{x}_1) \\ f(\mathbf{x}_2) \\ \vdots \\ f(\mathbf{x}_p) \end{bmatrix}, \quad \hat{\mathbf{y}} \;=\; \begin{bmatrix} \hat{y}_1 \\ \hat{y}_2 \\ \vdots \\ \hat{y}_p \end{bmatrix}.$$

Since there is one of these equations (each relating one scalar quantity to another) for each value of j from 1 up to m we can stack them, one on top of another, to create a relation between two vector quantities.

$$\begin{bmatrix} \mathbf{h}_1^\top \mathbf{f} \\ \mathbf{h}_2^\top \mathbf{f} \\ \vdots \\ \mathbf{h}_m^\top \mathbf{f} \end{bmatrix} + \begin{bmatrix} \lambda_1 \, \hat{w}_1 \\ \lambda_2 \, \hat{w}_2 \\ \vdots \\ \lambda_m \, \hat{w}_m \end{bmatrix} \;=\; \begin{bmatrix} \mathbf{h}_1^\top \hat{\mathbf{y}} \\ \mathbf{h}_2^\top \hat{\mathbf{y}} \\ \vdots \\ \mathbf{h}_m^\top \hat{\mathbf{y}} \end{bmatrix}.$$

However, using the laws of matrix multiplication, this is just equivalent to

$$\mathbf{H}^\top \mathbf{f} + \mathbf{\Lambda} \, \hat{\mathbf{w}} \;=\; \mathbf{H}^\top \hat{\mathbf{y}} \,, \tag{A.1}$$

where

$$\mathbf{\Lambda} \;=\; \begin{bmatrix} \lambda_1 & 0 & \dots & 0 \\ 0 & \lambda_2 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \lambda_m \end{bmatrix}.$$

and where $\mathbf{H}$, which is called the *design matrix*, has the vectors $\{\mathbf{h}_j\}_{j=1}^m$ as its columns,

$$\mathbf{H} \;=\; [\mathbf{h}_1 \; \mathbf{h}_2 \; \dots \; \mathbf{h}_m] \,,$$

and has p rows, one for each pattern in the training set. Written out in full it is

$$\mathbf{H} \;=\; \begin{bmatrix} h_1(\mathbf{x}_1) & h_2(\mathbf{x}_1) & \dots & h_m(\mathbf{x}_1) \\ h_1(\mathbf{x}_2) & h_2(\mathbf{x}_2) & \dots & h_m(\mathbf{x}_2) \\ \vdots & \vdots & \ddots & \vdots \\ h_1(\mathbf{x}_p) & h_2(\mathbf{x}_p) & \dots & h_m(\mathbf{x}_p) \end{bmatrix},$$

which is where equation 4.3 came from.

The vector $\mathbf{f}$ can be decomposed into the product of two terms, the design matrix and the weight vector, since each of its components is a dot-product between two m-dimensional vectors. For example, the i-th component of $\mathbf{f}$ when the weights are at their optimal values is

$$f_i \;=\; f(\mathbf{x}_i) \;=\; \sum_{j=1}^{m} \hat{w}_j \, h_j(\mathbf{x}_i) \;=\; \bar{\mathbf{h}}_i^\top \hat{\mathbf{w}} \,,$$

where

$$\bar{\mathbf{h}}_i \;=\; \begin{bmatrix} h_1(\mathbf{x}_i) \\ h_2(\mathbf{x}_i) \\ \vdots \\ h_m(\mathbf{x}_i) \end{bmatrix} .$$

Note that while $\mathbf{h}_j$ is one of the columns of $\mathbf{H}$, $\bar{\mathbf{h}}_i^\top$ is one of its rows. $\mathbf{f}$ is the result of stacking the $\{f_i\}_{i=1}^p$ one on top of the other, or

$$\mathbf{f} \;=\; \begin{bmatrix} f_1 \\ f_2 \\ \vdots \\ f_p \end{bmatrix} \;=\; \begin{bmatrix} \bar{\mathbf{h}}_1^\top \hat{\mathbf{w}} \\ \bar{\mathbf{h}}_2^\top \hat{\mathbf{w}} \\ \vdots \\ \bar{\mathbf{h}}_p^\top \hat{\mathbf{w}} \end{bmatrix} \;=\; \mathbf{H}\,\hat{\mathbf{w}} \,.$$

Finally, substituting this expression for $\mathbf{f}$ into the equation A.1 gives

$$\begin{aligned} \mathbf{H}^\top \hat{\mathbf{y}} &= \mathbf{H}^\top \mathbf{f} + \boldsymbol{\Lambda}\,\hat{\mathbf{w}} \\ &= \mathbf{H}^\top \mathbf{H}\,\hat{\mathbf{w}} + \boldsymbol{\Lambda}\,\hat{\mathbf{w}} \\ &= \left(\mathbf{H}^\top \mathbf{H} + \boldsymbol{\Lambda}\right)\,\hat{\mathbf{w}} \,, \end{aligned}$$

the solution to which is

$$\hat{\mathbf{w}} \;=\; \left(\mathbf{H}^\top \mathbf{H} + \boldsymbol{\Lambda}\right)^{-1} \mathbf{H}^\top \hat{\mathbf{y}} \,,$$

which is where equation 4.5 comes from.

The latter equation is the most general form of the normal equation which we deal with here. There are two special cases. In standard ridge regression (section 6) $\lambda_j = \lambda,\ 1 \le j \le m$, so

$$\hat{\mathbf{w}} \;=\; \left(\mathbf{H}^\top \mathbf{H} + \lambda\,\mathbf{I}_m\right)^{-1} \mathbf{H}^\top \hat{\mathbf{y}} \,,$$

which is where equation 6.3 comes from.

Ordinary least squares, where there is no weight penalty, is obtained by setting all regularisation parameters to zero so

$$\hat{\mathbf{w}} \;=\; \left(\mathbf{H}^\top \mathbf{H}\right)^{-1} \mathbf{H}^\top \hat{\mathbf{y}} \,.$$

A.5 The Variance Matrix

What is the variance of the weight vector $\mathbf{w}$? In other words, since the weights have been calculated upon the basis of a measurement, $\hat{\mathbf{y}}$, of a stochastic variable $\mathbf{y}$, what is the corresponding uncertainty in $\hat{\mathbf{w}}$?

The answer to this question depends on the nature and size of the uncertainty in $\hat{\mathbf{y}}$ and the relationship between $\mathbf{w}$ and $\mathbf{y}$ considered as random variables. In our case we assume that the noise affecting $\mathbf{y}$ is normal and independently, identically distributed:

$$\langle (\mathbf{y} - \bar{\mathbf{y}})\,(\mathbf{y} - \bar{\mathbf{y}}) \rangle \;=\; \sigma^2\,\mathbf{I}_p\,,$$

where σ is the standard deviation of the noise, $\bar{\mathbf{y}}$ the mean value of $\mathbf{y}$ and the expectation (averaging) is taken over training sets. Since we consider only linear networks (section 3), the relationship between $\mathbf{w}$ and $\mathbf{y}$ is given by

$$\mathbf{w} \;=\; \mathbf{A}^{-1}\,\mathbf{H}^\top \mathbf{y}\,,$$

where $\mathbf{A} = \mathbf{H}^\top\mathbf{H} + \mathbf{\Lambda}$ (see appendix A.4). Then the expected (mean) value of $\mathbf{w}$ is

$$\bar{\mathbf{w}} \;=\; \langle \mathbf{A}^{-1}\,\mathbf{H}^\top \mathbf{y} \rangle \;=\; \mathbf{A}^{-1}\,\mathbf{H}^\top \langle \mathbf{y} \rangle \;=\; \mathbf{A}^{-1}\,\mathbf{H}^\top \bar{\mathbf{y}}\,,$$

so the variance, $\hat{\mathbf{W}}$ is

$$\begin{aligned}
\hat{\mathbf{W}} &= \langle (\mathbf{w} - \hat{\mathbf{w}})\,(\mathbf{w} - \hat{\mathbf{w}})^\top \rangle \\
&= \mathbf{A}^{-1}\,\mathbf{H}^\top \langle (\mathbf{y} - \hat{\mathbf{y}})\,(\mathbf{y} - \hat{\mathbf{y}})^\top \rangle\,\mathbf{H}\,\mathbf{A}^{-1} \\
&= \sigma^2\,\mathbf{A}^{-1}\,\mathbf{H}^\top\mathbf{H}\,\mathbf{A}^{-1}\,.
\end{aligned}$$

In ordinary least squares (section 4) $\mathbf{H}^\top\mathbf{H} = \mathbf{A}$ so $\hat{\mathbf{W}} = \sigma^2\,\mathbf{A}^{-1}$. We shall refer to the matrix $\mathbf{A}^{-1}$ as the variance matrix, because of its close link to the variance of $\mathbf{w}$ in ordinary least squares. For standard ridge regression (section 6.1) when $\mathbf{H}^\top\mathbf{H} = \mathbf{A} - \lambda\,\mathbf{I}_m$

$$\begin{aligned}
\hat{\mathbf{W}} &= \sigma^2\,\mathbf{A}^{-1}(\mathbf{A} - \lambda\,\mathbf{I}_m)\,\mathbf{A}^{-1} \\
&= \sigma^2\,\left(\mathbf{A}^{-1} - \lambda\,\mathbf{A}^{-2}\right)\,.
\end{aligned}$$

An important point is that if the training set has been used to estimate the regularisation parameter(s), as in ridge regression (section 6), or to chose the basis functions, as in forward selection (section 7), then $\mathbf{A}$ (as well as $\mathbf{y}$) is a stochastic variable and there is no longer a simple linear relationship between uncertainty in $\hat{\mathbf{w}}$ and uncertainty in $\hat{\mathbf{y}}$. In this case it is probably necessary to resort to bootstrap techniques [14] to estimate $\hat{\mathbf{W}}$.

