Hence we have successfully factorized the joint as

$$p(\mathbf{x}_1, \mathbf{x}_2) = p(\mathbf{x}_1|\mathbf{x}_2)p(\mathbf{x}_2) \tag{4.118}$$
$$= \mathcal{N}(\mathbf{x}_1|\boldsymbol{\mu}_{1|2}, \boldsymbol{\Sigma}_{1|2})\mathcal{N}(\mathbf{x}_2|\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_{22}) \tag{4.119}$$

where the parameters of the conditional distribution can be read off from the above equations using

$$\boldsymbol{\mu}_{1|2} = \boldsymbol{\mu}_1 + \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}(\mathbf{x}_2 - \boldsymbol{\mu}_2) \tag{4.120}$$
$$\boldsymbol{\Sigma}_{1|2} = \boldsymbol{\Sigma}/\boldsymbol{\Sigma}_{22} = \boldsymbol{\Sigma}_{11} - \boldsymbol{\Sigma}_{12}\boldsymbol{\Sigma}_{22}^{-1}\boldsymbol{\Sigma}_{21} \tag{4.121}$$

We can also use the fact that $|\mathbf{M}| = |\mathbf{M}/\mathbf{H}||\mathbf{H}|$ to check the normalization constants are correct:

$$(2\pi)^{(d_1+d_2)/2}|\boldsymbol{\Sigma}|^{\frac{1}{2}} = (2\pi)^{(d_1+d_2)/2}(|\boldsymbol{\Sigma}/\boldsymbol{\Sigma}_{22}|\ |\boldsymbol{\Sigma}_{22}|)^{\frac{1}{2}} \tag{4.122}$$
$$= (2\pi)^{d_1/2}|\boldsymbol{\Sigma}/\boldsymbol{\Sigma}_{22}|^{\frac{1}{2}}\ (2\pi)^{d_2/2}|\boldsymbol{\Sigma}_{22}|^{\frac{1}{2}} \tag{4.123}$$

where $d_1 = \dim(\mathbf{x}_1)$ and $d_2 = \dim(\mathbf{x}_2)$.

We leave the proof of the other forms of the result in Equation 4.69 as an exercise.

4.4 Linear Gaussian systems

Suppose we have two variables, $\mathbf{x}$ and $\mathbf{y}$. Let $\mathbf{x} \in \mathbb{R}^{D_x}$ be a hidden variable, and $\mathbf{y} \in \mathbb{R}^{D_y}$ be a noisy observation of $\mathbf{x}$. Let us assume we have the following prior and likelihood:

$$\begin{aligned} p(\mathbf{x}) &= \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_x, \boldsymbol{\Sigma}_x) \\ p(\mathbf{y}|\mathbf{x}) &= \mathcal{N}(\mathbf{y}|\mathbf{A}\mathbf{x} + \mathbf{b}, \boldsymbol{\Sigma}_y) \end{aligned} \tag{4.124}$$

where $\mathbf{A}$ is a matrix of size $D_y \times D_x$. This is an example of a **linear Gaussian system**. We can represent this schematically as $\mathbf{x} \rightarrow \mathbf{y}$, meaning $\mathbf{x}$ generates $\mathbf{y}$. In this section, we show how to "invert the arrow", that is, how to infer $\mathbf{x}$ from $\mathbf{y}$. We state the result below, then give several examples, and finally we derive the result. We will see many more applications of these results in later chapters.

4.4.1 Statement of the result

Theorem 4.4.1 (Bayes rule for linear Gaussian systems)**.** *Given a linear Gaussian system, as in Equation 4.124, the posterior* $p(\mathbf{x}|\mathbf{y})$ *is given by the following:*

$$\begin{aligned} p(\mathbf{x}|\mathbf{y}) &= \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{x|y}, \boldsymbol{\Sigma}_{x|y}) \\ \boldsymbol{\Sigma}_{x|y}^{-1} &= \boldsymbol{\Sigma}_x^{-1} + \mathbf{A}^T\boldsymbol{\Sigma}_y^{-1}\mathbf{A} \\ \boldsymbol{\mu}_{x|y} &= \boldsymbol{\Sigma}_{x|y}[\mathbf{A}^T\boldsymbol{\Sigma}_y^{-1}\ (\mathbf{y} - \mathbf{b}) + \boldsymbol{\Sigma}_x^{-1}\boldsymbol{\mu}_x] \end{aligned} \tag{4.125}$$

In addition, the normalization constant $p(\mathbf{y})$ is given by

$$\boxed{p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{A}\boldsymbol{\mu}_x + \mathbf{b}, \boldsymbol{\Sigma}_y + \mathbf{A}\boldsymbol{\Sigma}_x\mathbf{A}^T)} \tag{4.126}$$

For the proof, see Section 4.4.3.

