7 *Linear regression*

7.1 Introduction

Linear regression is the “work horse” of statistics and (supervised) machine learning. When augmented with kernels or other forms of basis function expansion, it can model also non-linear relationships. And when the Gaussian output is replaced with a Bernoulli or multinoulli distribution, it can be used for classification, as we will see below. So it pays to study this model in detail.

7.2 Model specification

As we discussed in Section 1.4.5, linear regression is a model of the form

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(y|\mathbf{w}^T\mathbf{x}, \sigma^2) \tag{7.1}$$

Linear regression can be made to model non-linear relationships by replacing $\mathbf{x}$ with some non-linear function of the inputs, $\boldsymbol{\phi}(\mathbf{x})$. That is, we use

$$p(y|\mathbf{x}, \boldsymbol{\theta}) = \mathcal{N}(y|\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}), \sigma^2) \tag{7.2}$$

This is known as **basis function expansion**. (Note that the model is still linear in the parameters $\mathbf{w}$, so it is still called linear regression; the importance of this will become clear below.) A simple example are polynomial basis functions, where the model has the form

$$\boldsymbol{\phi}(x) = [1, x, x^2, \ldots, x^d] \tag{7.3}$$

Figure 1.18 illustrates the effect of changing d: increasing the degree d allows us to create increasingly complex functions.

We can also apply linear regression to more than 1 input. For example, consider modeling temperature as a function of location. Figure 7.1(a) plots $\mathbb{E}[y|\mathbf{x}] = w_0 + w_1x_1 + w_2x_2$, and Figure 7.1(b) plots $\mathbb{E}[y|\mathbf{x}] = w_0 + w_1x_1 + w_2x_2 + w_3x_1^2 + w_4x_2^2$.

7.3 Maximum likelihood estimation (least squares)

A common way to esitmate the parameters of a statistical model is to compute the MLE, which is defined as

$$\hat{\boldsymbol{\theta}} \triangleq \arg\max_{\boldsymbol{\theta}} \log p(\mathcal{D}|\boldsymbol{\theta}) \tag{7.4}$$

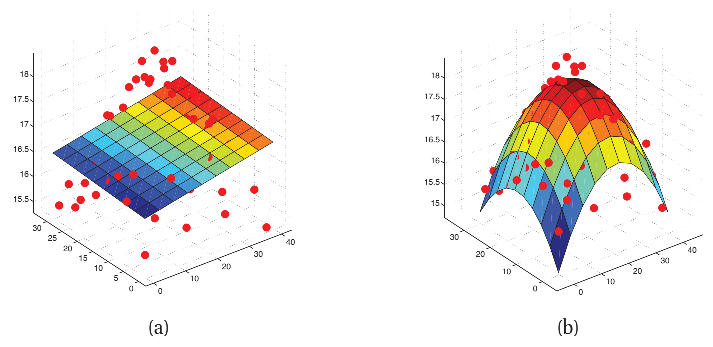

Figure 7.1 Linear regression applied to 2d data. Vertical axis is temperature, horizontal axes are location within a room. Data was collected by some remote sensing motes at Intel's lab in Berkeley, CA (data courtesy of Romain Thibaux). (a) The fitted plane has the form $\hat{f}(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2$. (b) Temperature data is fitted with a quadratic of the form $\hat{f}(\mathbf{x}) = w_0 + w_1 x_1 + w_2 x_2 + w_3 x_1^2 + w_4 x_2^2$. Produced by `surfaceFitDemo`.

It is common to assume the training examples are independent and identically distributed, commonly abbreviated to **iid**. This means we can write the log-likelihood as follows:

$$\ell(\boldsymbol{\theta}) \triangleq \log p(\mathcal{D}|\boldsymbol{\theta}) = \sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \boldsymbol{\theta}) \tag{7.5}$$

Instead of maximizing the log-likelihood, we can equivalently minimize the **negative log likelihood** or **NLL**:

$$\text{NLL}(\boldsymbol{\theta}) \triangleq -\sum_{i=1}^{N} \log p(y_i|\mathbf{x}_i, \boldsymbol{\theta}) \tag{7.6}$$

The NLL formulation is sometimes more convenient, since many optimization software packages are designed to find the minima of functions, rather than maxima.

Now let us apply the method of MLE to the linear regression setting. Inserting the definition of the Gaussian into the above, we find that the log likelihood is given by

$$\begin{aligned} \ell(\boldsymbol{\theta}) &= \sum_{i=1}^{N} \log \left[\left(\frac{1}{2\pi\sigma^2} \right)^{\frac{1}{2}} \exp \left(-\frac{1}{2\sigma^2} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 \right) \right] && (7.7) \\ &= \frac{-1}{2\sigma^2} RSS(\mathbf{w}) - \frac{N}{2} \log(2\pi\sigma^2) && (7.8) \end{aligned}$$

RSS stands for **residual sum of squares** and is defined by

$$\text{RSS}(\mathbf{w}) \triangleq \sum_{i=1}^{N} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 \tag{7.9}$$

The RSS is also called the **sum of squared errors**, or SSE, and SSE$/N$ is called the **mean squared error** or **MSE**. It can also be written as the square of the ℓ_2 **norm** of the vector of

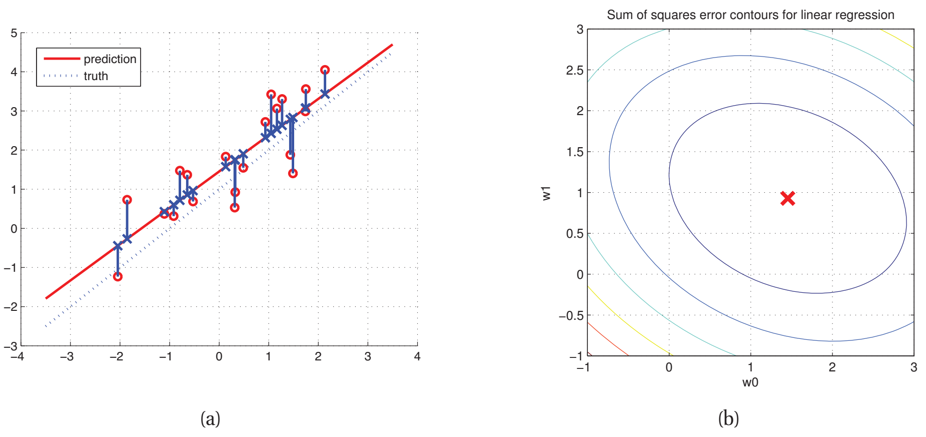

Figure 7.2 (a) In linear least squares, we try to minimize the sum of squared distances from each training point (denoted by a red circle) to its approximation (denoted by a blue cross), that is, we minimize the sum of the lengths of the little vertical blue lines. The red diagonal line represents $\hat{y}(x) = w_0 + w_1 x$, which is the least squares regression line. Note that these residual lines are not perpendicular to the least squares line, in contrast to Figure 12.5. Figure generated by `residualsDemo`. (b) Contours of the RSS error surface for the same example. The red cross represents the MLE, $\mathbf{w} = (1.45, 0.93)$. Figure generated by `contoursSSEdemo`.

residual errors:

$$\text{RSS}(\mathbf{w}) = ||\boldsymbol{\epsilon}||_2^2 = \sum_{i=1}^{N} \epsilon_i^2 \tag{7.10}$$

where $\epsilon_i = (y_i - \mathbf{w}^T \mathbf{x}_i)$.

We see that the MLE for $\mathbf{w}$ is the one that minimizes the RSS, so this method is known as **least squares**. This method is illustrated in Figure 7.2(a). The training data (x_i, y_i) are shown as red circles, the estimated values $(x_i, \hat{y}_i)$ are shown as blue crosses, and the residuals $\epsilon_i = y_i - \hat{y}_i$ are shown as vertical blue lines. The goal is to find the setting of the parameters (the slope w_1 and intercept w_0) such that the resulting red line minimizes the sum of squared residuals (the lengths of the vertical blue lines).

In Figure 7.2(b), we plot the NLL surface for our linear regression example. We see that it is a quadratic "bowl" with a unique minimum, which we now derive. (Importantly, this is true even if we use basis function expansion, such as polynomials, because the NLL is still *linear in the parameters* $\mathbf{w}$, even if it is not linear in the inputs $\mathbf{x}$.)

7.3.1 Derivation of the MLE

First, we rewrite the objective in a form that is more amenable to differentiation:

$$\text{NLL}(\mathbf{w}) = \frac{1}{2}(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) = \frac{1}{2}\mathbf{w}^T(\mathbf{X}^T\mathbf{X})\mathbf{w} - \mathbf{w}^T(\mathbf{X}^T\mathbf{y}) \tag{7.11}$$

where

$$\mathbf{X}^T\mathbf{X} = \sum_{i=1}^N \mathbf{x}_i\mathbf{x}_i^T = \sum_{i=1}^N \begin{pmatrix} x_{i,1}^2 & \cdots & x_{i,1}x_{i,D} \\ & \ddots & \\ x_{i,D}x_{i,1} & \cdots & x_{i,D}^2 \end{pmatrix} \tag{7.12}$$

is the **sum of squares** matrix and

$$\mathbf{X}^T\mathbf{y} = \sum_{i=1}^N \mathbf{x}_i y_i. \tag{7.13}$$

Using results from Equation 4.10, we see that the gradient of this is given by

$$\mathbf{g}(\mathbf{w}) \quad = \quad [\mathbf{X}^T\mathbf{X}\mathbf{w} - \mathbf{X}^T\mathbf{y}] = \sum_{i=1}^N \mathbf{x}_i(\mathbf{w}^T\mathbf{x}_i - y_i) \tag{7.14}$$

Equating to zero we get

$$\mathbf{X}^T\mathbf{X}\mathbf{w} \quad = \quad \mathbf{X}^T\mathbf{y} \tag{7.15}$$

This is known as the **normal equation**. The corresponding solution $\hat{\mathbf{w}}$ to this linear system of equations is called the **ordinary least squares** or **OLS** solution, which is given by

$$\boxed{\hat{\mathbf{w}}_{OLS} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}} \tag{7.16}$$

7.3.2 Geometric interpretation

This equation has an elegant geometrical intrepretation, as we now explain. We assume $N > D$, so we have more examples than features. The columns of $\mathbf{X}$ define a linear subspace of dimensionality D which is embedded in N dimensions. Let the j'th column be $\tilde{\mathbf{x}}_j$, which is a vector in $\mathbb{R}^N$. (This should not be confused with $\mathbf{x}_i \in \mathbb{R}^D$, which represents the i'th data case.) Similarly, $\mathbf{y}$ is a vector in $\mathbb{R}^N$. For example, suppose we have $N = 3$ examples in $D = 2$ dimensions:

$$\mathbf{X} = \begin{pmatrix} 1 & 2 \\ 1 & -2 \\ 1 & 2 \end{pmatrix}, \quad \mathbf{y} = \begin{pmatrix} 8.8957 \\ 0.6130 \\ 1.7761 \end{pmatrix} \tag{7.17}$$

These vectors are illustrated in Figure 7.3.