A.6 The Projection Matrix

The prediction of the output at any of the training set inputs by the linear model (section 3) using equation (equation 4.1) for the weight vector is

$$\begin{aligned} f(\mathbf{x}_i) &= \sum_{j=1}^{m} \hat{w}_j \, h_j(\mathbf{x}_i) \\ &= \mathbf{h}_i^\top \hat{\mathbf{w}} \,, \end{aligned}$$

where $\mathbf{h}_i^\top$ is the i-th row of the design matrix (equation 4.3). If we stack these predictions into a vector, $\mathbf{f}$, we get

$$\begin{aligned} \mathbf{f} &= \begin{bmatrix} \mathbf{h}_1^\top \hat{\mathbf{w}} \\ \mathbf{h}_2^\top \hat{\mathbf{w}} \\ \vdots \\ \mathbf{h}_p^\top \hat{\mathbf{w}} \end{bmatrix} \\ &= \mathbf{H} \, \hat{\mathbf{w}} \\ &= \mathbf{H} \, \mathbf{A}^{-1} \, \mathbf{H}^\top \hat{\mathbf{y}} \,. \end{aligned}$$

Therefore, the error vector between the predictions of the network from the training set inputs and the actual outputs observed in the training set is

$$\begin{aligned} \hat{\mathbf{y}} - \mathbf{f} &= \hat{\mathbf{y}} - \mathbf{H} \, \mathbf{A}^{-1} \, \mathbf{H}^\top \hat{\mathbf{y}} \\ &= \left(\mathbf{I}_p - \mathbf{H} \, \mathbf{A}^{-1} \, \mathbf{H}^\top \right) \hat{\mathbf{y}} \\ &= \mathbf{P} \, \hat{\mathbf{y}} \,, \end{aligned}$$

where

$$\mathbf{P} = \mathbf{I}_p - \mathbf{H} \, \mathbf{A}^{-1} \, \mathbf{H}^\top \,.$$

As explained in section 4.2 $\mathbf{P} \, \hat{\mathbf{y}}$ is the projection of $\hat{\mathbf{y}}$ perpendicular to the space spanned by the model and represents the error between the observed outputs and the least squares prediction (at least in the absence of any weight penalty). The sum-squared-error (equation 4.1) at the weight (equation 4.1) which minimises the cost function (equation 4.2), can be conveniently written in terms of $\mathbf{P}$ and $\hat{\mathbf{y}}$ by

$$\begin{aligned} \hat{S} &= (\hat{\mathbf{y}} - \mathbf{f})^\top (\hat{\mathbf{y}} - \mathbf{f}) \\ &= \hat{\mathbf{y}}^\top \mathbf{P}^\top \mathbf{P} \, \hat{\mathbf{y}} \\ &= \hat{\mathbf{y}}^\top \mathbf{P}^2 \, \hat{\mathbf{y}} \,, \end{aligned}$$

the last step following because $\mathbf{P}$ is symmetric ($\mathbf{P} = \mathbf{P}^\top$). Also, the cost function (equation 4.2) itself, is

$$\begin{aligned}
\hat{C} &= (\mathbf{H}\,\hat{\mathbf{w}} - \hat{\mathbf{y}})^\top (\mathbf{H}\,\hat{\mathbf{w}} - \hat{\mathbf{y}}) + \hat{\mathbf{w}}^\top \mathbf{\Lambda}\,\hat{\mathbf{w}} \\
&= \hat{\mathbf{y}}^\top (\mathbf{H}\,\mathbf{A}^{-1}\,\mathbf{H}^\top - \mathbf{I}_p)\,(\mathbf{H}\,\mathbf{A}^{-1}\,\mathbf{H} - \mathbf{I}_p)\,\hat{\mathbf{y}} + \hat{\mathbf{y}}^\top \mathbf{H}\,\mathbf{A}^{-1}\mathbf{\Lambda}\,\mathbf{A}^{-1}\,\mathbf{H}^\top \hat{\mathbf{y}} \\
&= \hat{\mathbf{y}}^\top \mathbf{P}^2\,\hat{\mathbf{y}} + \hat{\mathbf{y}}^\top \mathbf{H}\,\mathbf{A}^{-1}\mathbf{\Lambda}\,\mathbf{A}^{-1}\,\mathbf{H}^\top \hat{\mathbf{y}}\,.
\end{aligned}$$

However, because

$$\begin{aligned}
\mathbf{H}\,\mathbf{A}^{-1}\mathbf{\Lambda}\,\mathbf{A}^{-1}\,\mathbf{H}^\top &= \mathbf{H}\,\mathbf{A}^{-1}\left(\mathbf{A} - \mathbf{H}^\top\mathbf{H}\right)\,\mathbf{A}^{-1}\,\mathbf{H}^\top \\
&= \mathbf{H}\,\mathbf{A}^{-1}\,\mathbf{H}^\top - \left(\mathbf{H}\,\mathbf{A}^{-1}\,\mathbf{H}^\top\right)^2 \\
&= \mathbf{P} - \mathbf{P}^2\,,
\end{aligned}$$

the expression for the minimum cost simplifies to

$$\begin{aligned}
\hat{C} &= \hat{\mathbf{y}}^\top \mathbf{P}^2\,\hat{\mathbf{y}} + \hat{\mathbf{y}}^\top \left(\mathbf{P} - \mathbf{P}^2\right)\,\hat{\mathbf{y}} \\
&= \hat{\mathbf{y}}^\top \mathbf{P}\,\hat{\mathbf{y}}\,.
\end{aligned}$$

A.7 Incremental Operations

If a matrix $\mathbf{A}$ is square of size m and none of its columns (rows) are linear combinations of its other columns (rows) then there exists a unique matrix, $\mathbf{A}^{-1}$, called the inverse of $\mathbf{A}$, which satisfies

$$\begin{aligned}
\mathbf{A}^{-1}\,\mathbf{A} &= \mathbf{I}_m\,, \\
\mathbf{A}\,\mathbf{A}^{-1} &= \mathbf{I}_m\,.
\end{aligned}$$

The following are two useful lemmas for matrix inversion [12]. They find frequent use whenever the design matrix (equation 4.3) is partitioned, as for example, in any of the incremental operations (section 4.3).

The small rank adjustment is

$$\mathbf{A}_1 = \mathbf{A}_0 + \mathbf{X}\,\mathbf{R}\,\mathbf{Y}\,,$$

where the inverse of $\mathbf{A}_0 \in \mathcal{R}^{m\times m}$ is known, $\mathbf{X}$, $\mathbf{Y}^\top \in \mathcal{R}^{m\times r}$ are known ($m > r$), $\mathbf{R} \in \mathcal{R}^{r\times r}$ is known and the inverse of $\mathbf{A}_1$ is sought. Instead of recomputing the inverse from scratch, the formula

$$\mathbf{A}_1^{-1} = \mathbf{A}_0^{-1} - \mathbf{A}_0^{-1}\mathbf{X}\,(\mathbf{Y}\,\mathbf{A}_0^{-1}\mathbf{X} + \mathbf{R}^{-1})^{-1}\mathbf{Y}\,\mathbf{A}_0^{-1}\,, \tag{A.2}$$

may reach the same answer more efficiently since it involves numerically inverting an r-sized matrix instead of a larger m-sized one (the cost of one matrix inverse is roughly proportional to the cube of its size).

The inverse of a partitioned matrix,

$$\mathbf{A} = \begin{bmatrix} \mathbf{A}_{11} & \mathbf{A}_{12} \\ \mathbf{A}_{21} & \mathbf{A}_{22} \end{bmatrix},$$

can be expressed as

$$\mathbf{A}^{-1} = \begin{bmatrix} (\mathbf{A}_{11} - \mathbf{A}_{12}\,\mathbf{A}_{22}^{-1}\mathbf{A}_{21})^{-1} & \mathbf{A}_{11}^{-1}\mathbf{A}_{12}\,(\mathbf{A}_{21}\,\mathbf{A}_{11}^{-1}\mathbf{A}_{12} - \mathbf{A}_{22})^{-1} \\ (\mathbf{A}_{21}\,\mathbf{A}_{11}^{-1}\mathbf{A}_{12} - \mathbf{A}_{22})^{-1}\mathbf{A}_{21}\,\mathbf{A}_{11}^{-1} & (\mathbf{A}_{22} - \mathbf{A}_{21}\,\mathbf{A}_{11}^{-1}\mathbf{A}_{12})^{-1} \end{bmatrix},$$

assuming that all the relevant inverses exist. Alternatively, using the small rank adjustment formula (equation A.2) and defining $\Delta = \mathbf{A}_{22} - \mathbf{A}_{21}\,\mathbf{A}_{11}^{-1}\mathbf{A}_{12}$ the same inverse can be written

$$\mathbf{A}^{-1} = \begin{bmatrix} \mathbf{A}_{11}^{-1} + \mathbf{A}_{11}^{-1}\mathbf{A}_{12}\,\Delta^{-1}\mathbf{A}_{21}\,\mathbf{A}_{11}^{-1} & -\mathbf{A}_{11}^{-1}\mathbf{A}_{12}\,\Delta^{-1} \\ -\Delta^{-1}\mathbf{A}_{21}\,\mathbf{A}_{11}^{-1} & \Delta^{-1} \end{bmatrix}. \tag{A.3}$$

The next four subsections applies these formulae to working out the effects of incremental operations (section 4.3) on linear models (section 3) applied to supervised learning (section 2) by considering what happens to the variance matrix (appendix A.5) and the projection matrix (appendix A.6). We employ the general ridge regression formulae for both the variance matrix (equation 4.4) and the projection matrix (equation 4.6). Ordinary least squares (section 4) is the special case where all the regularisation parameters are zero.

Retraining a linear network with an incremental operation can lead to considerable compute time savings over the alternative of retraining from scratch, which involves constructing the new design matrix, multiplying it with itself, adding the regulariser (if there is one), taking the inverse to obtain the variance matrix (appendix A.5) and recomputing the projection matrix (appendix A.6). Table 3 gives the approximate number of multiplications (ignoring first order terms) required to compute $\mathbf{P}$.

operation	completely retrain	use operation
add a new basis function	$m^3 + p\,m^2 + p^2\,m$	p^2
remove an old basis function	$m^3 + p\,m^2 + p^2\,m$	p^2
add a new pattern	$m^3 + p\,m^2 + p^2\,m$	$2\,m^2 + p\,m + p^2$
remove an old pattern	$m^3 + p\,m^2 + p^2\,m$	$2\,m^2 + p\,m + p^2$

Table 3: The cost (in multiplications) of calculating the projection matrix by retraining from scratch compared with using the appropriate incremental operation.