4.4.2 Examples

In this section, we give some example applications of the above result.

4.4.2.1 Inferring an unknown scalar from noisy measurements

Suppose we make N noisy measurements y_i of some underlying quantity x; let us assume the measurement noise has fixed precision $\lambda_y = 1/\sigma^2$, so the likelihood is

$$p(y_i|x) = \mathcal{N}(y_i|x, \lambda_y^{-1}) \tag{4.127}$$

Now let us use a Gaussian prior for the value of the unknown source:

$$p(x) = \mathcal{N}(x|\mu_0, \lambda_0^{-1}) \tag{4.128}$$

We want to compute $p(x|y_1, \ldots, y_N, \sigma^2)$. We can convert this to a form that lets us apply Bayes rule for Gaussians by defining $\mathbf{y} = (y_1, \ldots, y_N)$, $\mathbf{A} = \mathbf{1}_N^T$ (an $1 \times N$ row vector of 1's), and $\boldsymbol{\Sigma}_y^{-1} = \mathrm{diag}(\lambda_y \mathbf{I})$. Then we get

$$p(x|\mathbf{y}) = \mathcal{N}(x|\mu_N, \lambda_N^{-1}) \tag{4.129}$$

$$\lambda_N = \lambda_0 + N\lambda_y \tag{4.130}$$

$$\mu_N = \frac{N\lambda_y \overline{y} + \lambda_0 \mu_0}{\lambda_N} = \frac{N\lambda_y}{N\lambda_y + \lambda_0}\overline{y} + \frac{\lambda_0}{N\lambda_y + \lambda_0}\mu_0 \tag{4.131}$$

These equations are quite intuitive: the posterior precision λ_N is the prior precision λ_0 plus N units of measurement precision λ_y. Also, the posterior mean μ_N is a convex combination of the MLE $\overline{y}$ and the prior mean μ_0. This makes it clear that the posterior mean is a compromise between the MLE and the prior. If the prior is weak relative to the signal strength (λ_0 is small relative to λ_y), we put more weight on the MLE. If the prior is strong relative to the signal strength (λ_0 is large relative to λ_y), we put more weight on the prior. This is illustrated in Figure 4.12, which is very similar to the analogous results for the beta-binomial model in Figure 3.6.

Note that the posterior mean is written in terms of $N\lambda_y\overline{y}$, so having N measurements each of precision λ_y is like having one measurement with value $\overline{y}$ and precision $N\lambda_y$.

We can rewrite the results in terms of the posterior variance, rather than posterior precision,

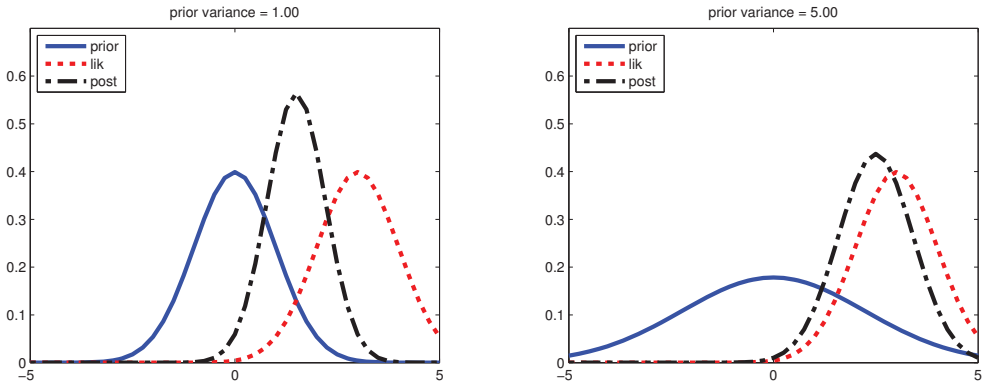


Figure 4.12 Inference about x given a noisy observation $y = 3$. (a) Strong prior $\mathcal{N}(0, 1)$. The posterior mean is "shrunk" towards the prior mean, which is 0. (a) Weak prior $\mathcal{N}(0, 5)$. The posterior mean is similar to the MLE. Figure generated by `gaussInferParamsMean1d`.