We seek a vector $\hat{\mathbf{y}} \in \mathbb{R}^N$ that lies in this linear subspace and is as close as possible to $\mathbf{y}$, i.e., we want to find

$$\underset{\hat{\mathbf{y}} \in \text{span}(\{\tilde{\mathbf{x}}_1,\ldots,\tilde{\mathbf{x}}_D\})}{\operatorname{argmin}} \quad \|\mathbf{y} - \hat{\mathbf{y}}\|_2. \tag{7.18}$$

Since $\hat{\mathbf{y}} \in \text{span}(\mathbf{X})$, there exists some weight vector $\mathbf{w}$ such that

$$\hat{\mathbf{y}} = w_1\tilde{\mathbf{x}}_1 + \cdots + w_D\tilde{\mathbf{x}}_D = \mathbf{X}\mathbf{w} \tag{7.19}$$

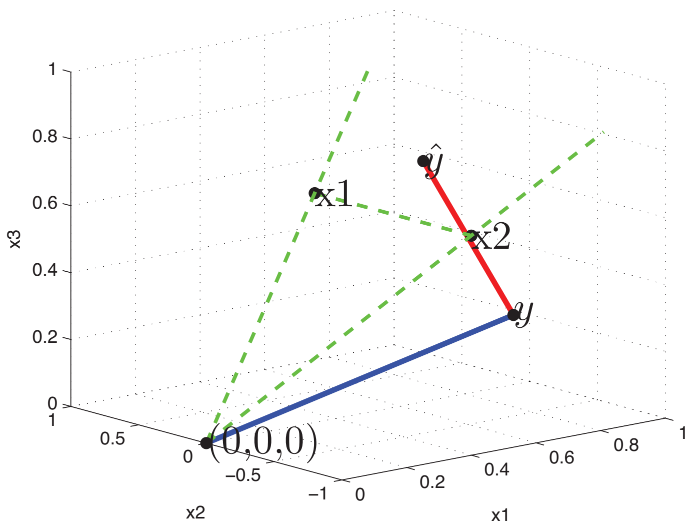

Figure 7.3 Graphical interpretation of least squares for $N = 3$ examples and $D = 2$ features. $\tilde{\mathbf{x}}_1$ and $\tilde{\mathbf{x}}_2$ are vectors in $\mathbb{R}^3$; together they define a 2D plane. $\mathbf{y}$ is also a vector in $\mathbb{R}^3$ but does not lie on this 2D plane. The orthogonal projection of $\mathbf{y}$ onto this plane is denoted $\hat{\mathbf{y}}$. The red line from $\mathbf{y}$ to $\hat{\mathbf{y}}$ is the residual, whose norm we want to minimize. For visual clarity, all vectors have been converted to unit norm. Figure generated by `leastSquaresProjection`.

To minimize the norm of the residual, $\mathbf{y} - \hat{\mathbf{y}}$, we want the residual vector to be orthogonal to every column of $\mathbf{X}$, so $\tilde{\mathbf{x}}_j^T(\mathbf{y} - \hat{\mathbf{y}}) = 0$ for $j = 1 : D$. Hence

$$\tilde{\mathbf{x}}_j^T(\mathbf{y} - \hat{\mathbf{y}}) = 0 \Rightarrow \mathbf{X}^T(\mathbf{y} - \mathbf{X}\mathbf{w}) = \mathbf{0} \Rightarrow \mathbf{w} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{7.20}$$

Hence our projected value of $\mathbf{y}$ is given by

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{w}} = \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{7.21}$$

This corresponds to an **orthogonal projection** of $\mathbf{y}$ onto the column space of $\mathbf{X}$. The projection matrix $\mathbf{P} \triangleq \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ is called the **hat matrix**, since it “puts the hat on $\mathbf{y}$”.

7.3.3 Convexity

When discussing least squares, we noted that the NLL had a bowl shape with a unique minimum. The technical term for functions like this is **convex**. Convex functions play a very important role in machine learning.

Let us define this concept more precisely. We say a *set* $\mathcal{S}$ is **convex** if for any $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathcal{S}$, we have

$$\lambda\boldsymbol{\theta} + (1-\lambda)\boldsymbol{\theta}' \in \mathcal{S}, \quad \forall\, \lambda \in [0,1] \tag{7.22}$$

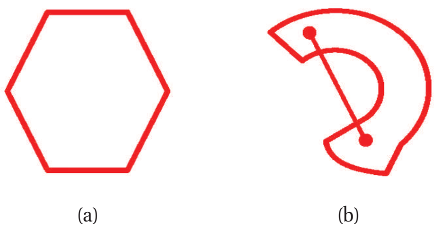

Figure 7.4 (a) Illustration of a convex set. (b) Illustration of a nonconvex set.

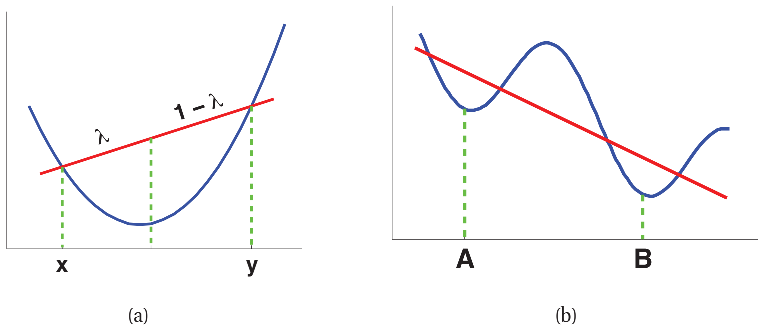

Figure 7.5 (a) Illustration of a convex function. We see that the chord joining $(x, f(x))$ to $(y, f(y))$ lies above the function. (b) A function that is neither convex nor concave. **A** is a local minimum, **B** is a global minimum. Figure generated by `convexFnHand`.

That is, if we draw a line from $\boldsymbol{\theta}$ to $\boldsymbol{\theta}'$, all points on the line lie inside the set. See Figure 7.4(a) for an illustration of a convex set, and Figure 7.4(b) for an illustration of a non-convex set.

A *function* $f(\boldsymbol{\theta})$ is called convex if its **epigraph** (the set of points above the function) defines a convex set. Equivalently, a function $f(\boldsymbol{\theta})$ is called convex if it is defined on a convex set and if, for any $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathcal{S}$, and for any $0 \leq \lambda \leq 1$, we have

$$f(\lambda\boldsymbol{\theta} + (1-\lambda)\boldsymbol{\theta}') \leq \lambda f(\boldsymbol{\theta}) + (1-\lambda) f(\boldsymbol{\theta}') \tag{7.23}$$

See Figure 7.5 for a 1d example. A function is called **strictly convex** if the inequality is strict. A function $f(\boldsymbol{\theta})$ is **concave** if $-f(\boldsymbol{\theta})$ is convex. Examples of scalar convex functions include θ^2, e^θ, and $\theta \log \theta$ (for $\theta > 0$). Examples of scalar concave functions include $\log(\theta)$ and $\sqrt{\theta}$.

Intuitively, a (strictly) convex function has a "bowl shape", and hence has a unique global minimum θ^* corresponding to the bottom of the bowl. Hence its second derivative must be positive everywhere, $\frac{d}{d\theta} f(\theta) > 0$. A twice-continuously differentiable, multivariate function f is convex iff its Hessian is positive definite for all $\boldsymbol{\theta}$.[1] In the machine learning context, the function f often corresponds to the NLL.

1. Recall that the Hessian is the matrix of second partial derivatives, defined by $H_{jk} = \frac{\partial f^2(\theta)}{\partial\theta_j \partial\theta_k}$. Also, recall that a matrix $\mathbf{H}$ is **positive definite** iff $\mathbf{v}^T\mathbf{H}\mathbf{v} > 0$ for any non-zero vector $\mathbf{v}$.

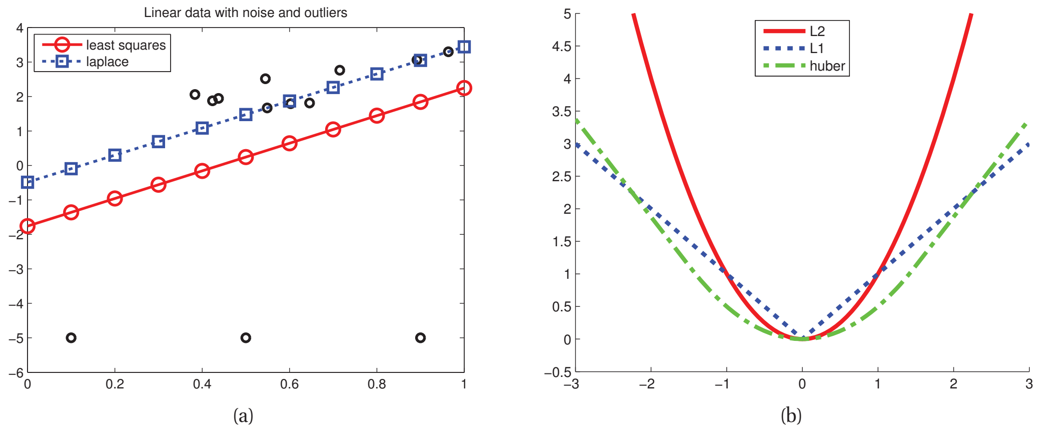

Figure 7.6 (a) Illustration of robust linear regression. Figure generated by `linregRobustDemoCombined`. (b) Illustration of ℓ_2, ℓ_1, and Huber loss functions. Figure generated by `huberLossDemo`.

Models where the NLL is convex are desirable, since this means we can always find the globally optimal MLE. We will see many examples of this later in the book. However, many models of interest will not have concave likelihoods. In such cases, we will discuss ways to derive locally optimal parameter estimates.

7.4 Robust linear regression *

It is very common to model the noise in regression models using a Gaussian distribution with zero mean and constant variance, $\epsilon_i \sim \mathcal{N}(0, \sigma^2)$, where $\epsilon_i = y_i - \mathbf{w}^T\mathbf{x}_i$. In this case, maximizing likelihood is equivalent to minimizing the sum of squared residuals, as we have seen. However, if we have **outliers** in our data, this can result in a poor fit, as illustrated in Figure 7.6(a). (The outliers are the points on the bottom of the figure.) This is because squared error penalizes deviations quadratically, so points far from the line have more affect on the fit than points near to the line.

One way to achieve **robustness** to outliers is to replace the Gaussian distribution for the response variable with a distribution that has **heavy tails**. Such a distribution will assign higher likelihood to outliers, without having to perturb the straight line to "explain" them.

One possibility is to use the Laplace distribution, introduced in Section 2.4.3. If we use this as our observation model for regression, we get the following likelihood:

$$p(y|\mathbf{x}, \mathbf{w}, b) = \text{Lap}(y|\mathbf{w}^T\mathbf{x}, b) \propto \exp(-\frac{1}{b}|y - \mathbf{w}^T\mathbf{x}|) \tag{7.24}$$

The robustness arises from the use of $|y - \mathbf{w}^T\mathbf{x}|$ instead of $(y - \mathbf{w}^T\mathbf{x})^2$. For simplicity, we will assume b is fixed. Let $r_i \triangleq y_i - \mathbf{w}^T\mathbf{x}_i$ be the i'th residual. The NLL has the form

$$\ell(\mathbf{w}) = \sum_i |r_i(\mathbf{w})| \tag{7.25}$$

Likelihood	Prior	Name	Section
Gaussian	Uniform	Least squares	7.3
Gaussian	Gaussian	Ridge	7.5
Gaussian	Laplace	Lasso	13.3
Laplace	Uniform	Robust regression	7.4
Student	Uniform	Robust regression	Exercise 11.12

Table 7.1 Summary of various likelihoods and priors used for linear regression. The likelihood refers to the distributional form of $p(y|\mathbf{x}, \mathbf{w}, \sigma^2)$, and the prior refers to the distributional form of $p(\mathbf{w})$. MAP estimation with a uniform distribution corresponds to MLE.