A.7.1 Adding a new basis function

Adding a new basis function, the $(m+1)$-th, to a linear model (section 3) which already has m basis functions and applying it to a training set (section 2) with p

patterns has the effect of adding an extra column to the design matrix (equation 4.3). If the old design matrix is $\mathbf{H}_m$, and the new basis function is $h_{m+1}(\mathbf{x})$ then the new design matrix is

$$\mathbf{H}_{m+1} \;=\; \left[\begin{array}{cc} \mathbf{H}_m & \mathbf{h}_{m+1} \end{array}\right],$$

where

$$\mathbf{h}_{m+1} \;=\; \left[\begin{array}{c} h_{m+1}(\mathbf{x}_1) \\ h_{m+1}(\mathbf{x}_2) \\ \vdots \\ h_{m+1}(\mathbf{x}_p) \end{array}\right].$$

The new variance matrix (appendix A.5) is $\mathbf{A}_{m+1}^{-1}$ where

$$\begin{aligned} \mathbf{A}_{m+1} &= \mathbf{H}_{m+1}^\top \mathbf{H}_{m+1} + \mathbf{\Lambda}_{m+1} \\ &= \left[\begin{array}{c} \mathbf{H}_m^\top \\ \mathbf{h}_{m+1}^\top \end{array}\right] \left[\begin{array}{cc} \mathbf{H}_m & \mathbf{h}_{m+1} \end{array}\right] + \left[\begin{array}{cc} \mathbf{\Lambda}_m & \mathbf{0} \\ \mathbf{0}^\top & \lambda_{m+1} \end{array}\right] \\ &= \left[\begin{array}{cc} \mathbf{A}_m & \mathbf{H}_m^\top \mathbf{h}_{m+1} \\ \mathbf{h}_{m+1}^\top \mathbf{H}_m & \lambda_{m+1} + \mathbf{h}_{m+1}^\top \mathbf{h}_{m+1} \end{array}\right], \end{aligned}$$

and where $\mathbf{A}_m = \mathbf{H}_m^\top \mathbf{H}_m + \mathbf{\Lambda}_m$. Applying the formula for the inverse of a partitioned matrix (equation A.3) yields

$$\mathbf{A}_{m+1}^{-1} \;=\; \left[\begin{array}{cc} \mathbf{A}_m^{-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{array}\right] \;+$$

$$\frac{1}{\lambda_{m+1} + \mathbf{h}_{m+1}^\top \mathbf{P}_m \, \mathbf{h}_{m+1}} \left[\begin{array}{c} \mathbf{A}_m^{-1} \mathbf{H}_m^\top \mathbf{h}_{m-1} \\ -1 \end{array}\right] \left[\begin{array}{c} \mathbf{A}_m^{-1} \mathbf{H}_m^\top \mathbf{h}_{m-1} \\ -1 \end{array}\right]^\top . \qquad \text{(A.4)}$$

where $\mathbf{P}_m \;=\; \mathbf{I}_p - \mathbf{H}_m \, \mathbf{A}_m^{-1} \mathbf{H}_m^\top$ is the projection matrix (appendix A.6) for the original network with m basis functions. The above result for $\mathbf{A}_{m+1}^{-1}$ can be used to calculate the new projection matrix after the $(m+1)$-th basis function has been added, which is

$$\begin{aligned} \mathbf{P}_{m+1} &= \mathbf{I}_p - \mathbf{H}_{m+1} \, \mathbf{A}_{m+1}^{-1} \mathbf{H}_{m+1}^\top \\ &= \mathbf{P}_m - \frac{\mathbf{P}_m \, \mathbf{h}_{m+1} \, \mathbf{h}_{m+1}^\top \mathbf{P}_m}{\lambda_{m+1} + \mathbf{h}_{m+1}^\top \mathbf{P}_m \, \mathbf{h}_{m+1}} . \qquad \text{(A.5)} \end{aligned}$$

A.7.2 Removing an old basis function

The line of reasoning is similar here to the section A.7.1 except we are interested in knowing the new projection matrix, $\mathbf{P}_{m-1}$, after the j-th basis function ($1 \leq j \leq m$) has been removed from a network with m basis functions and for which we know the old projection matrix, $\mathbf{P}_m$. The main difference is that any column can be removed from $\mathbf{H}_m$ whereas in the previous section the new column was inserted in a fixed position (right at the end of $\mathbf{H}_m$, just after the m-th column).

As noted in [19], the projection matrix (appendix A.6) is invariant to a permutation of the columns of the design matrix (equation 4.3) which is why it does not matter in which position a new column is inserted; putting it at the end is as good a choice as any other. If we want to remove a column, say the j-th one, we can shuffle it to the end position (without altering $\mathbf{P}_m$) and apply equation A.5 but with $\mathbf{P}_m$ in place of $\mathbf{P}_{m+1}$, $\mathbf{P}_{m-1}$ in place of $\mathbf{P}_m$, $\mathbf{h}_j$ in place of $\mathbf{h}_{m+1}$ and λ_j in place of λ_{m+1}. The result is

$$\mathbf{P}_m \;=\; \mathbf{P}_{m-1} - \frac{\mathbf{P}_{m-1}\,\mathbf{h}_j\,\mathbf{h}_j^\top \mathbf{P}_{m-1}}{\lambda_j + \mathbf{h}_j^\top \mathbf{P}_{m-1}\,\mathbf{h}_j}\,. \tag{A.6}$$

This equation gives the old known $\mathbf{P}_m$ in terms of the new unknown $\mathbf{P}_{m-1}$ and not the other way around, which would be more useful. If $\lambda_j \neq 0$, and only in that case (as far as I can see), it is possible to reverse the relationship to arrive at

$$\mathbf{P}_{m-1} \;=\; \mathbf{P}_m + \frac{\mathbf{P}_m\,\mathbf{h}_j\,\mathbf{h}_j^\top \mathbf{P}_m}{\lambda_j - \mathbf{h}_j^\top \mathbf{P}_m\,\mathbf{h}_j}\,. \tag{A.7}$$

(by first post- and then pre-multiplying by $\mathbf{h}_j$ to obtain expressions for $\mathbf{P}_{m-1}\,\mathbf{h}_j$ and $\mathbf{h}_j^\top \mathbf{P}_{m-1}\,\mathbf{h}_j$ in terms of $\mathbf{P}_m$). However, beware of round-off errors when using this equation in a computer program if λ_j is small.

A.7.3 Adding a new training pattern

As we have seen above, in the case of incremental adjustments to the basis functions the projection matrix (appendix A.6) keeps the same size ($p \times p$) while the variance matrix (appendix A.5) either shrinks or expands by one row and one column. If a single example is added (removed) from the training set then it is the projection matrix (appendix A.6) which expands (contracts) and the variance matrix (appendix A.5) which maintains the same size.

If a single new example, $(\mathbf{x}_{p+1},\, \hat{y}_{p+1})$, is added to the training set the design matrix (equation 4.3) acquires a new row,

$$\mathbf{h}_{p+1}^\top \;=\; [h_1(\mathbf{x}_{p+1})\;\; h_2(\mathbf{x}_{p+1})\;\; \ldots\;\; h_m(\mathbf{x}_{p+1})]\,.$$

The order in which the examples are listed does not matter, but for convenience we shall insert the new row in the last position, just after row p. The new design matrix is then

$$\mathbf{H}_{p+1} \;=\; \left[\begin{array}{c} \mathbf{H}_p \\ \mathbf{h}_{p+1}^\top \end{array}\right],$$

and the new variance matrix (appendix A.5) is $\mathbf{A}_{p+1}^{-1}$ where

$$\begin{aligned} \mathbf{A}_{p+1} &= \mathbf{H}_{p+1}^\top \mathbf{H}_{p+1} \\ &= \left[\begin{array}{cc} \mathbf{H}_p^\top & \mathbf{h}_{p+1} \end{array}\right] \left[\begin{array}{c} \mathbf{H}_p \\ \mathbf{h}_{p+1}^\top \end{array}\right] \\ &= \mathbf{A}_p + \mathbf{h}_{p+1}\,\mathbf{h}_{p+1}^\top, \end{aligned}$$

and this is true with or without ridge regression (section 6). Applying the formula for a small rank adjustment (equation A.2) (with $\mathbf{X}$, $\mathbf{Y}^\top = \mathbf{h}_{p+1}$ and $\mathbf{R} = 1$) then gives

$$\mathbf{A}_{p+1}^{-1} \;=\; \mathbf{A}_p^{-1} - \frac{\mathbf{A}_p^{-1}\mathbf{h}_{p+1}\,\mathbf{h}_{p+1}^\top \mathbf{A}_p^{-1}}{1 + \mathbf{h}_{p+1}^\top \mathbf{A}_p^{-1}\mathbf{h}_{p+1}}. \tag{A.8}$$

The new projection matrix is

$$\mathbf{P}_{p+1} \;=\; \mathbf{I}_p - \left[\begin{array}{c} \mathbf{H}_p \\ \mathbf{h}_{p+1}^\top \end{array}\right] \mathbf{A}_{p+1}^{-1} \left[\begin{array}{cc} \mathbf{H}_p^\top & \mathbf{h}_{p+1} \end{array}\right] \;=\; \left[\begin{array}{cc} \mathbf{P}_p^{-1} & \mathbf{0} \\ \mathbf{0}^\top & 0 \end{array}\right] +$$

$$\frac{1}{1 + \mathbf{h}_{p+1}^\top \mathbf{A}_p^{-1}\mathbf{h}_{p+1}} \left[\begin{array}{c} \mathbf{H}_p\,\mathbf{A}_p^{-1}\mathbf{h}_{p+1} \\ -1 \end{array}\right] \left[\begin{array}{c} \mathbf{H}_p\,\mathbf{A}_p^{-1}\mathbf{h}_{p+1} \\ -1 \end{array}\right]^\top. \tag{A.9}$$

A.7.4 Removing an old training pattern

The line of reasoning is similar here to section A.7.3 except we are interested in knowing the new variance matrix, $\mathbf{A}_{p-1}^{-1}$, after the i-th pattern ($1 \le i \le p$) has been removed from a network trained with p patterns and for which we know the old variance matrix, $\mathbf{A}_p^{-1}$. The main difference is that any row can be removed from $\mathbf{H}_p$ whereas in the last section the new row was inserted in a fixed position (right at the end of $\mathbf{H}_p$ just after the p-th row).

As noted before, however, the ordering of examples does not matter, so the variance matrix (appendix A.5) is invariant to a permutation of the rows of the

design matrix (equation 4.3). Consequently the relation between $\mathbf{A}_{p-1}$ and $\mathbf{A}_p$ is obtained from equation A.8 by a change of subscripts. If the removed row is $\mathbf{h}_i$ then

$$\mathbf{A}_p^{-1} = \mathbf{A}_{p-1}^{-1} - \frac{\mathbf{A}_{p-1}^{-1}\mathbf{h}_i\,\mathbf{h}_i^\top\mathbf{A}_{p-1}^{-1}}{1+\mathbf{h}_i^\top\mathbf{A}_{p-1}^{-1}\mathbf{h}_i}\,. \tag{A.10}$$

This equation gives the old known $\mathbf{A}_p$ in terms of the new unknown $\mathbf{A}_{p-1}$ and not the other way around, which would be more useful. However, it is easy to invert the relation ship by first deriving from it that

$$\begin{aligned}\mathbf{A}_p^{-1}\mathbf{h}_i &= \frac{\mathbf{A}_{p-1}^{-1}\mathbf{h}_i}{1+\mathbf{h}_i^\top\mathbf{A}_{p-1}^{-1}\mathbf{h}_i}\,,\\ \mathbf{h}_i^\top\mathbf{A}_p^{-1}\mathbf{h}_i &= \frac{\mathbf{h}_i^\top\mathbf{A}_{p-1}^{-1}\mathbf{h}_i}{1+\mathbf{h}_i^\top\mathbf{A}_{p-1}^{-1}\mathbf{h}_i}\,,\end{aligned}$$

from which it follows that

$$\begin{aligned}\mathbf{h}_i^\top\mathbf{A}_{p-1}^{-1}\mathbf{h}_i &= \frac{\mathbf{h}_i^\top\mathbf{A}_p^{-1}\mathbf{h}_i}{1-\mathbf{h}_i^\top\mathbf{A}_p^{-1}\mathbf{h}_i}\,,\\ \mathbf{A}_{p-1}^{-1}\mathbf{h}_i &= \frac{\mathbf{A}_p^{-1}\mathbf{h}_i}{1-\mathbf{h}_i^\top\mathbf{A}_p^{-1}\mathbf{h}_i}\,.\end{aligned}$$