as follows:

$$p(x|\mathcal{D}, \sigma^2) = \mathcal{N}(x|\mu_N, \tau_N^2) \tag{4.132}$$

$$\tau_N^2 = \frac{1}{\frac{N}{\sigma^2} + \frac{1}{\tau_0^2}} = \frac{\sigma^2\tau_0^2}{N\tau_0^2 + \sigma^2} \tag{4.133}$$

$$\mu_N = \tau_N^2\left(\frac{\mu_0}{\tau_0^2} + \frac{N\overline{y}}{\sigma^2}\right) = \frac{\sigma^2}{N\tau_0^2 + \sigma^2}\mu_0 + \frac{N\tau_0^2}{N\tau_0^2 + \sigma^2}\overline{y} \tag{4.134}$$

where $\tau_0^2 = 1/\lambda_0$ is the prior variance and $\tau_N^2 = 1/\lambda_N$ is the posterior variance.

We can also compute the posterior sequentially, by updating after each observation. If $N = 1$, we can rewrite the posterior after seeing a single observation as follows (where we define $\Sigma_y = \sigma^2$, $\Sigma_0 = \tau_0^2$ and $\Sigma_1 = \tau_1^2$ to be the variances of the likelihood, prior and posterior):

$$p(x|y) = \mathcal{N}(x|\mu_1, \Sigma_1) \tag{4.135}$$

$$\Sigma_1 = \left(\frac{1}{\Sigma_0} + \frac{1}{\Sigma_y}\right)^{-1} = \frac{\Sigma_y\Sigma_0}{\Sigma_0 + \Sigma_y} \tag{4.136}$$

$$\mu_1 = \Sigma_1\left(\frac{\mu_0}{\Sigma_0} + \frac{y}{\Sigma_y}\right) \tag{4.137}$$

We can rewrite the posterior mean in 3 different ways:

$$\mu_1 = \frac{\Sigma_y}{\Sigma_y + \Sigma_0}\mu_0 + \frac{\Sigma_0}{\Sigma_y + \Sigma_0}y \tag{4.138}$$

$$= \mu_0 + (y - \mu_0)\frac{\Sigma_0}{\Sigma_y + \Sigma_0} \tag{4.139}$$

$$= y - (y - \mu_0)\frac{\Sigma_y}{\Sigma_y + \Sigma_0} \tag{4.140}$$

The first equation is a convex combination of the prior and the data. The second equation is the prior mean adjusted towards the data. The third equation is the data adjusted towards the prior mean; this is called **shrinkage**. These are all equivalent ways of expressing the tradeoff between likelihood and prior. If Σ_0 is small relative to Σ_Y, corresponding to a strong prior, the amount of shrinkage is large (see Figure 4.12(a)), whereas if Σ_0 is large relative to Σ_y, corresponding to a weak prior, the amount of shrinkage is small (see Figure 4.12(b)).

Another way to quantify the amount of shrinkage is in terms of the **signal-to-noise ratio**, which is defined as follows:

$$\text{SNR} \triangleq \frac{\mathbb{E}\left[X^2\right]}{\mathbb{E}\left[\epsilon^2\right]} = \frac{\Sigma_0 + \mu_0^2}{\Sigma_y} \tag{4.141}$$

where $x \sim \mathcal{N}(\mu_0, \Sigma_0)$ is the true signal, $y = x + \epsilon$ is the observed signal, and $\epsilon \sim \mathcal{N}(0, \Sigma_y)$ is the noise term.