Unfortunately, this is a non-linear objective function, which is hard to optimize. Fortunately, we can convert the NLL to a linear objective, subject to linear constraints, using the following **split variable** trick. First we define

$$r_i \triangleq r_i^+ - r_i^- \tag{7.26}$$

and then we impose the linear inequality constraints that $r_i^+ \geq 0$ and $r_i^- \geq 0$. Now the constrained objective becomes

$$\min_{\mathbf{w},\mathbf{r}^+,\mathbf{r}^-} \sum_i (r_i^+ - r_i^-) \qquad \text{s.t.} \quad r_i^+ \geq 0, r_i^- \geq 0, \mathbf{w}^T\mathbf{x}_i + r_i^+ + r_i^- = y_i \tag{7.27}$$

This is an example of a **linear program** with $D + 2N$ unknowns and $3N$ constraints.

Since this is a convex optimization problem, it has a unique solution. To solve an LP, we must first write it in standard form, which as follows:

$$\min_{\boldsymbol{\theta}} \mathbf{f}^T\boldsymbol{\theta} \;\; \text{s.t.} \;\; \mathbf{A}\boldsymbol{\theta} \leq \mathbf{b}, \; \mathbf{A}_{eq}\boldsymbol{\theta} = \mathbf{b}_{eq}, \; \mathbf{l} \leq \boldsymbol{\theta} \leq \mathbf{u} \tag{7.28}$$

In our current example, $\boldsymbol{\theta} = (\mathbf{w}, \mathbf{r}^+, \mathbf{r}^-)$, $\mathbf{f} = [\mathbf{0}, \mathbf{1}, \mathbf{1}]$, $\mathbf{A} = []$, $\mathbf{b} = []$, $\mathbf{A}_{eq} = [\mathbf{X}, \mathbf{I}, -\mathbf{I}]$, $\mathbf{b}_{eq} = \mathbf{y}$, $\mathbf{l} = [-\infty\mathbf{1}, \mathbf{0}, \mathbf{0}]$, $\mathbf{u} = []$. This can be solved by any LP solver (see e.g., (Boyd and Vandenberghe 2004)). See Figure 7.6(a) for an example of the method in action.

An alternative to using NLL under a Laplace likelihood is to minimize the **Huber loss** function (Huber 1964), defined as follows:

$$L_H(r, \delta) = \begin{cases} r^2/2 & \text{if } |r| \leq \delta \\ \delta|r| - \delta^2/2 & \text{if } |r| > \delta \end{cases} \tag{7.29}$$

This is equivalent to ℓ_2 for errors that are smaller than δ, and is equivalent to ℓ_1 for larger errors. See Figure 7.6(b). The advantage of this loss function is that it is everywhere differentiable, using the fact that $\frac{d}{dr}|r| = \text{sign}(r)$ if $r \neq 0$. We can also check that the function is C_1 continuous, since the gradients of the two parts of the function match at $r = \pm\delta$, namely $\frac{d}{dr}L_H(r, \delta)|_{r=\delta} = \delta$. Consequently optimizing the Huber loss is much faster than using the Laplace likelihood, since we can use standard smooth optimization methods (such as quasi-Newton) instead of linear programming.

Figure 7.6(a) gives an illustration of the Huber loss function. The results are qualitatively similiar to the probabilistic methods. (In fact, it turns out that the Huber method also has a probabilistic interpretation, although it is rather unnatural (Pontil et al. 1998).)

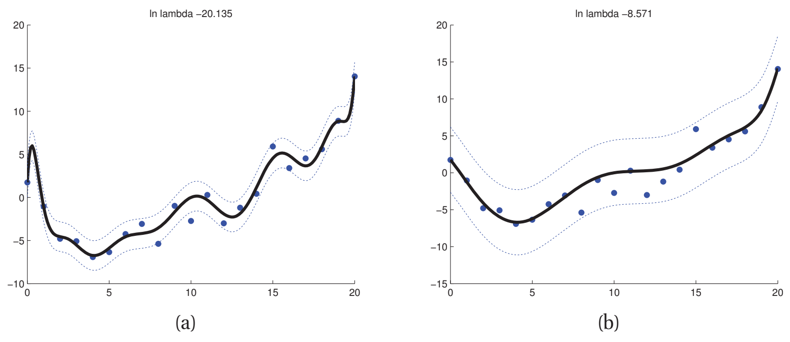

Figure 7.7 Degree 14 Polynomial fit to $N = 21$ data points with increasing amounts of ℓ_2 regularization. Data was generated from noise with variance $\sigma^2 = 4$. The error bars, representing the noise variance σ^2, get wider as the fit gets smoother, since we are ascribing more of the data variation to the noise. Figure generated by `linregPolyVsRegDemo`.

7.5 Ridge regression

One problem with ML estimation is that it can result in overfitting. In this section, we discuss a way to ameliorate this problem by using MAP estimation with a Gaussian prior. For simplicity, we assume a Gaussian likelihood, rather than a robust likelihood.

7.5.1 Basic idea

The reason that the MLE can overfit is that it is picking the parameter values that are the best for modeling the training data; but if the data is noisy, such parameters often result in complex functions. As a simple example, suppose we fit a degree 14 polynomial to $N = 21$ data points using least squares. The resulting curve is very "wiggly", as shown in Figure 7.7(a). The corresponding least squares coefficients (excluding w_0) are as follows:

```
6.560, -36.934, -109.255, 543.452, 1022.561, -3046.224, -3768.013,
8524.540, 6607.897, -12640.058, -5530.188, 9479.730, 1774.639, -2821.526
```

We see that there are many large positive and negative numbers. These balance out exactly to make the curve "wiggle" in just the right way so that it almost perfectly interpolates the data. But this situation is unstable: if we changed the data a little, the coefficients would change a lot.

We can encourage the parameters to be small, thus resulting in a smoother curve, by using a zero-mean Gaussian prior:

$$p(\mathbf{w}) = \prod_j \mathcal{N}(w_j|0, \tau^2) \tag{7.30}$$

where $1/\tau^2$ controls the strength of the prior. The corresponding MAP estimation problem becomes

$$\underset{\mathbf{w}}{\operatorname{argmax}} \sum_{i=1}^{N} \log \mathcal{N}(y_i|w_0 + \mathbf{w}^T \mathbf{x}_i, \sigma^2) + \sum_{j=1}^{D} \log \mathcal{N}(w_j|0, \tau^2) \tag{7.31}$$

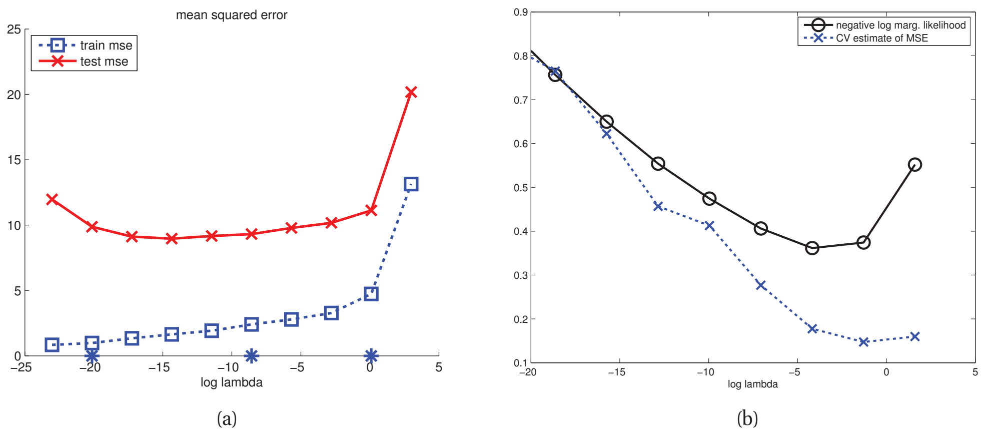

Figure 7.8 (a) Training error (dotted blue) and test error (solid red) for a degree 14 polynomial fit by ridge regression, plotted vs $\log(\lambda)$. Data was generated from noise with variance $\sigma^2 = 4$ (training set has size $N = 21$). Note: Models are ordered from complex (small regularizer) on the left to simple (large regularizer) on the right. The stars correspond to the values used to plot the functions in Figure 7.7. (b) Estimate of performance using training set. Dotted blue: 5-fold cross-validation estimate of future MSE. Solid black: negative log marginal likelihood, $-\log p(\mathcal{D}|\lambda)$. Both curves have been vertically rescaled to [0,1] to make them comparable. Figure generated by `linregPolyVsRegDemo`.

It is a simple exercise to show that this is equivalent to minimizing the following:

$$J(\mathbf{w}) = \frac{1}{N}\sum_{i=1}^{N}(y_i - (w_0 + \mathbf{w}^T\mathbf{x}_i))^2 + \lambda||\mathbf{w}||_2^2 \tag{7.32}$$

where $\lambda \triangleq \sigma^2/\tau^2$ and $||\mathbf{w}||_2^2 = \sum_j w_j^2 = \mathbf{w}^T\mathbf{w}$ is the squared two-norm. Here the first term is the MSE/ NLL as usual, and the second term, $\lambda \geq 0$, is a complexity penalty. The corresponding solution is given by

$$\boxed{\hat{\mathbf{w}}_{ridge} = (\lambda\mathbf{I}_D + \mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}} \tag{7.33}$$

This technique is known as **ridge regression**, or **penalized least squares**. In general, adding a Gaussian prior to the parameters of a model to encourage them to be small is called ℓ_2 **regularization** or **weight decay**. Note that the offset term w_0 is not regularized, since this just affects the height of the function, not its complexity. By penalizing the sum of the magnitudes of the weights, we ensure the function is simple (since $\mathbf{w} = \mathbf{0}$ corresponds to a straight line, which is the simplest possible function, corresponding to a constant.)

We illustrate this idea in Figure 7.7, where we see that increasing λ results in smoother functions. The resulting coefficients also become smaller. For example, using $\lambda = 10^{-3}$, we have

```
2.128, 0.807, 16.457, 3.704, -24.948, -10.472, -2.625, 4.360, 13.711,
10.063, 8.716, 3.966, -9.349, -9.232
```

In Figure 7.8(a), we plot the MSE on the training and test sets vs $\log(\lambda)$. We see that, as we increase λ (so the model becomes more constrained), the error on the training set increases. For the test set, we see the characteristic U-shaped curve, where the model overfits and then underfits. It is common to use cross validation to pick λ, as shown in Figure 7.8(b). In Section 1.4.8, we will discuss a more probabilistic approach.

We will consider a variety of different priors in this book. Each of these corresponds to a different form of **regularization**. This technique is very widely used to prevent overfitting.

7.5.2 Numerically stable computation *

Interestingly, ridge regression, which works better statistically, is also easier to fit numerically, since $(\lambda \mathbf{I}_D + \mathbf{X}^T\mathbf{X})$ is much better conditioned (and hence more likely to be invertible) than $\mathbf{X}^T\mathbf{X}$, at least for suitable largy λ.

Nevertheless, inverting matrices is still best avoided, for reasons of numerical stability. (Indeed, if you write `w=inv(X' * X)*X'*y` in Matlab, it will give you a warning.) We now describe a useful trick for fitting ridge regression models (and hence by extension, computing vanilla OLS estimates) that is more numerically robust. We assume the prior has the form $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \boldsymbol{\Lambda}^{-1})$, where $\boldsymbol{\Lambda}$ is the precision matrix. In the case of ridge regression, $\boldsymbol{\Lambda} = (1/\tau^2)\mathbf{I}$. To avoid penalizing the w_0 term, we should center the data first, as explained in Exercise 7.5.

First let us augment the original data with some "virtual data" coming from the prior:

$$\tilde{\mathbf{X}} = \begin{pmatrix} \mathbf{X}/\sigma \\ \sqrt{\boldsymbol{\Lambda}} \end{pmatrix}, \quad \tilde{\mathbf{y}} = \begin{pmatrix} \mathbf{y}/\sigma \\ \mathbf{0}_{D\times 1} \end{pmatrix} \tag{7.34}$$

where $\boldsymbol{\Lambda} = \sqrt{\boldsymbol{\Lambda}}\sqrt{\boldsymbol{\Lambda}}^T$ is a **Cholesky decomposition** of $\boldsymbol{\Lambda}$. We see that $\tilde{\mathbf{X}}$ is $(N + D) \times D$, where the extra rows represent pseudo-data from the prior.