Substituting these in equation A.10 gives

$$\mathbf{A}_{p-1}^{-1} = \mathbf{A}_p^{-1} + \frac{\mathbf{A}_p^{-1}\mathbf{h}_i\,\mathbf{h}_i^\top\mathbf{A}_p^{-1}}{1-\mathbf{h}_i^\top\mathbf{A}_p^{-1}\mathbf{h}_i}\,. \tag{A.11}$$

A.8 The Effective Number of Parameters

In the case of ordinary least squares (no regularisation) the variance matrix (appendix A.5) is

$$\mathbf{A}^{-1} = \left(\mathbf{H}^\top\mathbf{H}\right)^{-1}$$

so the effective number of parameters (equation 4.10) is

$$
\begin{aligned}
\gamma &= p - \text{trace}\left(\mathbf{I}_p - \mathbf{H}\,\mathbf{A}^{-1}\,\mathbf{H}^\top\right) \\
&= \text{trace}\left(\mathbf{H}\,\mathbf{A}^{-1}\,\mathbf{H}^\top\right) \\
&= \text{trace}\left(\mathbf{A}^{-1}\,\mathbf{H}^\top\mathbf{H}\right) \\
&= \text{trace}\left(\left(\mathbf{H}^\top\mathbf{H}\right)^{-1}\mathbf{H}^\top\mathbf{H}\right) \\
&= \text{trace}\left(\mathbf{I}_m\right) \\
&= m\,,
\end{aligned}
$$

so no surprise there: without regularisation there are m parameters, just the number of weights in the network. In standard ridge regression (section 6.1)

$$
\mathbf{A}^{-1} = \left(\mathbf{H}^\top\mathbf{H} + \lambda\,\mathbf{I}_m\right)^{-1}
$$

so the effective number of parameters (equation 4.10) is

$$
\begin{aligned}
\gamma &= \text{trace}\left(\mathbf{A}^{-1}\,\mathbf{H}^\top\mathbf{H}\right) \\
&= \text{trace}\left(\mathbf{A}^{-1}\left(\mathbf{A} - \lambda\,\mathbf{I}_m\right)\right) \\
&= \text{trace}\left(\mathbf{I}_m - \lambda\,\mathbf{A}^{-1}\right) \\
&= m - \lambda\,\text{trace}\left(\mathbf{A}^{-1}\right)\,.
\end{aligned}
$$

This is the same as MacKay's equation (4.9) in [17]. If the eigenvalues of the matrix $\mathbf{H}^\top\mathbf{H}$ are $\{\mu_j\}_{j=1}^m$ then

$$
\begin{aligned}
\gamma &= m - \lambda\,\text{trace}\left(\mathbf{A}^{-1}\right) \\
&= m - \lambda\sum_{j=1}^{m}\frac{1}{\lambda+\mu_j} \\
&= \sum_{j=1}^{m}\frac{\mu_j}{\lambda+\mu_j}\,,
\end{aligned}
$$

which is the same as Moody's equation (18) in [18].

A.9 Leave-one-out Cross-validation

We prove equation 5.1 for leave-one-out (LOO) cross-validation error prediction.

Let $f_i(\mathbf{x}_i)$ be the prediction of the model for the i-th pattern in the training set (section 2) after it has been trained on the $p-1$ other patterns. Then leave-one-out

(LOO) cross-validation [1] predicts the error variance

$$\hat{\sigma}^2_{\mathrm{LOO}} \;=\; \frac{1}{p}\sum_{i=1}^{p}\left(\hat{y}_i - f_i(\mathbf{x}_i)\right)^2 \;.$$

Let $\mathbf{H}_i$ be the design matrix (equation 4.3) of the reduced training set, $\mathbf{A}_i^{-1}$ be its variance matrix (equation 6.7) and $\hat{\mathbf{y}}_i$ its output vector so that the optimal weight (equation 6.8) is

$$\hat{\mathbf{w}}_i \;=\; \mathbf{A}_i^{-1}\mathbf{H}_i^\top \hat{\mathbf{y}}_i \;,$$

and therefore the prediction error for the i-th pattern is

$$\begin{aligned}\hat{y}_i - f_i(\mathbf{x}_i) \;&=\; \hat{y}_i - \hat{\mathbf{w}}_i^\top \mathbf{h}_i \\ &=\; \hat{y}_i - \hat{\mathbf{y}}_i^\top \mathbf{H}_i\, \mathbf{A}_i^{-1}\mathbf{h}_i \;,\end{aligned}$$

where $\mathbf{h}_i = [h_1(\mathbf{x}_i)\;\; h_2(\mathbf{x}_i)\;\; \ldots\;\; h_m(\mathbf{x}_i)\;]^\top$.

$\mathbf{H}_i$ and $\hat{\mathbf{y}}_i$ are both obtained by removing the i-th rows from their counterparts for the full training set, $\mathbf{H}$ and $\hat{\mathbf{y}}$. Consequently,

$$\mathbf{H}_i^\top \hat{\mathbf{y}}_i \;=\; \mathbf{H}^\top \hat{\mathbf{y}} - \hat{y}_i\, \mathbf{h}_i \;.$$

where $\mathbf{h}_i^\top$ is the row removed from the matrix $\mathbf{H}$ and $\hat{y}_i$ is the component taken from the vector $\hat{\mathbf{y}}$. In addition, $\mathbf{A}_i^{-1}$ is related to its counterpart, $\mathbf{A}^{-1}$, by equation A.11 for removing a training example, namely

$$\mathbf{A}_i^{-1} \;=\; \mathbf{A}^{-1} + \frac{\mathbf{A}^{-1}\,\mathbf{h}_i\,\mathbf{h}_i^\top \mathbf{A}^{-1}}{1 - \mathbf{h}_i^\top \mathbf{A}^{-1}\,\mathbf{h}_i} \;.$$

From this it is easy to verify that

$$\mathbf{A}_i^{-1}\mathbf{h}_i \;=\; \frac{\mathbf{A}^{-1}\,\mathbf{h}_i}{1 - \mathbf{h}_i^\top \mathbf{A}^{-1}\,\mathbf{h}_i} \;.$$

Substituting this expression, for $\mathbf{A}_i^{-1}\mathbf{h}_i$, and the one for $\mathbf{H}_i^\top \hat{\mathbf{y}}_i$ into the formula for the prediction error gives

$$\hat{y}_i - f_i(\mathbf{x}_i) \;=\; \frac{\hat{y}_i - \hat{\mathbf{y}}^\top \mathbf{H}\,\mathbf{A}^{-1}\,\mathbf{h}_i}{1 - \mathbf{h}_i^\top \mathbf{A}^{-1}\,\mathbf{h}_i} \;.$$

The numerator of this ratio is the i-th component of the vector $\mathbf{P}\,\hat{\mathbf{y}}$ and the denominator is the i-th component of the diagonal of $\mathbf{P}$ where $\mathbf{P}$ is the projection matrix (appendix 6.9). Therefore the full vector of prediction errors is

$$\begin{bmatrix} \hat{y}_1 - f_1(\mathbf{x}_1) \\ \hat{y}_2 - f_2(\mathbf{x}_2) \\ \vdots \\ \hat{y}_p - f_p(\mathbf{x}_p) \end{bmatrix} \;=\; \left(\mathrm{diag}\,(\mathbf{P})\right)^{-1}\mathbf{P}\,\hat{\mathbf{y}} \;.$$

The matrix $\mathrm{diag}\,(\mathbf{P})$ is the same size and has the same diagonal as $\mathbf{P}$ but is zero off the diagonal. LOO is the mean of the square of the p prediction errors (i.e. the dot product of the above vector of errors with itself divided by p) and so we finally arrive at

$$\hat{\sigma}^2_{\mathrm{LOO}} \;=\; \frac{1}{p}\,\hat{\mathbf{y}}^\top \mathbf{P}\,(\mathrm{diag}\,(\mathbf{P}))^{-2}\,\mathbf{P}\,\hat{\mathbf{y}}\,,$$

which is equation 5.1 which we set out to prove.

A.10 A Re-Estimation Formula for the Global Parameter

The GCV error prediction (section 5.2) is

$$\hat{\sigma}^2 \;=\; \frac{p\,\hat{\mathbf{y}}^\top \mathbf{P}^2\,\hat{\mathbf{y}}}{(\mathrm{trace}\,(\mathbf{P}))^2}\,,$$

where $\mathbf{P}$ is the projection matrix (equation 6.4) for standard ridge regression. To find the minimum of this error, as a function of the regularisation parameter λ, we differentiate it with respect to λ and set the result equal to zero. To perform the differentiation we will have to differentiate the variance matrix (equation 6.2) upon which $\mathbf{P}$ depends. To do this, assume that the matrix $\mathbf{H}$ has the singular value decomposition (appendix A.2)

$$\mathbf{H} \;=\; \mathbf{U}\,\mathbf{\Sigma}\,\mathbf{V}^\top\,,$$

where $\mathbf{U}$ and $\mathbf{V}$ are orthogonal and $\mathbf{\Sigma}$ is

$$\mathbf{\Sigma} \;=\; \begin{bmatrix} \sqrt{\mu_1} & 0 & \dots & 0 \\ 0 & \sqrt{\mu_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \sqrt{\mu_m} \\ 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots \\ 0 & 0 & \dots & 0 \end{bmatrix}\,.$$