4.4.2.2 Inferring an unknown vector from noisy measurements

Now consider N vector-valued observations, $\mathbf{y}_i \sim \mathcal{N}(\mathbf{x}, \mathbf{\Sigma}_y)$, and a Gaussian prior, $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}_0, \mathbf{\Sigma}_0)$. Setting $\mathbf{A} = \mathbf{I}$, $\mathbf{b} = \mathbf{0}$, and using $\overline{\mathbf{y}}$ for the effective observation with precision $N\mathbf{\Sigma}_y^{-1}$, we have

$$p(\mathbf{x}|\mathbf{y}_1, \ldots, \mathbf{y}_N) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_N, \mathbf{\Sigma}_N) \tag{4.142}$$

$$\mathbf{\Sigma}_N^{-1} = \mathbf{\Sigma}_0^{-1} + N\mathbf{\Sigma}_y^{-1} \tag{4.143}$$

$$\boldsymbol{\mu}_N = \mathbf{\Sigma}_N(\mathbf{\Sigma}_y^{-1}(N\overline{\mathbf{y}}) + \mathbf{\Sigma}_0^{-1}\boldsymbol{\mu}_0) \tag{4.144}$$

See Figure 4.13 for a 2d example. We can think of $\mathbf{x}$ as representing the true, but unknown, location of an object in 2d space, such as a missile or airplane, and the $\mathbf{y}_i$ as being noisy observations, such as radar "blips". As we receive more blips, we are better able to localize the source. In Section 18.3.1, we will see how to extend this example to track moving objects using the famous Kalman filter algorithm.

Now suppose we have multiple measuring devices, and we want to combine them together; this is known as **sensor fusion**. If we have multiple observations with different covariances (corresponding to sensors with different reliabilities), the posterior will be an appropriate weighted average of the data. Consider the example in Figure 4.14. We use an uninformative prior on $\mathbf{x}$, namely $p(\mathbf{x}) = \mathcal{N}(\boldsymbol{\mu}_0, \mathbf{\Sigma}_0) = \mathcal{N}(\mathbf{0}, 10^{10}\mathbf{I}_2)$. We get 2 noisy observations, $\mathbf{y}_1 \sim \mathcal{N}(\mathbf{x}, \mathbf{\Sigma}_{y,1})$ and $\mathbf{y}_2 \sim \mathcal{N}(\mathbf{x}, \mathbf{\Sigma}_{y,2})$. We then compute $p(\mathbf{x}|\mathbf{y}_1, \mathbf{y}_2)$.

In Figure 4.14(a), we set $\mathbf{\Sigma}_{y,1} = \mathbf{\Sigma}_{y,2} = 0.01\mathbf{I}_2$, so both sensors are equally reliable. In this case, the posterior mean is half way between the two observations, $\mathbf{y}_1$ and $\mathbf{y}_2$. In Figure 4.14(b), we set $\mathbf{\Sigma}_{y,1} = 0.05\mathbf{I}_2$ and $\mathbf{\Sigma}_{y,2} = 0.01\mathbf{I}_2$, so sensor 2 is more reliable than sensor 1. In this case, the posterior mean is closer to $\mathbf{y}_2$. In Figure 4.14(c), we set

$$\mathbf{\Sigma}_{y,1} = 0.01\begin{pmatrix} 10 & 1 \\ 1 & 1 \end{pmatrix}, \quad \mathbf{\Sigma}_{y,2} = 0.01\begin{pmatrix} 1 & 1 \\ 1 & 10 \end{pmatrix} \tag{4.145}$$

so sensor 1 is more reliable in the y_2 component (vertical direction), and sensor 2 is more reliable in the y_1 component (horizontal direction). In this case, the posterior mean uses $\mathbf{y}_1$'s vertical component and $\mathbf{y}_2$'s horizontal component.