We now show that the NLL on this expanded data is equivalent to *penalized* NLL on the original data:

$$\begin{aligned}
f(\mathbf{w}) &= (\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\mathbf{w})^T(\tilde{\mathbf{y}} - \tilde{\mathbf{X}}\mathbf{w}) && (7.35)\\
&= \left(\begin{pmatrix} \mathbf{y}/\sigma \\ \mathbf{0} \end{pmatrix} - \begin{pmatrix} \mathbf{X}/\sigma \\ \sqrt{\boldsymbol{\Lambda}} \end{pmatrix}\mathbf{w}\right)^T \left(\begin{pmatrix} \mathbf{y}/\sigma \\ \mathbf{0} \end{pmatrix} - \begin{pmatrix} \mathbf{X}/\sigma \\ \sqrt{\boldsymbol{\Lambda}} \end{pmatrix}\mathbf{w}\right) && (7.36)\\
&= \begin{pmatrix} \frac{1}{\sigma}(\mathbf{y} - \mathbf{X}\mathbf{w}) \\ -\sqrt{\boldsymbol{\Lambda}}\mathbf{w} \end{pmatrix}^T \begin{pmatrix} \frac{1}{\sigma}(\mathbf{y} - \mathbf{X}\mathbf{w}) \\ -\sqrt{\boldsymbol{\Lambda}}\mathbf{w} \end{pmatrix} && (7.37)\\
&= \frac{1}{\sigma^2}(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) + (\sqrt{\boldsymbol{\Lambda}}\mathbf{w})^T(\sqrt{\boldsymbol{\Lambda}}\mathbf{w}) && (7.38)\\
&= \frac{1}{\sigma^2}(\mathbf{y} - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mathbf{X}\mathbf{w}) + \mathbf{w}^T\boldsymbol{\Lambda}\mathbf{w} && (7.39)
\end{aligned}$$

Hence the MAP estimate is given by

$$\hat{\mathbf{w}}_{ridge} = (\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1}\tilde{\mathbf{X}}^T\tilde{\mathbf{y}} \tag{7.40}$$

as we claimed.

Now let

$$\tilde{\mathbf{X}} = \mathbf{Q}\mathbf{R} \tag{7.41}$$

be the **QR decomposition** of $\mathbf{X}$, where $\mathbf{Q}$ is orthonormal (meaning $\mathbf{Q}^T\mathbf{Q} = \mathbf{Q}\mathbf{Q}^T = \mathbf{I}$), and $\mathbf{R}$ is upper triangular. Then

$$(\tilde{\mathbf{X}}^T\tilde{\mathbf{X}})^{-1} = (\mathbf{R}^T\mathbf{Q}^T\mathbf{Q}\mathbf{R})^{-1} = (\mathbf{R}^T\mathbf{R})^{-1} = \mathbf{R}^{-1}\mathbf{R}^{-T} \tag{7.42}$$

Hence

$$\hat{\mathbf{w}}_{ridge} = \mathbf{R}^{-1}\mathbf{R}^{-T}\mathbf{R}^T\mathbf{Q}^T\tilde{\mathbf{y}} = \mathbf{R}^{-1}\mathbf{Q}\tilde{\mathbf{y}} \tag{7.43}$$

Note that $\mathbf{R}$ is easy to invert since it is upper triangular. This gives us a way to compute the ridge estimate while avoiding having to invert $(\mathbf{\Lambda} + \mathbf{X}^T\mathbf{X})$.

We can use this technique to find the MLE, by simply computing the QR decomposition of the unaugmented matrix $\mathbf{X}$, and using the original $\mathbf{y}$. This is the method of choice for solving least squares problems. (In fact, it is so sommon that it can be implemented in one line of Matlab, using the **backslash operator**: `w=X\y`.) Note that computing the QR decomposition of an $N \times D$ matrix takes $O(ND^2)$ time, and is numerically very stable.

If $D \gg N$, we should first perform an SVD decomposition. In particular, let $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ be the SVD of $\mathbf{X}$, where $\mathbf{V}^T\mathbf{V} = \mathbf{I}_N$, $\mathbf{U}\mathbf{U}^T = \mathbf{U}^T\mathbf{U} = \mathbf{I}_N$, and $\mathbf{S}$ is a diagonal $N \times N$ matrix. Now let $\mathbf{Z} = \mathbf{U}\mathbf{D}$ be an $N \times N$ matrix. Then we can rewrite the ridge estimate thus:

$$\hat{\mathbf{w}}_{ridge} = \mathbf{V}(\mathbf{Z}^T\mathbf{Z} + \lambda\mathbf{I}_N)^{-1}\mathbf{Z}^T\mathbf{y} \tag{7.44}$$

In other words, we can replace the D-dimensional vectors $\mathbf{x}_i$ with the N-dimensional vectors $\mathbf{z}_i$ and perform our penalized fit as before. We then transform the N-dimensional solution to the D-dimensional solution by multiplying by $\mathbf{V}$. Geometrically, we are rotating to a new coordinate system in which all but the first N coordinates are zero. This does not affect the solution since the spherical Gaussian prior is rotationally invariant. The overall time is now $O(DN^2)$ operations.

7.5.3 Connection with PCA *

In this section, we discuss an interesting connection between ridge regression and PCA (Section 12.2), which gives further insight into why ridge regression works well. Our discussion is based on (Hastie et al. 2009, p66).

Let $\mathbf{X} = \mathbf{U}\mathbf{S}\mathbf{V}^T$ be the SVD of $\mathbf{X}$. From Equation 7.44, we have

$$\hat{\mathbf{w}}_{ridge} = \mathbf{V}(\mathbf{S}^2 + \lambda\mathbf{I})^{-1}\mathbf{S}\mathbf{U}^T\mathbf{y} \tag{7.45}$$

Hence the ridge predictions on the training set are given by

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{w}}_{ridge} = \mathbf{U}\mathbf{S}\mathbf{V}^T\mathbf{V}(\mathbf{S}^2 + \lambda\mathbf{I})^{-1}\mathbf{S}\mathbf{U}^T\mathbf{y} \tag{7.46}$$

$$= \mathbf{U}\tilde{\mathbf{S}}\mathbf{U}^T\mathbf{y} = \sum_{j=1}^{D} \mathbf{u}_j\tilde{S}_{jj}\mathbf{u}_j^T\mathbf{y} \tag{7.47}$$

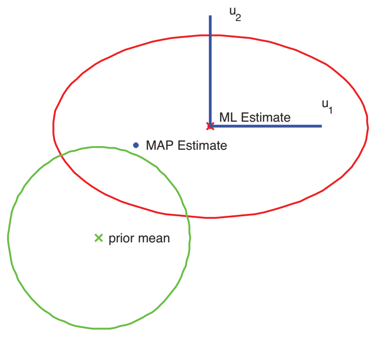

Figure 7.9 Geometry of ridge regression. The likelihood is shown as an ellipse, and the prior is shown as a circle centered on the origin. Based on Figure 3.15 of (Bishop 2006b). Figure generated by `geomRidge`

where

$$\tilde{S}_{jj} \triangleq [\mathbf{S}(\mathbf{S}^2 + \lambda I)^{-1}\mathbf{S}]_{jj} = \frac{\sigma_j^2}{\sigma_j^2 + \lambda} \tag{7.48}$$

and σ_j are the singular values of $\mathbf{X}$. Hence

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{w}}_{ridge} = \sum_{j=1}^{D} \mathbf{u}_j \frac{\sigma_j^2}{\sigma_j^2 + \lambda} \mathbf{u}_j^T \mathbf{y} \tag{7.49}$$

In contrast, the least squares prediction is

$$\hat{\mathbf{y}} = \mathbf{X}\hat{\mathbf{w}}_{ls} = (\mathbf{USV}^T)(\mathbf{VS}^{-1}\mathbf{U}^T\mathbf{y}) = \mathbf{UU}^T\mathbf{y} = \sum_{j=1}^{D} \mathbf{u}_j\mathbf{u}_j^T\mathbf{y} \tag{7.50}$$

If σ_j^2 is small compared to λ, then direction $\mathbf{u}_j$ will not have much effect on the prediction. In view of this, we *define* the effective number of **degrees of freedom** of the model as follows:

$$\mathrm{dof}(\lambda) = \sum_{j=1}^{D} \frac{\sigma_j^2}{\sigma_j^2 + \lambda} \tag{7.51}$$

When $\lambda = 0$, $\mathrm{dof}(\lambda) = D$, and as $\lambda \to \infty$, $\mathrm{dof}(\lambda) \to 0$.

Let us try to understand why this behavior is desirable. In Section 7.6, we show that $\mathrm{cov}[\mathbf{w}|\mathcal{D}] = \sigma^2(\mathbf{X}^T\mathbf{X})^{-1}$, if we use a uniform prior for $\mathbf{w}$. Thus the directions in which we are most uncertain about $\mathbf{w}$ are determined by the eigenvectors of this matrix with the smallest eigenvalues, as shown in Figure 4.1. Furthermore, in Section 12.2.3, we show that the squared singular values σ_j^2 are equal to the eigenvalues of $\mathbf{X}^T\mathbf{X}$. Hence small singular values σ_j correspond to directions with high posterior variance. It is these directions which ridge shrinks the most.

This process is illustrated in Figure 7.9. The horizontal w_1 parameter is not-well determined by the data (has high posterior variance), but the vertical w_2 parameter is well-determined. Hence w_2^{map} is close to $\hat{w}_2^{mle}$, but w_1^{map} is shifted strongly towards the prior mean, which is 0. (Compare to Figure 4.14(c), which illustrated sensor fusion with sensors of different reliabilities.) In this way, ill-determined parameters are reduced in size towards 0. This is called **shrinkage**.

There is a related, but different, technique called **principal components regression**. The idea is this: first use PCA to reduce the dimensionality to K dimensions, and then use these low dimensional features as input to regression. However, this technique does not work as well as ridge in terms of predictive accuracy (Hastie et al. 2001, p70). The reason is that in PC regression, only the first K (derived) dimensions are retained, and the remaining $D - K$ dimensions are entirely ignored. By contrast, ridge regression uses a "soft" weighting of all the dimensions.

7.5.4 Regularization effects of big data

Regularization is the most common way to avoid overfitting. However, another effective approach — which is not always available — is to use lots of data. It should be intuitively obvious that the more training data we have, the better we will be able to learn.[2] So we expect the test set error to decrease to some plateau as N increases.

This is illustrated in Figure 7.10, where we plot the mean squared error incurred on the test set achieved by polynomial regression models of different degrees vs N (a plot of error vs training set size is known as a **learning curve**). The level of the plateau for the test error consists of two terms: an irreducible component that all models incur, due to the intrinsic variability of the generating process (this is called the **noise floor**); and a component that depends on the discrepancy between the generating process (the "truth") and the model: this is called **structural error**.

In Figure 7.10, the truth is a degree 2 polynomial, and we try fitting polynomials of degrees 1, 2 and 25 to this data. Call the 3 models $\mathcal{M}_1$, $\mathcal{M}_2$ and $\mathcal{M}_{25}$. We see that the structural error for models $\mathcal{M}_2$ and $\mathcal{M}_{25}$ is zero, since both are able to capture the true generating process. However, the structural error for $\mathcal{M}_1$ is substantial, which is evident from the fact that the plateau occurs high above the noise floor.