$\{\sqrt{\mu_j}\}_{j=1}^m$ are the singular values of $\mathbf{H}$ and $\{\mu_j\}_{j=1}^m$ are the eigenvalues of $\mathbf{H}^\top\mathbf{H}$. We assume $\mathbf{H}$ has more rows than columns ($p > m$) though the same results hold for the opposite case too. It then follows that

$$\mathbf{H}^\top\mathbf{H} \;=\; \mathbf{V}\,\mathbf{\Sigma}^\top\mathbf{\Sigma}\,\mathbf{V}^\top\,,$$

and therefore that

$$\begin{aligned} \mathbf{A}^{-1} &= \left(\mathbf{V}\,\mathbf{\Sigma}^\top\mathbf{\Sigma}\,\mathbf{V}^\top + \lambda\,\mathbf{I}_m\right)^{-1} \\ &= \left(\mathbf{V}\,\mathbf{\Sigma}^\top\mathbf{\Sigma}\,\mathbf{V}^\top + \lambda\,\mathbf{V}\,\mathbf{V}^\top\right)^{-1} \\ &= \left(\mathbf{V}\,(\mathbf{\Sigma}^\top\mathbf{\Sigma} + \lambda\,\mathbf{I}_m)\,\mathbf{V}^\top\right)^{-1} \\ &= \mathbf{V}\,\left(\mathbf{\Sigma}^\top\mathbf{\Sigma} + \lambda\,\mathbf{I}_m\right)^{-1}\mathbf{V}^\top\,, \end{aligned}$$

where we have made use of some useful matrix properties (appendix A.2). The inverse is easy because the matrix is diagonal,

$$\left(\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma}+\lambda\,\mathbf{I}_m\right)^{-1} = \begin{bmatrix} \frac{1}{\lambda+\mu_1} & 0 & \dots & 0 \\ 0 & \frac{1}{\lambda+\mu_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{1}{\lambda+\mu_m} \end{bmatrix},$$

and is the only part of $\mathbf{A}^{-1}$ which depends on λ. Its derivative is also straight forward,

$$\begin{aligned} \frac{\partial}{\partial\lambda}\left(\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma}+\lambda\,\mathbf{I}_m\right)^{-1} &= \begin{bmatrix} \frac{-1}{(\lambda+\mu_1)^2} & 0 & \dots & 0 \\ 0 & \frac{-1}{(\lambda+\mu_2)^2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{-1}{(\lambda+\mu_m)^2} \end{bmatrix} \\ &= -\left(\left(\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma}+\lambda\,\mathbf{I}_m\right)^{-1}\right)^2 \\ &= -\left(\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma}+\lambda\,\mathbf{I}_m\right)^{-2}, \end{aligned}$$

and consequently

$$\begin{aligned} \frac{\partial\mathbf{A}^{-1}}{\partial\lambda} &= \mathbf{V}\frac{\partial}{\partial\lambda}\left(\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma}+\lambda\,\mathbf{I}_m\right)^{-1}\mathbf{V}^\top \\ &= -\mathbf{V}\left(\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma}+\lambda\,\mathbf{I}_m\right)^{-2}\mathbf{V}^\top \\ &= -\mathbf{V}\left(\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma}+\lambda\,\mathbf{I}_m\right)^{-1}\mathbf{V}^\top\mathbf{V}\left(\boldsymbol{\Sigma}^\top\boldsymbol{\Sigma}+\lambda\,\mathbf{I}_m\right)^{-1}\mathbf{V}^\top \\ &= -\mathbf{A}^{-2}. \end{aligned}$$

At this point the reader may think that all this complication just goes to show that matrices can be treated like scalars and that the derivative of $\mathbf{A}^{-1}$ is obviously

$$\begin{aligned} \frac{\partial\mathbf{A}^{-1}}{\partial\lambda} &= -\mathbf{A}^{-2}\frac{\partial\mathbf{A}}{\partial\lambda} \\ &= -\mathbf{A}^{-2}\,\mathbf{I}_m \\ &= -\mathbf{A}^{-2}. \end{aligned}$$

However, I don't think this is true in general. The reader might like to try differentiating the variance matrix (equation 6.7) for local ridge regression with respect to λ_j if there is any doubt about this point.

Now that we know the derivative of $\mathbf{A}^{-1}$ we can proceed to the derivative of the projection matrix (equation 6.4) and the other derivatives we need to differentiate the GCV error prediction (section 5.2).

$$\begin{aligned}\frac{\partial \mathbf{P}}{\partial \lambda} &= \frac{\partial}{\partial \lambda}\left(\mathbf{I}_p - \mathbf{H}\,\mathbf{A}^{-1}\,\mathbf{H}^\top\right)\\ &= -\,\mathbf{H}\,\frac{\partial \mathbf{A}^{-1}}{\partial \lambda}\,\mathbf{H}^\top\\ &= \mathbf{H}\,\mathbf{A}^{-2}\,\mathbf{H}^\top\,.\end{aligned}$$

Similarly,

$$\begin{aligned}\frac{\partial}{\partial \lambda}\,\mathrm{trace}\,(\mathbf{P}) &= -\,\mathrm{trace}\left(\mathbf{H}\,\frac{\partial \mathbf{A}^{-1}}{\partial \lambda}\,\mathbf{H}^\top\right)\\ &= \mathrm{trace}\left(\mathbf{H}\,\mathbf{A}^{-2}\,\mathbf{H}^\top\right)\\ &= \mathrm{trace}\left(\mathbf{A}^{-2}\,\mathbf{H}^\top\mathbf{H}\right)\\ &= \mathrm{trace}\left(\mathbf{A}^{-2}\,(\mathbf{A} - \lambda\,\mathbf{I}_m)\right)\\ &= \mathrm{trace}\left(\mathbf{A}^{-1} - \lambda\,\mathbf{A}^{-2}\right)\,,\end{aligned}$$

and also

$$\begin{aligned}\frac{\partial}{\partial \lambda}\,\hat{\mathbf{y}}^\top\mathbf{P}^2\,\hat{\mathbf{y}} &= 2\,\hat{\mathbf{y}}^\top\mathbf{P}\,\frac{\partial \mathbf{P}}{\partial \lambda}\,\hat{\mathbf{y}}\\ &= 2\,\hat{\mathbf{y}}^\top\mathbf{P}\,\mathbf{H}\,\mathbf{A}^{-2}\,\mathbf{H}^\top\hat{\mathbf{y}}\\ &= 2\,\lambda\,\hat{\mathbf{y}}^\top\mathbf{H}\,\mathbf{A}^{-3}\,\mathbf{H}^\top\hat{\mathbf{y}}\\ &= 2\,\lambda\,\hat{\mathbf{w}}^\top\mathbf{A}^{-1}\,\hat{\mathbf{w}}\,.\end{aligned}$$

The ingredients for differentiating the GCV error prediction (section 5.2) are now all in place.

$$\frac{\partial \hat{\sigma}^2}{\partial \lambda} = \frac{p}{(\mathrm{trace}\,(\mathbf{P}))^2}\,\frac{\partial}{\partial \lambda}\,\hat{\mathbf{y}}^\top\mathbf{P}^2\,\hat{\mathbf{y}} - \frac{2\,p\,\hat{\mathbf{y}}^\top\mathbf{P}^2\,\hat{\mathbf{y}}}{(\mathrm{trace}\,(\mathbf{P}))^3}\,\frac{\partial}{\partial \lambda}\,\mathrm{trace}\,(\mathbf{P}) =$$

$$\frac{2\,p}{(\mathrm{trace}\,(\mathbf{P}))^3}\,\left\{\lambda\,\hat{\mathbf{w}}^\top\mathbf{A}^{-1}\,\hat{\mathbf{w}}\,\mathrm{trace}\,(\mathbf{P}) - \hat{\mathbf{y}}^\top\mathbf{P}^2\,\hat{\mathbf{y}}\,\mathrm{trace}\left(\mathbf{A}^{-1} - \lambda\,\mathbf{A}^{-2}\right)\right\}\,.$$

The last step is to equate this to zero which finally leaves us with

$$\hat{\lambda}\,\hat{\mathbf{w}}^\top\mathbf{A}^{-1}\,\hat{\mathbf{w}}\,\mathrm{trace}\,(\mathbf{P}) = \hat{\mathbf{y}}^\top\mathbf{P}^2\,\hat{\mathbf{y}}\,\mathrm{trace}\left(\mathbf{A}^{-1} - \hat{\lambda}\,\mathbf{A}^{-2}\right)\,,$$

at the optimal value, $\hat{\lambda}$, of the regularisation parameter, λ. Isolating $\hat{\lambda}$ on the left hand side leads to the re-estimation formula (equation 6.5) that we set out to prove, namely

$$\hat{\lambda} = \frac{\hat{\mathbf{y}}^\top\mathbf{P}^2\,\hat{\mathbf{y}}\,\mathrm{trace}\left(\mathbf{A}^{-1} - \hat{\lambda}\,\mathbf{A}^{-2}\right)}{\hat{\mathbf{w}}^\top\mathbf{A}^{-1}\,\hat{\mathbf{w}}\,\mathrm{trace}\,(\mathbf{P})}\,.$$

The other GCV-like model selection criteria (section 5.2) lead to similar formulae. Using $p - \gamma$ in place of trace $(\mathbf{P})$, where γ is the effective number of parameters (equation 4.10), we get

$$\text{(UEV)} \qquad \hat{\lambda} \;=\; \frac{\hat{\mathbf{y}}^\top \mathbf{P}^2\, \hat{\mathbf{y}}\, \text{trace}\left(\mathbf{A}^{-1} - \hat{\lambda}\, \mathbf{A}^{-2}\right)}{\hat{\mathbf{w}}^\top \mathbf{A}^{-1}\, \hat{\mathbf{w}}}\, \frac{1}{2\,(p-\gamma)}$$

$$\text{(FPE)} \qquad \hat{\lambda} \;=\; \frac{\hat{\mathbf{y}}^\top \mathbf{P}^2\, \hat{\mathbf{y}}\, \text{trace}\left(\mathbf{A}^{-1} - \hat{\lambda}\, \mathbf{A}^{-2}\right)}{\hat{\mathbf{w}}^\top \mathbf{A}^{-1}\, \hat{\mathbf{w}}}\, \frac{p}{(p-\gamma)\,(p+\gamma)}$$

$$\text{(GCV)} \qquad \hat{\lambda} \;=\; \frac{\hat{\mathbf{y}}^\top \mathbf{P}^2\, \hat{\mathbf{y}}\, \text{trace}\left(\mathbf{A}^{-1} - \hat{\lambda}\, \mathbf{A}^{-2}\right)}{\hat{\mathbf{w}}^\top \mathbf{A}^{-1}\, \hat{\mathbf{w}}}\, \frac{1}{(p-\gamma)}$$

$$\text{(BIC)} \qquad \hat{\lambda} \;=\; \frac{\hat{\mathbf{y}}^\top \mathbf{P}^2\, \hat{\mathbf{y}}\, \text{trace}\left(\mathbf{A}^{-1} - \hat{\lambda}\, \mathbf{A}^{-2}\right)}{\hat{\mathbf{w}}^\top \mathbf{A}^{-1}\, \hat{\mathbf{w}}}\, \frac{p\,\log(p)}{2\,(p-\gamma)\,(p+(log(p)-1)\,\gamma)}\,.$$

Leave-one-out cross-validation (section 5.1) involves the awkward term diag $(\mathbf{P})$ and I have not managed to find a re-estimation formula for it.