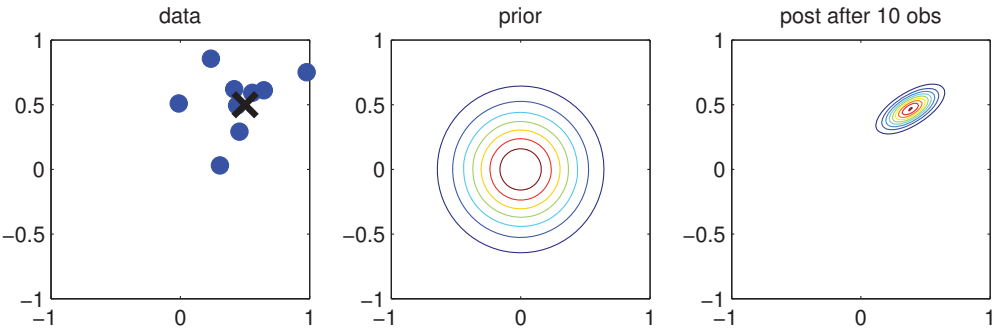


Figure 4.13 Illustration of Bayesian inference for the mean of a 2d Gaussian. (a) The data is generated from $\mathbf{y}_i \sim \mathcal{N}(\mathbf{x}, \mathbf{\Sigma}_y)$, where $\mathbf{x} = [0.5, 0.5]^T$ and $\mathbf{\Sigma}_y = 0.1[2, 1; 1, 1]$). We assume the sensor noise covariance $\mathbf{\Sigma}_y$ is known but $\mathbf{x}$ is unknown. The black cross represents $\mathbf{x}$. (b) The prior is $p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\mathbf{0}, 0.1\mathbf{I}_2)$. (c) We show the posterior after 10 data points have been observed. Figure generated by `gaussInferParamsMean2d`.

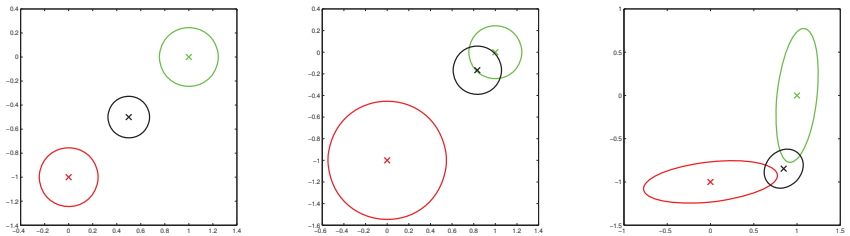


Figure 4.14 We observe $\mathbf{y}_1 = (0, -1)$ (red cross) and $\mathbf{y}_2 = (1, 0)$ (green cross) and infer $E(\boldsymbol{\mu}|\mathbf{y}_1, \mathbf{y}_2, \boldsymbol{\theta})$ (black cross). (a) Equally reliable sensors, so the posterior mean estimate is in between the two circles. (b) Sensor 2 is more reliable, so the estimate shifts more towards the green circle. (c) Sensor 1 is more reliable in the vertical direction, Sensor 2 is more reliable in the horizontal direction. The estimate is an appropriate combination of the two measurements. Figure generated by `sensorFusion2d`.

Note that this technique crucially relies on modeling our uncertainty of each sensor; computing an unweighted average would give the wrong result. However, we have assumed the sensor precisions are known. When they are not, we should model out uncertainty about $\mathbf{\Sigma}_1$ and $\mathbf{\Sigma}_2$ as well. See Section 4.6.4 for details.

4.4.2.3 Interpolating noisy data

We now revisit the example of Section 4.3.2.2. This time we no longer assume noise-free observations. Instead, let us assume that we obtain N noisy observations y_i; without loss of generality, assume these correspond to $x_1, \ldots, x_N$. We can model this setup as a linear

Gaussian system:

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \boldsymbol{\epsilon} \tag{4.146}$$

where $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_y)$, $\boldsymbol{\Sigma}_y = \sigma^2\mathbf{I}$, σ^2 is the observation noise, and $\mathbf{A}$ is a $N \times D$ projection matrix that selects out the observed elements. For example, if $N = 2$ and $D = 4$ we have

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \tag{4.147}$$

Using the same improper prior as before, $\boldsymbol{\Sigma}_x = (\mathbf{L}^T\mathbf{L})^{-1}$, we can easily compute the posterior mean and variance. In Figure 4.15, we plot the posterior mean, posterior variance, and some posterior samples. Now we see that the prior precision λ effects the posterior mean as well as the posterior variance. In particular, for a strong prior (large λ), the estimate is very smooth, and the uncertainty is low. but for a weak prior (small λ), the estimate is wiggly, and the uncertainty (away from the data) is high.

The posterior mean can also be computed by solving the following optimization problem:

$$\min_{\mathbf{x}} \frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i - y_i)^2 + \frac{\lambda}{2}\sum_{j=1}^{D}\left[(x_j - x_{j-1})^2 + (x_j - x_{j+1})^2\right] \tag{4.148}$$

where we have defined $x_0 = x_1$ and $x_{D+1} = x_D$ for notational simplicity. We recognize this as a discrete approximation to the following problem:

$$\min_{f} \frac{1}{2\sigma^2}\int (f(t) - y(t))^2 dt + \frac{\lambda}{2}\int [f'(t)]^2 dt \tag{4.149}$$

where $f'(t)$ is the first derivative of f. The first term measures fit to the data, and the second term penalizes functions that are "too wiggly". This is an example of **Tikhonov regularization**, which is a popular approach to **functional data analysis**. See Chapter 15 for more sophisticated approaches, which enforce higher order smoothness (so the resulting samples look less "jagged").