For any model that is expressive enough to capture the truth (i.e., one with small structural error), the test error will go to the noise floor as $N \to \infty$. However, it will typically go to zero faster for simpler models, since there are fewer parameters to estimate. In particular, for finite training sets, there will be some discrepancy between the parameters that we estimate and the best parameters that we could estimate given the particular model class. This is called **approximation error**, and goes to zero as $N \to \infty$, but it goes to zero faster for simpler models. This is illustrated in Figure 7.10. See also Exercise 7.1.

In domains with lots of data, simple methods can work surprisingly well (Halevy et al. 2009). However, there are still reasons to study more sophisticated learning methods, because there will always be problems for which we have little data. For example, even in such a data-rich domain as web search, as soon as we want to start personalizing the results, the amount of data available for any given user starts to look small again (relative to the complexity of the problem).

2. This assumes the training data is randomly sampled, and we don't just get repetitions of the same examples. Having informatively sampled data can help even more; this is the motivation for an approach known as active learning, where you get to choose your training data.

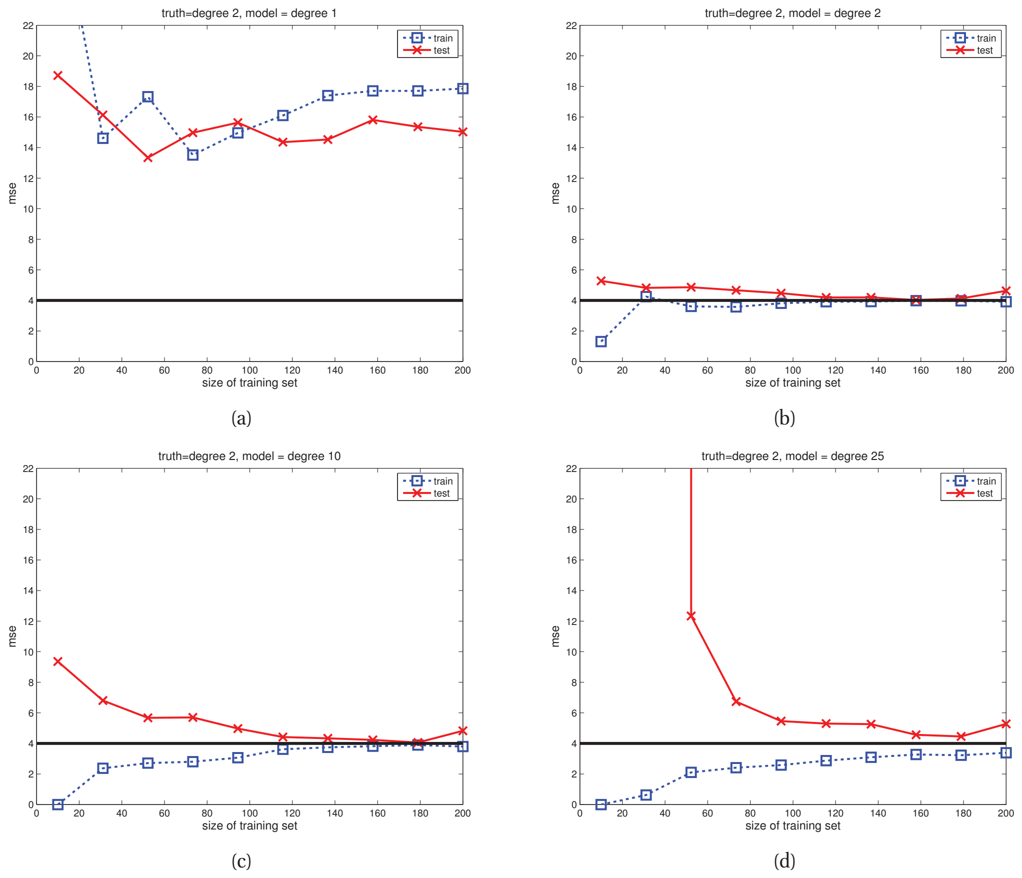

Figure 7.10 MSE on training and test sets vs size of training set, for data generated from a degree 2 polynomial with Gaussian noise of variance $\sigma^2 = 4$. We fit polynomial models of varying degree to this data. (a) Degree 1. (b) Degree 2. (c) Degree 10. (d) Degree 25. Note that for small training set sizes, the test error of the degree 25 polynomial is higher than that of the degree 2 polynomial, due to overfitting, but this difference vanishes once we have enough data. Note also that the degree 1 polynomial is too simple and has high test error even given large amounts of training data. Figure generated by `linregPolyVsN`.

In such cases, we may want to learn multiple related models at the same time, which is known as multi-task learning. This will allow us to "borrow statistical strength" from tasks with lots of data and to share it with tasks with little data. We will discuss ways to do later in the book.

7.6 Bayesian linear regression

Although ridge regression is a useful way to compute a point estimate, sometimes we want to compute the full posterior over $\mathbf{w}$ and σ^2. For simplicity, we will initially assume the noise variance σ^2 is known, so we focus on computing $p(\mathbf{w}|\mathcal{D}, \sigma^2)$. Then in Section 7.6.3 we consider the general case, where we compute $p(\mathbf{w}, \sigma^2|\mathcal{D})$. We assume throughout a Gaussian likelihood model. Performing Bayesian inference with a robust likelihood is also possible, but requires more advanced techniques (see Exercise 24.5).

7.6.1 Computing the posterior

In linear regression, the likelihood is given by

$$\begin{aligned} p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \mu, \sigma^2) &= \mathcal{N}(\mathbf{y}|\mu + \mathbf{X}\mathbf{w}, \sigma^2\mathbf{I}_N) && (7.52) \\ &\propto \exp\left(-\frac{1}{2\sigma^2}(\mathbf{y} - \mu\mathbf{1}_N - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mu\mathbf{1}_N - \mathbf{X}\mathbf{w})\right) && (7.53) \end{aligned}$$

where μ is an offset term. If the inputs are centered, so $\sum_i x_{ij} = 0$ for each j, the mean of the output is equally likely to be positive or negative. So let us put an improper prior on μ of the form $p(\mu) \propto 1$, and then integrate it out to get

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma^2) \propto \exp\left(-\frac{1}{2\sigma^2}||\mathbf{y} - \overline{y}\mathbf{1}_N - \mathbf{X}\mathbf{w}||_2^2\right) \tag{7.54}$$

where $\overline{y} = \frac{1}{N}\sum_{i=1}^N y_i$ is the empirical mean of the output. For notational simplicity, we shall assume the output has been centered, and write $\mathbf{y}$ for $\mathbf{y} - \overline{y}\mathbf{1}_N$.

The conjugate prior to the above Gaussian likelihood is also a Gaussian, which we will denote by $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{w}_0, \mathbf{V}_0)$. Using Bayes rule for Gaussians, Equation 4.125, the posterior is given by

$$\begin{aligned} p(\mathbf{w}|\mathbf{X}, \mathbf{y}, \sigma^2) &\propto \mathcal{N}(\mathbf{w}|\mathbf{w}_0, \mathbf{V}_0)\mathcal{N}(\mathbf{y}|\mathbf{X}\mathbf{w}, \sigma^2\mathbf{I}_N) = \mathcal{N}(\mathbf{w}|\mathbf{w}_N, \mathbf{V}_N) && (7.55) \\ \mathbf{w}_N &= \mathbf{V}_N\mathbf{V}_0^{-1}\mathbf{w}_0 + \frac{1}{\sigma^2}\mathbf{V}_N\mathbf{X}^T\mathbf{y} && (7.56) \\ \mathbf{V}_N^{-1} &= \mathbf{V}_0^{-1} + \frac{1}{\sigma^2}\mathbf{X}^T\mathbf{X} && (7.57) \\ \mathbf{V}_N &= \sigma^2(\sigma^2\mathbf{V}_0^{-1} + \mathbf{X}^T\mathbf{X})^{-1} && (7.58) \end{aligned}$$

If $\mathbf{w}_0 = \mathbf{0}$ and $\mathbf{V}_0 = \tau^2\mathbf{I}$, then the posterior mean reduces to the ridge estimate, if we define $\lambda = \frac{\sigma^2}{\tau^2}$. This is because the mean and mode of a Gaussian are the same.

To gain insight into the posterior distribution (and not just its mode), let us consider a 1D example:

$$y(x, \mathbf{w}) = w_0 + w_1 x + \epsilon \tag{7.59}$$

where the "true" parameters are $w_0 = -0.3$ and $w_1 = 0.5$. In Figure 7.11 we plot the prior, the likelihood, the posterior, and some samples from the posterior predictive. In particular, the right hand column plots the function $y(x, \mathbf{w}^{(s)})$ where x ranges over $[-1, 1]$, and $\mathbf{w}^{(s)} \sim \mathcal{N}(\mathbf{w}|\mathbf{w}_N, \mathbf{V}_N)$ is a sample from the parameter posterior. Initially, when we sample from the prior (first row), our predictions are "all over the place", since our prior is uniform. After we see one data point (second row), our posterior becomes constrained by the corresponding likelihood, and our predictions pass close to the observed data. However, we see that the posterior has a ridge-like shape, reflecting the fact that there are many possible solutions, with different

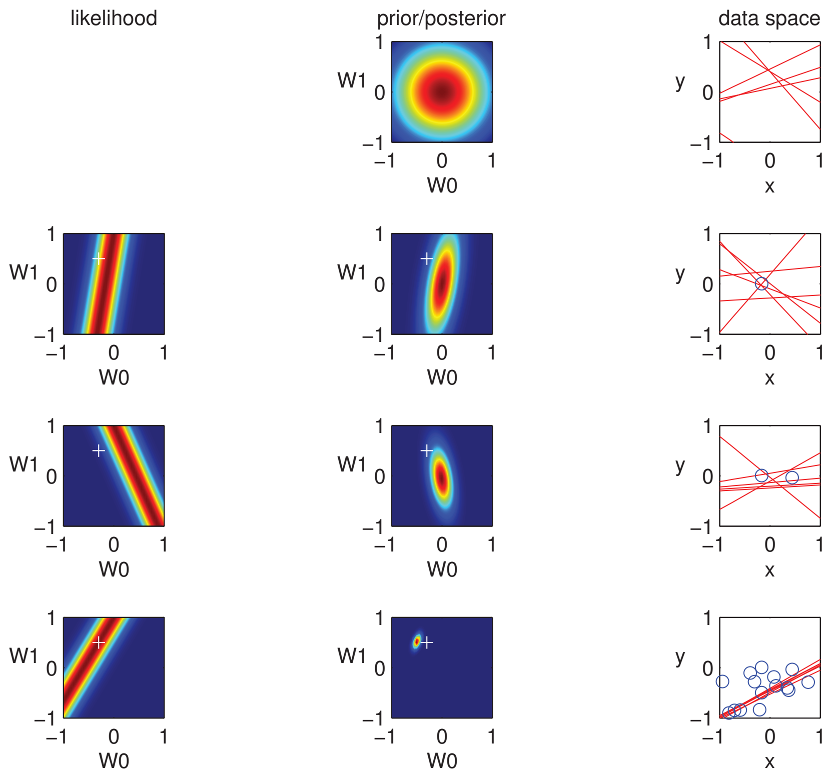

Figure 7.11 Sequential Bayesian updating of a linear regression model $p(y|\mathbf{x}) = \mathcal{N}(y|w_0 x_0 + w_1 x_1, \sigma^2)$. Row 0 represents the prior, row 1 represents the first data point (x_1, y_1), row 2 represents the second data point (x_2, y_2), row 3 represents the 20th data point (x_{20}, y_{20}). Left column: likelihood function for current data point. Middle column: posterior given data so far, $p(\mathbf{w}|\mathbf{x}_{1:n}, y_{1:n})$ (so the first line is the prior). Right column: samples from the current prior/posterior predictive distribution. The white cross in columns 1 and 2 represents the true parameter value; we see that the mode of the posterior rapidly (after 20 samples) converges to this point. The blue circles in column 3 are the observed data points. Based on Figure 3.7 of (Bishop 2006a). Figure generated by `bayesLinRegDemo2d`.

slopes/intercepts. This makes sense since we cannot uniquely infer two parameters from one observation. After we see two data points (third row), the posterior becomes much narrower, and our predictions all have similar slopes and intercepts. After we observe 20 data points (last row), the posterior is essentially a delta function centered on the true value, indicated by a white cross. (The estimate converges to the truth since the data was generated from this model, and because Bayes is a consistent estimator; see Section 6.4.1 for discussion of this point.)