A.11 Optimal Values for the Local Parameters

We take as our starting point equation A.6 for removing a basis function,

$$\mathbf{P} \;=\; \mathbf{P}_j - \frac{1}{\Delta_j}\, \mathbf{P}_j\, \mathbf{h}_j\, \mathbf{h}_j^\top \mathbf{P}_j\,,$$

where

$$\Delta_j \;=\; \lambda_j + \mathbf{h}_j^\top \mathbf{P}_j\, \mathbf{h}_j\,,$$

$\mathbf{P}_j$ is the projection matrix after the j-th basis function has been removed and $\mathbf{h}_j$ is the j-th column of the design matrix (equation 4.3). Actually, we are not going to use this equation to remove any basis functions (unless $\lambda_j = \infty$ — see below) we simply employ it to make the dependence of $\mathbf{P}$ on λ_j explicit.

The GCV criterion (equation 5.2), which we wish to minimise, is

$$\hat{\sigma}^2_{\text{GCV}} \;=\; \frac{p\, \hat{\mathbf{y}}^\top \mathbf{P}^2 \hat{\mathbf{y}}}{(\text{trace}\,(\mathbf{P}))^2}\,.$$

As will shortly become apparent, this function is a rational polynomial of order 2 in λ_j and fairly easy to analyse (i.e. to find its minimum for $\lambda_j \geq 0$). In principle any of the other model selection criteria (section 5) could be used instead of GCV but they lead to higher-order rational polynomials which are not so simple. That is why, as mentioned earlier (section 6.2), GCV is so nice.

Substituting $\mathbf{P}$ in terms of $\mathbf{P}_j$ into GCV

$$\hat{\sigma}^2(\lambda_j) \;=\; \frac{p\,(a\,\Delta_j^2 - 2\,b\,\Delta_j + c)}{(\alpha\,\Delta_j - \beta)^2}\,,$$

where

$$\begin{aligned} a &= \mathbf{y}^\top \mathbf{P}_j^2\,\mathbf{y}\,, \\ b &= \mathbf{y}^\top \mathbf{P}_j^2\,\mathbf{h}_j\,\mathbf{y}^\top \mathbf{P}_j\,\mathbf{h}_j\,, \\ c &= \mathbf{h}_j^\top \mathbf{P}_j^2\,\mathbf{h}_j\,(\mathbf{y}^\top \mathbf{P}_j\,\mathbf{h}_j)^2\,, \\ \alpha &= \mathrm{trace}\,(\mathbf{P}_j)\,, \\ \beta &= \mathbf{h}_j^\top \mathbf{P}_j^2\,\mathbf{h}_j\,. \end{aligned}$$

This is how GCV depends on λ_j when all the other regularisation parameters are held constant — a rational polynomial in Δ_j (or λ_j). It has a pole (becomes infinite) at $\Delta_j = \beta/\alpha$ which never occurs at a positive value of λ_j since it is always true that

$$\mathbf{h}_j^\top \mathbf{P}_j\,\mathbf{h}_j \geq \frac{\mathbf{h}_j^\top \mathbf{P}_j^2\,\mathbf{h}_j}{\mathrm{trace}\,(\mathbf{P}_j)}\,,$$

(the proof is left as an exercise for the reader). The single minimum can be calculated by differentiating $\hat{\sigma}^2$ with respect to λ_j.

$$\begin{aligned} \frac{\partial \hat{\sigma}^2}{\partial \lambda_j} \;=\; \frac{\partial \hat{\sigma}^2}{\partial \Delta_j}\,\frac{\partial \Delta_j}{\partial \lambda_j} \;&=\; \frac{\partial \hat{\sigma}^2}{\partial \Delta_j} \\ &= \frac{2\,p\,(a\,\Delta_j - b)}{(\alpha\,\Delta_j - \beta)^2} - \frac{2\,p\,\alpha\,(a\,\Delta_j^2 - 2\,b\,\Delta_j + c)}{(\alpha\,\Delta_j - \beta)^3} \\ &= \frac{2\,p}{(\alpha\,\Delta_j - \beta)^3}\,((b\,\alpha - a\,\beta)\,\Delta_j - (c\,\alpha - b\,\beta))\,. \end{aligned}$$

If $b\,\alpha = a\,\beta$ this derivative will only attain the value zero at $\Delta_j = \pm\infty$ ($\lambda_j = \pm\infty$). Since we are only interested in $\lambda_j \geq 0$ the solution of interest in that case is $\hat{\lambda}_j = \infty$. Assuming that $b\,\alpha \neq a\,\beta$, when the derivative is set equal to zero the solution is

$$\hat{\Delta}_j \;=\; \frac{c\,\alpha - b\,\beta}{b\,\alpha - a\,\beta}\,,$$

or

$$\hat{\lambda}_j \;=\; \frac{c\,\alpha - b\,\beta}{b\,\alpha - a\,\beta} - \mathbf{h}_j^\top \mathbf{P}_j\,\mathbf{h}_j\,,$$

and this can be either positive or negative. If it is positive we are done – we have found the minimum error prediction for $\lambda_j > 0$. If it is negative there are two cases. Firstly, if

$$\frac{\beta}{\alpha} > \frac{b}{a}$$

the pole (at $\Delta_j = \beta/\alpha$) occurs to the right of the minimum. As λ_j goes from $-\infty$ to ∞ the derivative of the error prediction goes through the sequence

$$\text{negative} \rightarrow \text{minimum} \rightarrow \text{positive} \rightarrow \text{pole} \rightarrow \text{negative}$$

(see figure 16(a)). Therefore the error prediction is falling as λ_j passes through zero (after the pole) and keeps on decreasing thereafter. The minimum error for $\lambda_j \geq 0$ must be at $\hat{\lambda}_j = \infty$. Alternatively, if

$$\frac{\beta}{\alpha} < \frac{b}{a}$$

the pole occurs to the left of the minimum. As λ_j goes from $-\infty$ to ∞ the derivative of the error prediction goes through the sequence

$$\text{positive} \rightarrow \text{pole} \rightarrow \text{negative} \rightarrow \text{minimum} \rightarrow \text{positive}$$

(see figure 16(b)). Therefore the error prediction is rising as λ_j passes through zero (after the minimum) and keeps on increasing thereafter. The minimum error for $\lambda_j \geq 0$ must be at $\hat{\lambda}_j = 0$. In summary

$$\hat{\lambda}_j = \begin{cases} \infty & \text{if } a\,\beta = \alpha\,b \\ \infty & \text{if } \mathbf{h}_j^\top \mathbf{P}_j\,\mathbf{h}_j > \frac{c\,\alpha - b\,\beta}{b\,\alpha - a\,\beta} \text{ and } a\,\beta > \alpha\,b \\ 0 & \text{if } \mathbf{h}_j^\top \mathbf{P}_j\,\mathbf{h}_j > \frac{c\,\alpha - b\,\beta}{b\,\alpha - a\,\beta} \text{ and } a\,\beta < \alpha\,b \\ \frac{c\,\alpha - b\,\beta}{b\,\alpha - a\,\beta} - \mathbf{h}_j^\top \mathbf{P}_j\,\mathbf{h}_j & \text{if } \mathbf{h}_j^\top \mathbf{P}_j\,\mathbf{h}_j \leq \frac{c\,\alpha - b\,\beta}{b\,\alpha - a\,\beta} \end{cases}$$

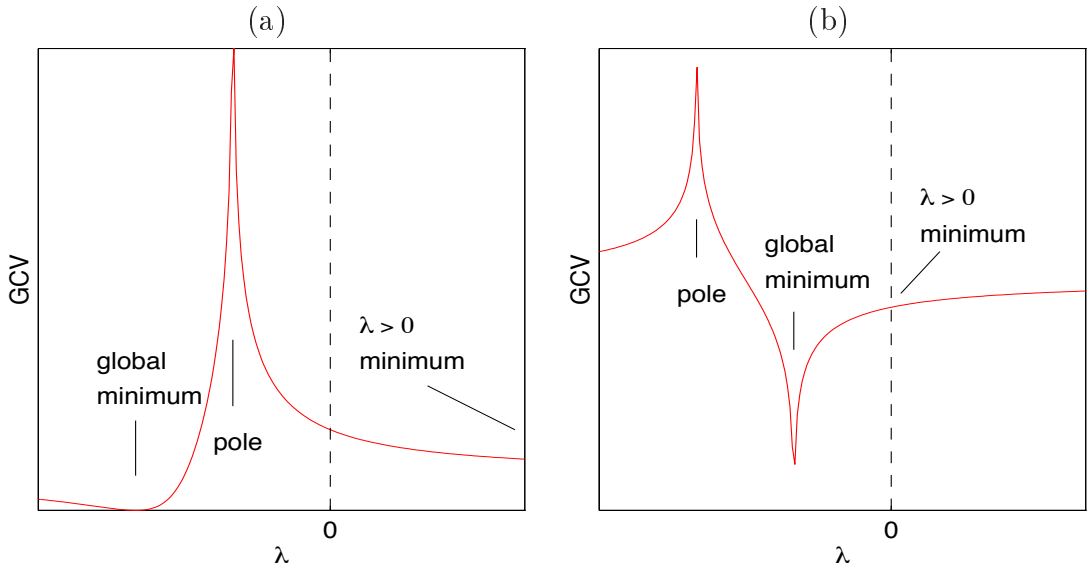

Figure 16: When the global minimum is negative either (a) $a\,\beta > b\,\alpha$ and the minimum for nonnegative λ_j is $\hat{\lambda}_j = \infty$ or (b) $a\,\beta < b\,\alpha$ and the minimum for nonnegative λ_j is $\hat{\lambda}_j = 0$.

Each individual optimisation is guaranteed to reduce, or at least not increase, the GCV error prediction. All m regularisation parameters are optimised one after the other and, if necessary, repeatedly until GCV has stopped decreasing or is only decreasing by a very small amount. Naturally, the order in which the parameters are optimised as well as their initial values will affect which local minimum is reached.

The computational requirements of optimisation in local ridge regression are heavier than standard ridge regression (section 6.2) or forward selection (section 7) since $\mathbf{P}$ changes and has to be updated after each individual optimisation. Some increase in efficiency can be gained by updating only the quantities needed to calculate the five coefficients (a, b, c, α and β) such as $\mathbf{P}\,\hat{\mathbf{y}}$ and $\mathbf{P}\,\mathbf{H}$, rather than $\mathbf{P}$ itself [19].