4.4.3 Proof of the result *

We now derive Equation 4.125. The basic idea is to derive the joint distribution, $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y}|\mathbf{x})$, and then to use the results from Section 4.3.1 for computing $p(\mathbf{x}|\mathbf{y})$.

In more detail, we proceed as follows. The log of the joint distribution is as follows (dropping irrelevant constants):

$$\log p(\mathbf{x}, \mathbf{y}) = -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_x)^T\boldsymbol{\Sigma}_x^{-1}(\mathbf{x} - \boldsymbol{\mu}_x) - \frac{1}{2}(\mathbf{y} - \mathbf{A}\mathbf{x} - \mathbf{b})^T\boldsymbol{\Sigma}_y^{-1}(\mathbf{y} - \mathbf{A}\mathbf{x} - \mathbf{b}) \tag{4.150}$$

This is clearly a joint Gaussian distribution, since it is the exponential of a quadratic form.

Expanding out the quadratic terms involving $\mathbf{x}$ and $\mathbf{y}$, and ignoring linear and constant terms, we have

$$\begin{aligned} Q &= -\frac{1}{2}\mathbf{x}^T\boldsymbol{\Sigma}_x^{-1}\mathbf{x} - \frac{1}{2}\mathbf{y}^T\boldsymbol{\Sigma}_y^{-1}\mathbf{y} - \frac{1}{2}(\mathbf{A}\mathbf{x})^T\boldsymbol{\Sigma}_y^{-1}(\mathbf{A}\mathbf{x}) + \mathbf{y}^T\boldsymbol{\Sigma}_y^{-1}\mathbf{A}\mathbf{x} && (4.151) \\ &= -\frac{1}{2}\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix}^T\begin{pmatrix}\boldsymbol{\Sigma}_x^{-1} + \mathbf{A}^T\boldsymbol{\Sigma}_y^{-1}\mathbf{A} & -\mathbf{A}^T\boldsymbol{\Sigma}_y^{-1} \\ -\boldsymbol{\Sigma}_y^{-1}\mathbf{A} & \boldsymbol{\Sigma}_y^{-1}\end{pmatrix}\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix} && (4.152) \\ &= -\frac{1}{2}\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix}^T\boldsymbol{\Sigma}^{-1}\begin{pmatrix}\mathbf{x}\\ \mathbf{y}\end{pmatrix} && (4.153) \end{aligned}$$

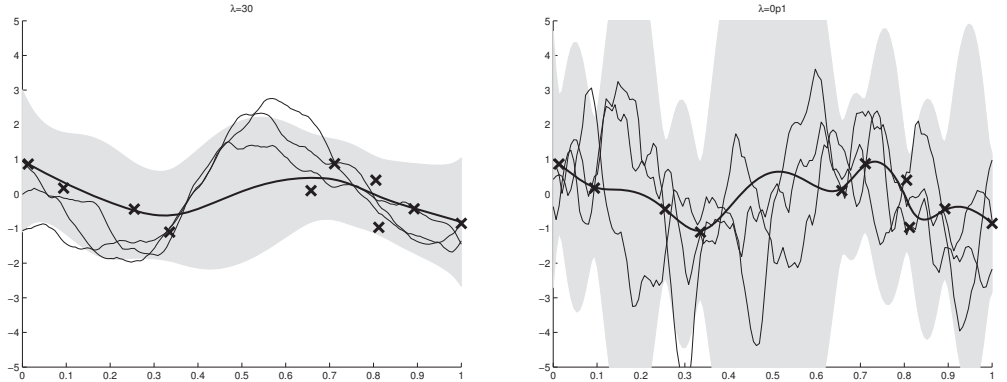


Figure 4.15 Interpolating noisy data (noise variance $\sigma^2 = 1$) using a Gaussian with prior precision λ. (a) $\lambda = 30$. (b) $\lambda = 0.01$. See also Figure 4.10. Based on Figure 7.1 of (Calvetti and Somersalo 2007). Figure generated by `gaussInterpNoisyDemo`. See also `splineBasisDemo`.