7.6.2 Computing the posterior predictive

It's tough to make predictions, especially about the future. — Yogi Berra

In machine learning, we often care more about predictions than about interpreting the parameters. Using Equation 4.126, we can easily show that the posterior predictive distribution at a test point $\mathbf{x}$ is also Gaussian:

$$\begin{aligned} p(y|\mathbf{x}, \mathcal{D}, \sigma^2) &= \int \mathcal{N}(y|\mathbf{x}^T\mathbf{w}, \sigma^2)\mathcal{N}(\mathbf{w}|\mathbf{w}_N, \mathbf{V}_N)d\mathbf{w} && (7.60) \\ &= \mathcal{N}(y|\mathbf{w}_N^T\mathbf{x}, \sigma_N^2(\mathbf{x})) && (7.61) \\ \sigma_N^2(\mathbf{x}) &= \sigma^2 + \mathbf{x}^T\mathbf{V}_N\mathbf{x} && (7.62) \end{aligned}$$

The variance in this prediction, $\sigma_N^2(\mathbf{x})$, depends on two terms: the variance of the observation noise, σ^2, and the variance in the parameters, $\mathbf{V}_N$. The latter translates into variance about observations in a way which depends on how close $\mathbf{x}$ is to the training data $\mathcal{D}$. This is illustrated in Figure 7.12, where we see that the error bars get larger as we move away from the training points, representing increased uncertainty. This is important for applications such as active learning, where we want to model what we don't know as well as what we do. By contrast, the plugin approximation has constant sized error bars, since

$$p(y|\mathbf{x}, \mathcal{D}, \sigma^2) \approx \int \mathcal{N}(y|\mathbf{x}^T\mathbf{w}, \sigma^2)\delta_{\hat{\mathbf{w}}}(\mathbf{w})d\mathbf{w} = p(y|\mathbf{x}, \hat{\mathbf{w}}, \sigma^2) \quad (7.63)$$

See Figure 7.12(a).

7.6.3 Bayesian inference when σ^2 is unknown *

In this section, we apply the results in Section 4.6.3 to the problem of computing $p(\mathbf{w}, \sigma^2|\mathcal{D})$ for a linear regression model. This generalizes the results from Section 7.6.1 where we assumed σ^2 was known. In the case where we use an uninformative prior, we will see some interesting connections to frequentist statistics.

7.6.3.1 Conjugate prior

As usual, the likelihood has the form

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \sigma^2) = \mathcal{N}(\mathbf{y}|\mathbf{X}\mathbf{w}, \sigma^2\mathbf{I}_N) \quad (7.64)$$

By analogy to Section 4.6.3, one can show that the natural conjugate prior has the following form:

$$\begin{aligned} p(\mathbf{w}, \sigma^2) &= \mathrm{NIG}(\mathbf{w}, \sigma^2|\mathbf{w}_0, \mathbf{V}_0, a_0, b_0) && (7.65) \\ &\triangleq \mathcal{N}(\mathbf{w}|\mathbf{w}_0, \sigma^2\mathbf{V}_0)\mathrm{IG}(\sigma^2|a_0, b_0) && (7.66) \\ &= \frac{b_0^{a_0}}{(2\pi)^{D/2}|\mathbf{V}_0|^{\frac{1}{2}}\Gamma(a_0)}(\sigma^2)^{-(a_0+(D/2)+1)} && (7.67) \\ &\quad \times \exp\left[-\frac{(\mathbf{w}-\mathbf{w}_0)^T\mathbf{V}_0^{-1}(\mathbf{w}-\mathbf{w}_0)+2b_0}{2\sigma^2}\right] && (7.68) \end{aligned}$$

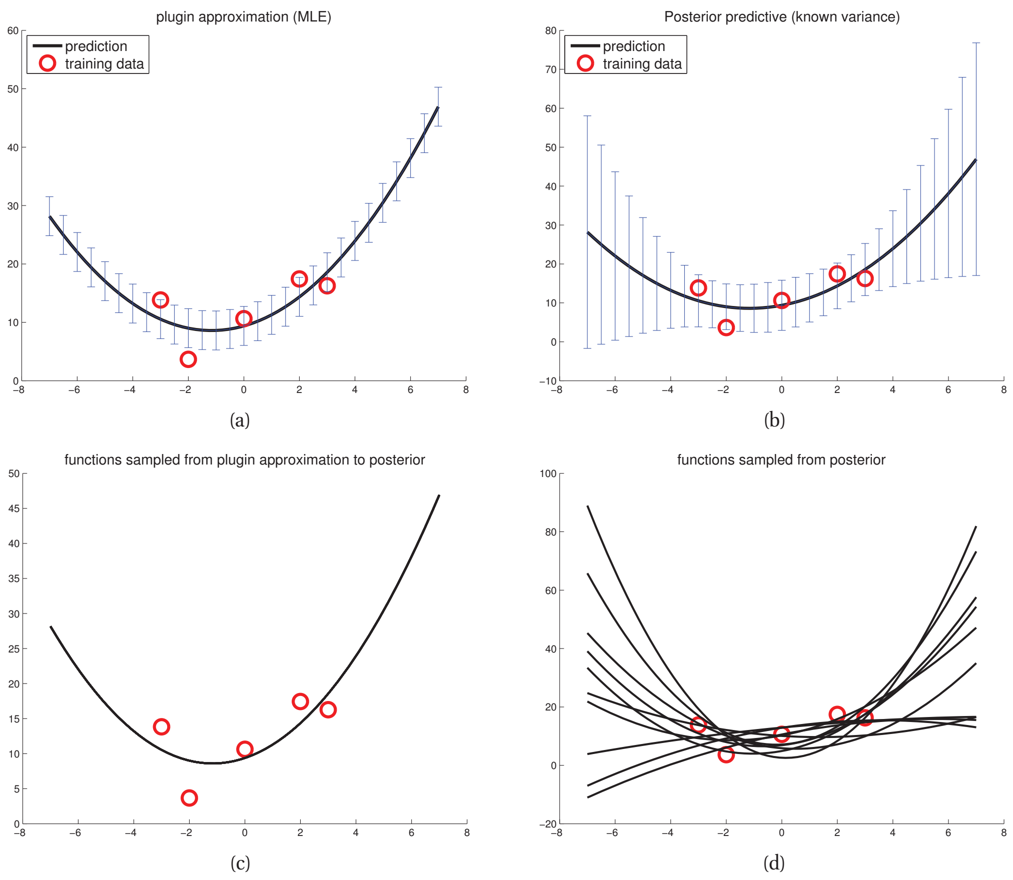

Figure 7.12 (a) Plug-in approximation to predictive density (we plug in the MLE of the parameters). (b) Posterior predictive density, obtained by integrating out the parameters. Black curve is posterior mean, error bars are 2 standard deviations of the posterior predictive density. (c) 10 samples from the plugin approximation to posterior predictive. (d) 10 samples from the posterior predictive. Figure generated by `linregPostPredDemo`.

With this prior and likelihood, one can show that the posterior has the following form:

$$p(\mathbf{w}, \sigma^2|\mathcal{D}) = \mathrm{NIG}(\mathbf{w}, \sigma^2|\mathbf{w}_N, \mathbf{V}_N, a_N, b_N) \tag{7.69}$$

$$\mathbf{w}_N = \mathbf{V}_N(\mathbf{V}_0^{-1}\mathbf{w}_0 + \mathbf{X}^T\mathbf{y}) \tag{7.70}$$

$$\mathbf{V}_N = (\mathbf{V}_0^{-1} + \mathbf{X}^T\mathbf{X})^{-1} \tag{7.71}$$

$$a_N = a_0 + n/2 \tag{7.72}$$

$$b_N = b_0 + \frac{1}{2}\left(\mathbf{w}_0^T\mathbf{V}_0^{-1}\mathbf{w}_0 + \mathbf{y}^T\mathbf{y} - \mathbf{w}_N^T\mathbf{V}_N^{-1}\mathbf{w}_N\right) \tag{7.73}$$

The expressions for $\mathbf{w}_N$ and $\mathbf{V}_N$ are similar to the case where σ^2 is known. The expression for a_N is also intuitive, since it just updates the counts. The expression for b_N can be interpreted as follows: it is the prior sum of squares, b_0, plus the empirical sum of squares, $\mathbf{y}^T\mathbf{y}$, plus a term due to the error in the prior on $\mathbf{w}$.

The posterior marginals are as follows:

$$p(\sigma^2|\mathcal{D}) = \text{IG}(a_N, b_N) \tag{7.74}$$

$$p(\mathbf{w}|\mathcal{D}) = \mathcal{T}(\mathbf{w}_N, \frac{b_N}{a_N}\mathbf{V}_N, 2a_N) \tag{7.75}$$

We give a worked example of using these equations in Section 7.6.3.3.

By analogy to Section 4.6.3.6, the posterior predictive distribution is a Student T distribution. In particular, given m new test inputs $\tilde{\mathbf{X}}$, we have

$$p(\tilde{\mathbf{y}}|\tilde{\mathbf{X}}, \mathcal{D}) = \mathcal{T}(\tilde{\mathbf{y}}|\tilde{\mathbf{X}}\mathbf{w}_N, \frac{b_N}{a_N}(\mathbf{I}_m + \tilde{\mathbf{X}}\mathbf{V}_N\tilde{\mathbf{X}}^T), 2a_N) \tag{7.76}$$

The predictive variance has two components: $(b_N/a_N)\mathbf{I}_m$ due to the measurement noise, and $(b_N/a_N)\tilde{\mathbf{X}}\mathbf{V}_N\tilde{\mathbf{X}}^T$ due to the uncertainty in $\mathbf{w}$. This latter terms varies depending on how close the test inputs are to the training data.

It is common to set $a_0 = b_0 = 0$, corresponding to an uninformative prior for σ^2, and to set $\mathbf{w}_0 = \mathbf{0}$ and $\mathbf{V}_0 = g(\mathbf{X}^T\mathbf{X})^{-1}$ for any positive value g. This is called Zellner's **g-prior** (Zellner 1986). Here g plays a role analogous to $1/\lambda$ in ridge regression. However, the prior covariance is proportional to $(\mathbf{X}^T\mathbf{X})^{-1}$ rather than $\mathbf{I}$. This ensures that the posterior is invariant to scaling of the inputs (Minka 2000b). See also Exercise 7.10.

We will see below that if we use an uninformative prior, the posterior precision given N measurements is $\mathbf{V}_N^{-1} = \mathbf{X}^T\mathbf{X}$. The **unit information prior** is defined to contain as much information as one sample (Kass and Wasserman 1995). To create a unit information prior for linear regression, we need to use $\mathbf{V}_0^{-1} = \frac{1}{N}\mathbf{X}^T\mathbf{X}$, which is equivalent to the g-prior with $g = N$.