A.12 Forward Selection

To prove equation 7.2 for calculating the change in sum-squared-error with standard ridge regression (section 6.1), equation 4.8 for the sum-squared error,

$$\begin{aligned} \hat{S}_m &= \hat{\mathbf{y}}^\top \mathbf{P}_m^2\, \hat{\mathbf{y}}\,, \\ \hat{S}_{m+1} &= \hat{\mathbf{y}}^\top \mathbf{P}_{m+1}^2\, \hat{\mathbf{y}}\,, \end{aligned}$$

is combined with equation 7.1 for updating the projection matrix. In the absence of any regularisation $\mathbf{P}_m$ and $\mathbf{P}_{m+1}$ are true projection matrices and so are idempotent ($\mathbf{P}^2 = \mathbf{P}$). Therefore

$$\begin{aligned} \hat{S}_m &= \hat{\mathbf{y}}^\top \mathbf{P}_m\, \hat{\mathbf{y}}\,, \\ \hat{S}_{m+1} &= \hat{\mathbf{y}}^\top \mathbf{P}_{m+1}\, \hat{\mathbf{y}}\,, \end{aligned}$$

and

$$\begin{aligned} \hat{S}_m - \hat{S}_{m+1} &= \hat{\mathbf{y}}^\top \left(\mathbf{P}_m - \mathbf{P}_{m+1}\right) \hat{\mathbf{y}} \\ &= \hat{\mathbf{y}}^\top \frac{\mathbf{P}_m\, \mathbf{f}_J\, \mathbf{f}_J^\top \mathbf{P}_m}{\mathbf{f}_J^\top \mathbf{P}_m\, \mathbf{f}_J}\, \hat{\mathbf{y}} \\ &= \frac{(\hat{\mathbf{y}}^\top \mathbf{P}_m\, \mathbf{f}_J)^2}{\mathbf{f}_J^\top \mathbf{P}_m\, \mathbf{f}_J}\,. \end{aligned}$$

This calculation requires of order p^2 floating point operations and has to be done for all M candidates to find the maximum. Appendix A.13 describes a way of reducing this cost by a factor of p (the number of patterns in the training set (section 2)).

A.13 Orthogonal Least Squares

First we show that the factored form (equation 7.3) of the design matrix (equation 4.3) results in a simple expression for the projection matrix (equation 4.6) and then that this leads to an efficient method of computing the change in sum-squared-error (equation 4.1) due to a new basis function.

Substituting the factored form (equation 7.3) into the projection matrix (equation 4.6) gives

$$\begin{aligned}
\mathbf{P}_m &= \mathbf{I}_p - \tilde{\mathbf{H}}_m \mathbf{U}_m \left(\mathbf{U}_m^\top \tilde{\mathbf{H}}_m^\top \tilde{\mathbf{H}}_m \mathbf{U}_m \right)^{-1} \mathbf{U}_m^\top \tilde{\mathbf{H}}_m^\top \\
&= \mathbf{I}_p - \tilde{\mathbf{H}}_m \left(\tilde{\mathbf{H}}_m^\top \tilde{\mathbf{H}}_m \right)^{-1} \tilde{\mathbf{H}}_m^\top \\
&= \mathbf{I}_p - \sum_{j=1}^{m} \frac{\tilde{\mathbf{h}}_j \tilde{\mathbf{h}}_j^\top}{\tilde{\mathbf{h}}_j^\top \tilde{\mathbf{h}}_j} \,.
\end{aligned}$$

The first step is due to one of the useful properties of matrices (section A.2) and the last step follows because the $\{\tilde{\mathbf{h}}_j\}_{j=1}^m$ — the columns of $\tilde{\mathbf{H}}$ — are mutually orthogonal.

The $(m+1)$-th projection matrix, with an extra column, $\tilde{\mathbf{h}}_{m+1}$, in the design matrix, just adds another term to the sum. If the new column is from the full design matrix, i.e. if $\tilde{\mathbf{h}}_{m+1} = \tilde{\mathbf{f}}_J$, then

$$\mathbf{P}_{m+1} = \mathbf{I}_p - \sum_{j=1}^{m} \frac{\tilde{\mathbf{h}}_j \tilde{\mathbf{h}}_j^\top}{\tilde{\mathbf{h}}_j^\top \tilde{\mathbf{h}}_j} - \frac{\tilde{\mathbf{f}}_J \tilde{\mathbf{f}}_J^\top}{\tilde{\mathbf{f}}_J^\top \tilde{\mathbf{f}}_J} \,,$$

and consequently in the absence of regularisation ($\mathbf{P}^2 = \mathbf{P}$)

$$\begin{aligned}
\hat{S}_m - \hat{S}_{m+1} &= \hat{\mathbf{y}}^\top (\mathbf{P}_m - \mathbf{P}_{m+1}) \, \hat{\mathbf{y}} \\
&= \hat{\mathbf{y}}^\top \frac{\tilde{\mathbf{f}}_J \tilde{\mathbf{f}}_J^\top}{\tilde{\mathbf{f}}_J^\top \tilde{\mathbf{f}}_J} \hat{\mathbf{y}} \\
&= \frac{(\hat{\mathbf{y}}^\top \tilde{\mathbf{f}}_J)^2}{\tilde{\mathbf{f}}_J^\top \tilde{\mathbf{f}}_J} \,.
\end{aligned}$$

This is much faster to compute than the corresponding equation (7.2) without orthogonal least squares. The orthogonalised weight vector is

$$
\begin{aligned}
\tilde{\mathbf{w}}_m &= \left(\tilde{\mathbf{H}}_m^\top \tilde{\mathbf{H}}_m\right)^{-1} \tilde{\mathbf{H}}_m^\top \hat{\mathbf{y}} \\
&= \begin{bmatrix} \frac{1}{\tilde{\mathbf{h}}_1^\top \tilde{\mathbf{h}}_1} & 0 & \dots & 0 \\ 0 & \frac{1}{\tilde{\mathbf{h}}_2^\top \tilde{\mathbf{h}}_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{1}{\tilde{\mathbf{h}}_m^\top \tilde{\mathbf{h}}_m} \end{bmatrix} \begin{bmatrix} \tilde{\mathbf{h}}_1^\top \\ \tilde{\mathbf{h}}_2^\top \\ \vdots \\ \tilde{\mathbf{h}}_m^\top \end{bmatrix} \hat{\mathbf{y}} \,,
\end{aligned}
$$

and its j-th component is therefore

$$
(\tilde{\mathbf{w}}_m)_j = \frac{\hat{\mathbf{y}}^\top \tilde{\mathbf{h}}_j}{\tilde{\mathbf{h}}_j^\top \tilde{\mathbf{h}}_j} \,.
$$

It is related to the unnormalised weight vector by

$$
\begin{aligned}
\hat{\mathbf{w}}_m &= \left(\mathbf{H}_m^\top \mathbf{H}_m\right)^{-1} \mathbf{H}_m^\top \hat{\mathbf{y}} \\
&= \mathbf{U}_m^{-1} \left(\tilde{\mathbf{H}}_m^\top \tilde{\mathbf{H}}_m\right)^{-1} \left(\mathbf{U}_m^\top\right)^{-1} \mathbf{U}_m^\top \tilde{\mathbf{H}}_m^\top \hat{\mathbf{y}} \\
&= \mathbf{U}_m^{-1} \left(\tilde{\mathbf{H}}_m^\top \tilde{\mathbf{H}}_m\right)^{-1} \tilde{\mathbf{H}}_m^\top \hat{\mathbf{y}} \\
&= \mathbf{U}_m^{-1} \, \tilde{\mathbf{w}}_m \,.
\end{aligned}
$$

For the purposes of model selection (section 5)

$$
\begin{aligned}
\operatorname{trace}(\mathbf{P}_m) &= \operatorname{trace}(\mathbf{I}_p) - \sum_{j=1}^{m} \frac{\operatorname{trace}\left(\tilde{\mathbf{h}}_j \, \tilde{\mathbf{h}}_j^\top\right)}{\tilde{\mathbf{h}}_j^\top \tilde{\mathbf{h}}_j} \\
&= p - \sum_{j=1}^{m} \frac{\tilde{\mathbf{h}}_j^\top \tilde{\mathbf{h}}_j}{\tilde{\mathbf{h}}_j^\top \tilde{\mathbf{h}}_j} \\
&= p - m \,,
\end{aligned}
$$

and so, for example, GCV (equation 5.2) is

$$
\begin{aligned}
\hat{\sigma}_m^2 &= \frac{p \, \hat{\mathbf{y}}^\top \mathbf{P}^2 \, \hat{\mathbf{y}}}{\left(\operatorname{trace}(\mathbf{P})\right)^2} \\
&= \frac{p \, \hat{\mathbf{y}}^\top}{(p-m)^2} \left(\mathbf{I}_p - \sum_{j=1}^{m} \frac{\tilde{\mathbf{h}}_j \, \tilde{\mathbf{h}}_j^\top}{\tilde{\mathbf{h}}_j^\top \tilde{\mathbf{h}}_j}\right) \hat{\mathbf{y}} \\
&= \frac{p}{(p-m)^2} \left(\hat{\mathbf{y}}^\top \hat{\mathbf{y}} - \sum_{j=1}^{m} \frac{(\hat{\mathbf{y}}^\top \tilde{\mathbf{h}}_j)^2}{\tilde{\mathbf{h}}_j^\top \tilde{\mathbf{h}}_j}\right) \,.
\end{aligned}
$$

A.14 Regularised Forward Selection

To prove equation 7.6 for calculating the change in sum-squared-error with standard ridge regression (section 6.1) we combine equation 4.8 for the sum-squared error,

$$\begin{aligned} \hat{S}_m &= \hat{\mathbf{y}}^\top \mathbf{P}_m^2 \, \hat{\mathbf{y}} \,, \\ \hat{S}_{m+1} &= \hat{\mathbf{y}}^\top \mathbf{P}_{m+1}^2 \, \hat{\mathbf{y}} \,, \end{aligned}$$

with equation 7.5 for updating the projection matrix, which gives

$$\begin{aligned} \hat{S}_m - \hat{S}_{m+1} &= \hat{\mathbf{y}}^\top \left(\mathbf{P}_m^2 - \left(\mathbf{P}_m - \frac{\mathbf{P}_m \mathbf{f}_J \mathbf{f}_J^\top \mathbf{P}_m}{\lambda + \mathbf{f}_J^\top \mathbf{P}_m \mathbf{f}_J} \right)^2 \right) \hat{\mathbf{y}} \\ &= \frac{2 \, \hat{\mathbf{y}}^\top \mathbf{P}_m^2 \, \mathbf{f}_J \, \hat{\mathbf{y}}^\top \mathbf{P}_m \mathbf{f}_J}{\lambda + \mathbf{f}_J^\top \mathbf{P}_m \mathbf{f}_J} - \frac{(\hat{\mathbf{y}}^\top \mathbf{P}_m \mathbf{f}_J)^2 \, \mathbf{f}_J^\top \mathbf{P}_m^2 \, \mathbf{f}_J}{(\lambda + \mathbf{f}_J^\top \mathbf{P}_m \mathbf{f}_J)^2} \,. \end{aligned}$$

An alternative is to seek to maximise the decrease in the cost function (equation 6.1) which is

$$\begin{aligned} \hat{C}_m - \hat{C}_{m+1} &= \hat{\mathbf{y}}^\top \mathbf{P}_m \, \hat{\mathbf{y}} - \hat{\mathbf{y}} \, \mathbf{P}_{m+1} \, \hat{\mathbf{y}} \\ &= \hat{\mathbf{y}}^\top \frac{\mathbf{P}_m \mathbf{f}_J \mathbf{f}_J^\top \mathbf{P}_m}{\lambda + \mathbf{f}_J^\top \mathbf{P}_m \mathbf{f}_J} \, \hat{\mathbf{y}} \\ &= \frac{(\hat{\mathbf{y}}^\top \mathbf{P}_m \mathbf{f}_J)^2}{\lambda + \mathbf{f}_J^\top \mathbf{P}_m \mathbf{f}_J} \,. \end{aligned}$$

A.15 Regularised Orthogonal Least Squares

If computational speed is important then orthogonal least squares (section 7.1) can be adapted to standard ridge regression (section 6.1), although, as explained in section 7.3, a change of cost function is involved. This appendix gives some more of the details.

The equations are very similar to orthogonal least squares (appendix A.13) except for the extra term λ and the fact that the projection matrix (equation 6.4) is no longer idempotent so, when we calculate the change in sum-squared-error (equation 4.1) we cannot use the simplification $\mathbf{P}_m^2 = \mathbf{P}_m$. Writing $\mathbf{P}_m$ as a sum we get

$$
\begin{aligned}
\mathbf{P}_m &= \mathbf{I}_p - \tilde{\mathbf{H}}_m \left(\tilde{\mathbf{H}}_m^\top \tilde{\mathbf{H}}_m + \lambda \mathbf{I}_m \right)^{-1} \tilde{\mathbf{H}}_m^\top \\
&= \mathbf{I}_p - \tilde{\mathbf{H}}_m \begin{bmatrix} \frac{1}{\lambda + \tilde{\mathbf{h}}_1^\top \tilde{\mathbf{h}}_1} & 0 & \dots & 0 \\ 0 & \frac{1}{\lambda + \tilde{\mathbf{h}}_2^\top \tilde{\mathbf{h}}_2} & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & \frac{1}{\lambda + \tilde{\mathbf{h}}_m^\top \tilde{\mathbf{h}}_m} \end{bmatrix} \tilde{\mathbf{H}}_m^\top \\
&= \mathbf{I}_p - \sum_{j=1}^{m} \frac{\tilde{\mathbf{h}}_j \, \tilde{\mathbf{h}}_j^\top}{\lambda + \tilde{\mathbf{h}}_j^\top \tilde{\mathbf{h}}_j} \, .
\end{aligned}
$$

The $(m+1)$-th projection matrix adds another term to this sum. The change in sum-squared-error (equation 4.1) due to the addition of $\tilde{\mathbf{f}}_J$ from the full design matrix (i.e. if $\mathbf{h}_{m+1} = \tilde{\mathbf{f}}_J$) is then

$$
\begin{aligned}
\hat{S}_m - \hat{S}_{m+1} &= \hat{\mathbf{y}}^\top \left(\mathbf{P}_m^2 - \mathbf{P}_{m+1}^2 \right) \hat{\mathbf{y}} \\
&= \frac{(\hat{\mathbf{y}}^\top \tilde{\mathbf{f}}_J)^2}{\lambda + \tilde{\mathbf{f}}_J^\top \tilde{\mathbf{f}}_J} \, \frac{2\,\lambda + \tilde{\mathbf{f}}_J^\top \tilde{\mathbf{f}}_J}{\lambda + \tilde{\mathbf{f}}_J^\top \tilde{\mathbf{f}}_J} \, .
\end{aligned}
$$

An alternative search criterion, the change in the modified cost function (equation 7.10), is

$$
\hat{C}_m - \hat{C}_{m+1} = \frac{(\hat{\mathbf{y}}^\top \tilde{\mathbf{f}}_J)^2}{\lambda + \tilde{\mathbf{f}}_J^\top \tilde{\mathbf{f}}_J} \, .
$$

The j-th component of the orthogonalised weight vector is

$$
(\tilde{\mathbf{w}}_m)_j = \frac{\hat{\mathbf{y}}^\top \tilde{\mathbf{h}}_j}{\lambda + \tilde{\mathbf{h}}_j^\top \tilde{\mathbf{h}}_j} \, ,
$$

and it is related to the unnormalised weight vector by the same equation we had previously, namely

$$
\hat{\mathbf{w}}_m = \mathbf{U}_m^{-1} \, \tilde{\mathbf{w}}_m \, .
$$

For the purposes of model selection (section 5)

$$
\begin{aligned}
\text{trace}\,(\mathbf{P}_m) &= \text{trace}\,(\mathbf{I}_p) - \sum_{j=1}^{m} \frac{\text{trace}\left(\tilde{\mathbf{h}}_j \, \tilde{\mathbf{h}}_j^\top \right)}{\lambda + \tilde{\mathbf{h}}_j^\top \tilde{\mathbf{h}}_j} \\
&= p - \sum_{j=1}^{m} \frac{\tilde{\mathbf{h}}_j^\top \tilde{\mathbf{h}}_j}{\lambda + \tilde{\mathbf{h}}_j^\top \tilde{\mathbf{h}}_j} \\
&= p - \gamma_m \, ,
\end{aligned}
$$

where γ_m is the effective number of parameters (equation 4.4), in this case

$$\gamma_m \;=\; \sum_{j=1}^{m} \frac{\tilde{\mathbf{h}}_j^\top \tilde{\mathbf{h}}_j}{\lambda + \tilde{\mathbf{h}}_j^\top \tilde{\mathbf{h}}_j} \,.$$

Then GCV (equation 5.6) is

$$\begin{aligned}
\hat{\sigma}_m^2 &= \frac{p\, \hat{\mathbf{y}}^\top \mathbf{P}^2\, \hat{\mathbf{y}}}{\left(\mathrm{trace}\,(\mathbf{P})\right)^2} \\
&= \frac{p}{(p-\gamma_m)^2} \left(\hat{\mathbf{y}}^\top \hat{\mathbf{y}} - \sum_{j=1}^{m} \frac{(\hat{\mathbf{y}}^\top \tilde{\mathbf{h}}_j)^2}{\lambda + \tilde{\mathbf{h}}_j^\top \tilde{\mathbf{h}}_j} \, \frac{2\,\lambda + \tilde{\mathbf{h}}_j^\top \tilde{\mathbf{h}}_j}{\lambda + \tilde{\mathbf{h}}_j^\top \tilde{\mathbf{h}}_j} \right) .
\end{aligned}$$

References

[1] D. M. Allen. The relationship between variable selection and data augmentation and a method for prediction. *Technometrics*, 16(1):125–127, 1974.

[2] L. Breiman and J. Friedman. Predicting multivariate responses in multiple linear regression. Technical report, Department of Statistics, University of California, Berkeley, 1994.

[3] D.S. Broomhead and D. Lowe. Multivariate functional interpolation and adaptive networks. *Complex Systems*, 2:321–355, 1988.

[4] S. Chen, C.F.N. Cowan, and P.M. Grant. Orthogonal least squares learning for radial basis function networks. *IEEE Transactions on Neural Networks*, 2(2):302–309, 1991.

[5] S. Geman, E. Bienenstock, and R. Doursat. Neural networks and the bias/variance dilemma. *Neural Computation*, 4(1):1–58, 1992.

[6] G.H. Golub, M. Heath, and G. Wahba. Generalised cross-validation as a method for choosing a good ridge parameter. *Technometrics*, 21(2):215–223, 1979.

[7] C. Gu and G. Wahba. Minimising GCV/GML scores with multiple smoothing parameters via the Newton method. *SIAM Journal on Scientific and Statistical Computing*, 12(2):383–398, 1991.

[8] J. Hertz, A. Krough, and R.G. Palmer. *Introduction to the Theory of Neural Computation*. Addison Wesley, Redwood City, CA, 1991.

[9] R.R. Hocking. The analysis and selection of variables in linear regression. *Biometrics*, 32:1–49, 1976.

[10] R.R. Hocking. Developments in linear regression methodology: 1959-1982 (with discussion). *Technometrics*, 25:219–249, 1983.

[11] A.E. Hoerl and R.W. Kennard. Ridge regression: Biased estimation for nonorthogonal problems. *Technometrics*, 12(3):55–67, 1970.

[12] R.A. Horn and C.R. Johnson. *Matrix Analysis*. Cambridge University Press, Cambridge, UK, 1985.

[13] A.S. Weigend, A.S.M. Mangeas, and A.N. Srivastava. Nonlinear gated experts for time series: Discovering regimes and avoiding overfitting. *International Journal of Neural Systems*, 6:373–399, 1995.

[14] B. Efron and R.J. Tibshirani. *An Introduction to the Bootstrap*. Chapman and Hall, 1993.

[15] B. Fritzke. Growing cell structures - a self-organizing network for unsupervised and supervised learning. *Neural Networks*, 7(9):1441–1460, 1994.

[16] C. Bishop. Improving the generalisation properties of radial basis function neural networks. *Neural Computation*, 3(4):579–588, 1991.

[17] D.J.C. MacKay. Bayesian interpolation. *Neural Computation*, 4(3):415–447, 1992.

[18] J.E. Moody. The effective number of parameters: An analysis of generalisation and regularisation in nonlinear learning systems. In J.E. Moody, S.J. Hanson, and R.P. Lippmann, editors, *Neural Information Processing Systems 4*, pages 847–854. Morgan Kaufmann, San Mateo CA, 1992.

[19] M.J.L. Orr. Local Smoothing of Radial Basis Function Networks (long version). In *International Symposium on Artificial Neural Networks*, Hsinchu, Taiwan, 1995.

[20] M.J.L. Orr. Regularisation in the Selection of Radial Basis Function Centres. *Neural Computation*, 7(3):606–623, 1995.

[21] V. Kadirkamanathan and M. Niranjan. A function estimation approach to sequential learning with neural networks. *Neural Computation*, 5(6):954–975, 1993.

[22] C. Mallows. Some comments on C_p. *Technometrics*, 15:661–675, 1973.

[23] D. Michie, D.J. Spiegelhalter, and C.C. Taylor, editors. *Machine Learning: Neural and Statistical Classification*. Ellis Horwood, London, 1994.

[24] J. Platt. A resource-allocating network for function interpolation. *Neural Computation*, 3(2):213–225, 1991.

[25] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C*. Cambridge University Press, Cambridge, UK, second edition, 1992.

[26] J.O. Rawlings. *Applied Regression Analysis*. Wadsworth & Brooks/Cole, Pacific Grove, CA, 1988.

[27] W.S. Sarle. Neural Networks and Statistical Models. In *Proceedings of the Nineteenth Annual SAS Users Group International Conference*, pages 1538–1550, Cary, NC, 1994.

[28] G. Schwarz. Estimating the dimension of a model. *Annals of Statistics*, 6:461–464, 1978.

[29] A.N. Tikhonov and V.Y. Arsenin. *Solutions of Ill-Posed Problems*. Winston, Washington, 1977.