7.6.3.2 Uninformative prior

An uninformative prior can be obtained by considering the uninformative limit of the conjugate g-prior, which corresponds to setting $g = \infty$. This is equivalent to an improper NIG prior with $\mathbf{w}_0 = 0$, $\mathbf{V}_0 = \infty\mathbf{I}$, $a_0 = 0$ and $b_0 = 0$, which gives $p(\mathbf{w}, \sigma^2) \propto \sigma^{-(D+2)}$.

Alternatively, we can start with the semi-conjugate prior $p(\mathbf{w}, \sigma^2) = p(\mathbf{w})p(\sigma^2)$, and take each term to its uninformative limit individually, which gives $p(\mathbf{w}, \sigma^2) \propto \sigma^{-2}$. This is equivalent to an improper NIG prior with $\mathbf{w}_0 = \mathbf{0}$,$\mathbf{V} = \infty\mathbf{I}$, $a_0 = -D/2$ and $b_0 = 0$. The corresponding posterior is given by

$$p(\mathbf{w}, \sigma^2|\mathcal{D}) = \text{NIG}(\mathbf{w}, \sigma^2|\mathbf{w}_N, \mathbf{V}_N, a_N, b_N) \tag{7.77}$$

$$\mathbf{w}_N = \hat{\mathbf{w}}_{mle} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y} \tag{7.78}$$

$$\mathbf{V}_N = (\mathbf{X}^T\mathbf{X})^{-1} \tag{7.79}$$

$$a_N = \frac{N-D}{2} \tag{7.80}$$

$$b_N = \frac{s^2}{2} \tag{7.81}$$

$$s^2 \triangleq (\mathbf{y} - \mathbf{X}\hat{\mathbf{w}}_{mle})^T(\mathbf{y} - \mathbf{X}\hat{\mathbf{w}}_{mle} \tag{7.82}$$

w_j	$\mathbb{E}[w_j\|\mathcal{D}]$	$\sqrt{\text{var}[w_j\|\mathcal{D}]}$	95% CI	sig
w0	10.998	3.06027	[4.652, 17.345]	*
w1	-0.004	0.00156	[-0.008, -0.001]	*
w2	-0.054	0.02190	[-0.099, -0.008]	*
w3	0.068	0.09947	[-0.138, 0.274]	
w4	-1.294	0.56381	[-2.463, -0.124]	*
w5	0.232	0.10438	[0.015, 0.448]	*
w6	-0.357	1.56646	[-3.605, 2.892]	
w7	-0.237	1.00601	[-2.324, 1.849]	
w8	0.181	0.23672	[-0.310, 0.672]	
w9	-1.285	0.86485	[-3.079, 0.508]	
w10	-0.433	0.73487	[-1.957, 1.091]	

Table 7.2 Posterior mean, standard deviation and credible intervals for a linear regression model with an uninformative prior fit to the caterpillar data. Produced by `linregBayesCaterpillar`.

The marginal distribution of the weights is given by

$$p(\mathbf{w}|\mathcal{D}) = \mathcal{T}(\mathbf{w}|\hat{\mathbf{w}}, \frac{s^2}{N-D}\mathbf{C}, N-D) \tag{7.83}$$

where $\mathbf{C} = (\mathbf{X}^T\mathbf{X})^{-1}$ and $\hat{\mathbf{w}}$ is the MLE. We discuss the implications of these equations below.

7.6.3.3 An example where Bayesian and frequentist inference coincide *

The use of a (semi-conjugate) uninformative prior is interesting because the resulting posterior turns out to be equivalent to the results from frequentist statistics (see also Section 4.6.3.9). In particular, from Equation 7.83 we have

$$p(w_j|\mathcal{D}) = T(w_j|\hat{w}_j, \frac{C_{jj}s^2}{N-D}, N-D) \tag{7.84}$$

This is equivalent to the sampling distribution of the MLE which is given by the following (see e.g., (Rice 1995, p542), (Casella and Berger 2002, p554)):

$$\frac{w_j - \hat{w}_j}{s_j} \sim t_{N-D} \tag{7.85}$$

where

$$s_j = \sqrt{\frac{s^2 C_{jj}}{N-D}} \tag{7.86}$$

is the standard error of the estimated parameter. (See Section 6.2 for a discussion of sampling distributions.) Consequently, the frequentist confidence interval and the Bayesian marginal credible interval for the parameters are the same in this case.

As a worked example of this, consider the caterpillar dataset from (Marin and Robert 2007). (The details of what the data mean don't matter for our present purposes.) We can compute the posterior mean and standard deviation, and the 95% credible intervals (CI) for the regression coefficients using Equation 7.84. The results are shown in Table 7.2. It is easy to check that these 95% credible intervals are identical to the 95% confidence intervals computed using standard frequentist methods (see `linregBayesCaterpillar` for the code).

We can also use these marginal posteriors to compute if the coefficients are "significantly" different from 0. An informal way to do this (without using decision theory) is to check if its 95% CI excludes 0. From Table 7.2, we see that the CIs for coefficients 0, 1, 2, 4, 5 are all significant by this measure, so we put a little star by them. It is easy to check that these results are the same as those produced by standard frequentist software packages which compute p-values at the 5% level.

Although the correspondence between the Bayesian and frequentist results might seem appealing to some readers, recall from Section 6.6 that frequentist inference is riddled with pathologies. Also, note that the MLE does not even exist when $N < D$, so standard frequentist inference theory breaks down in this setting. Bayesian inference theory still works, although it requires the use of proper priors. (See (Maruyama and George 2008) for one extension of the g-prior to the case where $D > N$.)

7.6.4 EB for linear regression (evidence procedure)

So far, we have assumed the prior is known. In this section, we describe an empirical Bayes procedure for picking the hyper-parameters. More precisely, we choose $\boldsymbol{\eta} = (\alpha, \lambda)$ to maximize the marignal likelihood, where $\lambda = 1/\sigma^2$ be the precision of the observation noise and $\boldsymbol{\alpha}$ is the precision of the prior, $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}\mathbf{I})$. This is known as the **evidence procedure** (MacKay 1995b).[3] See Section 13.7.4 for the algorithmic details.

The evidence procedure provides an alternative to using cross validation. For example, in Figure 7.13(b), we plot the log marginal likelihood for different values of α, as well as the maximum value found by the optimizer. We see that, in this example, we get the same result as 5-CV, shown in Figure 7.13(a). (We kept $\lambda = 1/\sigma^2$ fixed in both methods, to make them comparable.)

The principle practical advantage of the evidence procedure over CV will become apparent in Section 13.7, where we generalize the prior by allowing a different α_j for every feature. This can be used to perform feature selection, using a technique known as automatic relevancy determination or ARD. By contrast, it would not be possible to use CV to tune D different hyper-parameters.

The evidence procedure is also useful when comparing different kinds of models, since it provides a good approximation to the evidence:

$$p(\mathcal{D}|m) = \int\int p(\mathcal{D}|\mathbf{w}, m)p(\mathbf{w}|m, \boldsymbol{\eta})p(\boldsymbol{\eta}|m)d\mathbf{w}d\boldsymbol{\eta} \tag{7.87}$$

$$\approx \max_{\boldsymbol{\eta}} \int p(\mathcal{D}|\mathbf{w}, m)p(\mathbf{w}|m, \boldsymbol{\eta})p(\boldsymbol{\eta}|m)d\mathbf{w} \tag{7.88}$$

It is important to (at least approximately) integrate over $\boldsymbol{\eta}$ rather than setting it arbitrarily, for reasons discussed in Section 5.3.2.5. Indeed, this is the method we used to evaluate the marginal

3. Alternatively, we could integrate out λ analytically, as shown in Section 7.6.3, and just optimize α (Buntine and Weigend 1991). However, it turns out that this is less accurate than optimizing both α and λ (MacKay 1999).

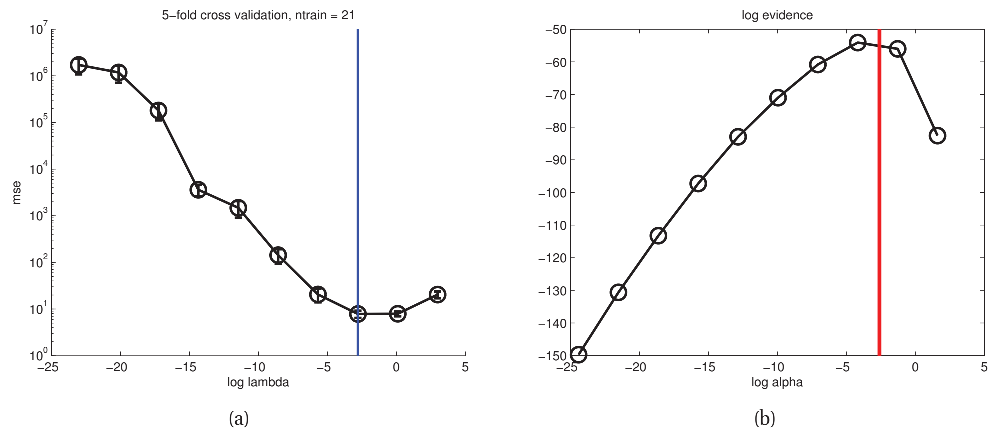

Figure 7.13 (a) Estimate of test MSE produced by 5-fold cross-validation vs $\log(\lambda)$. The smallest value is indicated by the vertical line. Note the vertical scale is in log units. (c) Log marginal likelihood vs $\log(\alpha)$. The largest value is indicated by the vertical line. Figure generated by `linregPolyVsRegDemo`.

likelihood for the polynomial regression models in Figures 5.7 and 5.8. For a "more Bayesian" approach, in which we model our uncertainty about $\boldsymbol{\eta}$ rather than computing point estimates, see Section 21.5.2.

Exercises

Exercise 7.1 Behavior of training set error with increasing sample size

The error on the test will always decrease as we get more training data, since the model will be better estimated. However, as shown in Figure 7.10, for sufficiently complex models, the error on the training set can increase we we get more training data, until we reach some plateau. Explain why.

Exercise 7.2 Multi-output linear regression

(Source: Jaakkola.)

When we have multiple independent outputs in linear regression, the model becomes

$$p(\mathbf{y}|\mathbf{x}, \mathbf{W}) = \prod_{j=1}^{M} \mathcal{N}(y_j|\mathbf{w}_j^T \mathbf{x}_i, \sigma_j^2) \tag{7.89}$$

Since the likelihood factorizes across dimensions, so does the MLE. Thus

$$\hat{\mathbf{W}} = [\hat{\mathbf{w}}_1, \ldots, \hat{\mathbf{w}}_M] \tag{7.90}$$

where $\hat{\mathbf{w}}_j = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{Y}_{:,j}$.

In this exercise we apply this result to a model with 2 dimensional response vector $\mathbf{y}_i \in \mathbb{R}^2$. Suppose we have some binary input data, $x_i \in \{0, 1\}$. The training data is as follows:

x	y
0	$(-1,-1)^T$
0	$(-1,-2)^T$
0	$(-2,-1)^T$
1	$(1,1)^T$
1	$(1,2)^T$
1	$(2,1)^T$

Let us embed each x_i into 2d using the following basis function:

$$\boldsymbol{\phi}(0) = (1,0)^T, \quad \boldsymbol{\phi}(1) = (0,1)^T \tag{7.91}$$

The model becomes

$$\hat{\mathbf{y}} = \mathbf{W}^T \boldsymbol{\phi}(x) \tag{7.92}$$

where $\mathbf{W}$ is a 2×2 matrix. Compute the MLE for $\mathbf{W}$ from the above data.

Exercise 7.3 Centering and ridge regression

Assume that $\overline{\mathbf{x}} = 0$, so the input data has been centered. Show that the optimizer of

$$J(\mathbf{w}, w_0) \quad = \quad (\mathbf{y} - \mathbf{X}\mathbf{w} - w_0\mathbf{1})^T(\mathbf{y} - \mathbf{X}\mathbf{w} - w_0\mathbf{1}) + \lambda\mathbf{w}^T\mathbf{w} \tag{7.93}$$

is

$$\hat{w}_0 \quad = \quad \overline{y} \tag{7.94}$$

$$\mathbf{w} \quad = \quad (\mathbf{X}^T\mathbf{X} + \lambda\mathbf{I})^{-1}\mathbf{X}^T\mathbf{y} \tag{7.95}$$

Exercise 7.4 MLE for σ^2 for linear regression

Show that the MLE for the error variance in linear regression is given by

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}(y_i - \mathbf{x}_i^T\hat{\mathbf{w}})^2 \tag{7.96}$$

This is just the empirical variance of the residual errors when we plug in our estimate of $\hat{\mathbf{w}}$.

Exercise 7.5 MLE for the offset term in linear regression

Linear regression has the form $\mathbb{E}[y|\mathbf{x}] = w_0 + \mathbf{w}^T\mathbf{x}$. It is common to include a column of 1's in the design matrix, so we can solve for the offset term w_0 term and the other parameters $\mathbf{w}$ at the same time using the normal equations. However, it is also possible to solve for $\mathbf{w}$ and w_0 separately. Show that

$$\hat{w}_0 \quad = \quad \frac{1}{N}\sum_i y_i - \frac{1}{N}\sum_i \mathbf{x}_i^T\mathbf{w} = \overline{y} - \overline{\mathbf{x}}^T\mathbf{w} \tag{7.97}$$

So $\hat{w}_0$ models the difference in the average output from the average predicted output. Also, show that

$$\hat{\mathbf{w}} = (\mathbf{X}_c^T\mathbf{X}_c)^{-1}\mathbf{X}_c^T\mathbf{y}_c = \left[\sum_{i=1}^{N}(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^T\right]^{-1}\left[\sum_{i=1}^{N}(y_i - \overline{y})(\mathbf{x}_i - \overline{\mathbf{x}})\right] \tag{7.98}$$

where $\mathbf{X}_c$ is the centered input matrix containing $\mathbf{x}_i^c = \mathbf{x}_i - \overline{\mathbf{x}}$ along its rows, and $\mathbf{y}_c = \mathbf{y} - \overline{\mathbf{y}}$ is the centered output vector. Thus we can first compute $\hat{\mathbf{w}}$ on centered data, and then estimate w_0 using $\overline{y} - \overline{\mathbf{x}}^T\hat{\mathbf{w}}$.

Exercise 7.6 MLE for simple linear regression

Simple linear regression refers to the case where the input is scalar, so $D = 1$. Show that the MLE in this case is given by the following equations, which may be familiar from basic statistics classes:

$$w_1 = \frac{\sum_i (x_i - \overline{x})(y_i - \bar{y})}{\sum_i (x_i - \bar{x})^2} = \frac{\sum_i x_i y_i - N\overline{x}\,\overline{y}}{\sum_i x_i^2 - N\overline{x}^2} \approx \frac{\text{cov}[X,Y]}{\text{var}[X]} \tag{7.99}$$

$$w_0 = \bar{y} - w_1 \bar{x} \approx \mathbb{E}[Y] - w_1 \mathbb{E}[X] \tag{7.100}$$

See `linregDemo1` for a demo.

Exercise 7.7 Sufficient statistics for online linear regression

(Source: Jaakkola.) Consider fitting the model $\hat{y} = w_0 + w_1 x$ using least squares. Unfortunately we did not keep the original data, x_i, y_i, but we do have the following functions (statistics) of the data:

$$\overline{x}^{(n)} = \frac{1}{n}\sum_{i=1}^{n} x_i, \quad \overline{y}^{(n)} = \frac{1}{n}\sum_{i=1}^{n} y_i \tag{7.101}$$

$$C_{xx}^{(n)} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})^2, \quad C_{xy}^{(n)} = \frac{1}{n}\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y}), \quad C_{yy}^{(n)} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \overline{y})^2 \tag{7.102}$$

a. What are the minimal set of statistics that we need to estimate w_1? (Hint: see Equation 7.99.)
b. What are the minimal set of statistics that we need to estimate w_0? (Hint: see Equation 7.97.)
c. Suppose a new data point, x_{n+1}, y_{n+1} arrives, and we want to update our sufficient statistics without looking at the old data, which we have not stored. (This is useful for online learning.) Show that we can this for $\overline{x}$ as follows.

$$\overline{x}^{(n+1)} \triangleq \frac{1}{n+1}\sum_{i=1}^{n+1} x_i = \frac{1}{n+1}\left(n\overline{x}^{(n)} + x_{n+1}\right) \tag{7.103}$$

$$= \overline{x}^{(n)} + \frac{1}{n+1}(x_{n+1} - \overline{x}^{(n)}) \tag{7.104}$$

   This has the form: new estimate is old estimate plus correction. We see that the size of the correction diminishes over time (i.e., as we get more samples). Derive a similar expression to update $\overline{y}$

d. Show that one can update $C_{xy}^{(n+1)}$ recursively using

$$C_{xy}^{(n+1)} = \frac{1}{n+1}\left[x_{n+1}y_{n+1} + nC_{xy}^{(n)} + n\overline{x}^{(n)}\overline{y}^{(n)} - (n+1)\overline{x}^{(n+1)}\overline{y}^{(n+1)}\right] \tag{7.105}$$

   Derive a similar expression to update C_{xx}.

e. Implement the online learning algorithm, i.e., write a function of the form `[w,ss] = linregUpdateSS(ss, x, y)`, where x and y are scalars and ss is a structure containing the sufficient statistics.
f. Plot the coefficients over "time", using the dataset in `linregDemo1`. (Specifically, use `[x,y] = polyDataMake('sampling','thibaux')`.) Check that they converge to the solution given by the batch (offline) learner (i.e, ordinary least squares). Your result should look like Figure 7.14.

Turn in your derivation, code and plot.

Exercise 7.8 Bayesian linear regression in 1d with known σ^2

(Source: Bolstad.) Consider fitting a model of the form

$$p(y|x, \boldsymbol{\theta}) = \mathcal{N}(y|w_0 + w_1 x, \sigma^2) \tag{7.106}$$

to the data shown below:

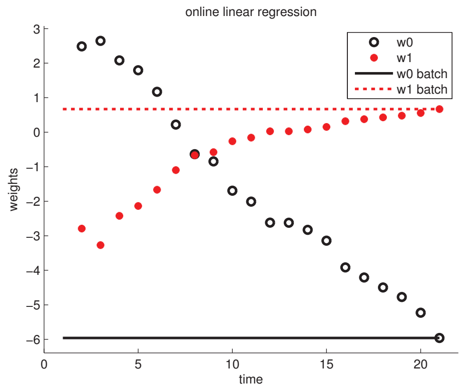

Figure 7.14 Regression coefficients over time. Produced by Exercise 7.7.

```
x = [94,96,94,95,104,106,108,113,115,121,131];
y = [0.47, 0.75, 0.83, 0.98, 1.18, 1.29, 1.40, 1.60, 1.75, 1.90, 2.23];
```

a. Compute an unbiased estimate of σ^2 using

$$\hat{\sigma}^2 = \frac{1}{N-2}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2 \tag{7.107}$$

(The denominator is $N-2$ since we have 2 inputs, namely the offset term and x.) Here $\hat{y}_i = \hat{w}_0 + \hat{w}_1 x_i$, and $\hat{\mathbf{w}} = (\hat{w}_0, \hat{w}_1)$ is the MLE.

b. Now assume the following prior on $\mathbf{w}$:

$$p(\mathbf{w}) = p(w_0)p(w_1) \tag{7.108}$$

Use an (improper) uniform prior on w_0 and a $\mathcal{N}(0,1)$ prior on w_1. Show that this can be written as a Gaussian prior of the form $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{w}_0, \mathbf{V}_0)$. What are $\mathbf{w}_0$ and $\mathbf{V}_0$?

c. Compute the marginal posterior of the slope, $p(w_1|\mathcal{D}, \sigma^2)$, where $\mathcal{D}$ is the data above, and σ^2 is the unbiased estimate computed above. What is $\mathbb{E}\left[w_1|\mathcal{D}, \sigma^2\right]$ and $\text{var}\left[w_1|\mathcal{D}, \sigma^2\right]$ Show your work. (You can use Matlab if you like.) Hint: the posterior variance is a very small number!

d. What is a 95% credible interval for w_1?

Exercise 7.9 Generative model for linear regression

Linear regression is the problem of estimating $E[Y|\mathbf{x}]$ using a linear function of the form $w_0 + \mathbf{w}^T\mathbf{x}$. Typically we assume that the conditional distribution of Y given $\mathbf{X}$ is Gaussian. We can either estimate this conditional Gaussian directly (a discriminative approach), or we can fit a Gaussian to the joint distribution of $\mathbf{X}, Y$ and then derive $E[Y|\mathbf{X} = \mathbf{x}]$.

In Exercise 7.5 we showed that the discriminative approach leads to these equations

$$E[Y|\mathbf{x}] = w_0 + \mathbf{w}^T\mathbf{x} \tag{7.109}$$

$$w_0 = \overline{y} - \overline{\mathbf{x}}^T\mathbf{w} \tag{7.110}$$

$$\mathbf{w} = (\mathbf{X}_c^T\mathbf{X}_c)^{-1}\mathbf{X}_c^T\mathbf{y}_c \tag{7.111}$$

where $\mathbf{X}_c = \mathbf{X} - \bar{\mathbf{X}}$ is the centered input matrix, and $\bar{\mathbf{X}} = \mathbf{1}_n \overline{\mathbf{x}}^T$ replicates $\overline{\mathbf{x}}$ across the rows. Similarly, $\mathbf{y}_c = \mathbf{y} - \overline{\mathbf{y}}$ is the centered output vector, and $\overline{\mathbf{y}} = \mathbf{1}_n \overline{y}$ replicates $\overline{y}$ across the rows.

a. By finding the maximum likelihood estimates of $\mathbf{\Sigma}_{XX}$, $\mathbf{\Sigma}_{XY}$, $\boldsymbol{\mu}_X$ and $\boldsymbol{\mu}_Y$, derive the above equations by fitting a joint Gaussian to $\mathbf{X}, Y$ and using the formula for conditioning a Gaussian (see Section 4.3.1). Show your work.

b. What are the advantages and disadvantages of this approach compared to the standard discriminative approach?

Exercise 7.10 Bayesian linear regression using the g-prior

Show that when we use the g-prior, $p(\mathbf{w}, \sigma^2) = \text{NIG}(\mathbf{w}, \sigma^2 | \mathbf{0}, g(\mathbf{X}^T\mathbf{X})^{-1}, 0, 0)$, the posterior has the following form:

$$p(\mathbf{w}, \sigma^2|\mathcal{D}) = \text{NIG}(\mathbf{w}, \sigma^2|\mathbf{w}_N, \mathbf{V}_N, a_N, b_N) \tag{7.112}$$

$$\mathbf{V}_N = \frac{g}{g+1}(\mathbf{X}^T\mathbf{X})^{-1} \tag{7.113}$$

$$\mathbf{w}_N = \frac{g}{g+1}\hat{\mathbf{w}}_{mle} \tag{7.114}$$

$$a_N = N/2 \tag{7.115}$$

$$b_N = \frac{s^2}{2} + \frac{1}{2(g+1)}\hat{\mathbf{w}}_{mle}^T\mathbf{X}^T\mathbf{X}\hat{\mathbf{w}}_{mle} \tag{7.116}$$

(7.117)