where the precision matrix of the joint is defined as

$$\boldsymbol{\Sigma}^{-1} = \begin{pmatrix} \boldsymbol{\Sigma}_x^{-1} + \mathbf{A}^T \boldsymbol{\Sigma}_y^{-1} \mathbf{A} & -\mathbf{A}^T \boldsymbol{\Sigma}_y^{-1} \\ -\boldsymbol{\Sigma}_y^{-1} \mathbf{A} & \boldsymbol{\Sigma}_y^{-1} \end{pmatrix} \triangleq \boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{\Lambda}_{xx} & \boldsymbol{\Lambda}_{xy} \\ \boldsymbol{\Lambda}_{yx} & \boldsymbol{\Lambda}_{yy} \end{pmatrix} \tag{4.154}$$

From Equation 4.69, and using the fact that $\boldsymbol{\mu}_y = \mathbf{A}\boldsymbol{\mu}_x + \mathbf{b}$, we have

$$\begin{aligned} p(\mathbf{x}|\mathbf{y}) &= \mathcal{N}(\boldsymbol{\mu}_{x|y}, \boldsymbol{\Sigma}_{x|y}) && (4.155) \\ \boldsymbol{\Sigma}_{x|y} &= \boldsymbol{\Lambda}_{xx}^{-1} = (\boldsymbol{\Sigma}_x^{-1} + \mathbf{A}^T \boldsymbol{\Sigma}_y^{-1} \mathbf{A})^{-1} && (4.156) \\ \boldsymbol{\mu}_{x|y} &= \boldsymbol{\Sigma}_{x|y} \left(\boldsymbol{\Lambda}_{xx} \boldsymbol{\mu}_x - \boldsymbol{\Lambda}_{xy} (\mathbf{y} - \boldsymbol{\mu}_y) \right) && (4.157) \\ &= \boldsymbol{\Sigma}_{x|y} \left(\boldsymbol{\Sigma}_x^{-1} \boldsymbol{\mu} + \mathbf{A}^T \boldsymbol{\Sigma}_y^{-1} (\mathbf{y} - \mathbf{b}) \right) && (4.158) \end{aligned}$$

4.5 Digression: The Wishart distribution *

The **Wishart** distribution is the generalization of the Gamma distribution to positive definite matrices. Press (Press 2005, p107) has said "The Wishart distribution ranks next to the (multivariate) normal distribution in order of importance and usefuleness in multivariate statistics". We will mostly use it to model our uncertainty in covariance matrices, $\boldsymbol{\Sigma}$, or their inverses, $\boldsymbol{\Lambda} = \boldsymbol{\Sigma}^{-1}$.

The pdf of the Wishart is defined as follows:

$$\mathrm{Wi}(\boldsymbol{\Lambda}|\mathbf{S}, \nu) = \frac{1}{Z_{\mathrm{Wi}}} |\boldsymbol{\Lambda}|^{(\nu - D - 1)/2} \exp\left(-\frac{1}{2} \mathrm{tr}(\boldsymbol{\Lambda}\mathbf{S}^{-1}) \right) \tag{4.159}$$

Here ν is called the "degrees of freedom" and $\mathbf{S}$ is the "scale matrix". (We shall get more intuition for these parameters shortly.) The normalization constant for this distribution (which

requires integrating over all symmetric pd matrices) is the following formidable expression

$$Z_{\text{Wi}} = 2^{\nu D/2}\Gamma_D(\nu/2)|\mathbf{S}|^{\nu/2} \tag{4.160}$$

where $\Gamma_D(a)$ is the **multivariate gamma function**:

$$\Gamma_D(x) = \pi^{D(D-1)/4}\prod_{i=1}^{D}\Gamma\left(x + (1-i)/2\right) \tag{4.161}$$

Hence $\Gamma_1(a) = \Gamma(a)$ and

$$\Gamma_D(\nu_0/2) = \prod_{i=1}^{D}\Gamma(\frac{\nu_0 + 1 - i}{2}) \tag{4.162}$$

The normalization constant only exists (and hence the pdf is only well defined) if $\nu > D - 1$.

There is a connection between the Wishart distribution and the Gaussian. In particular, let $\mathbf{x}_i \sim \mathcal{N}(0, \mathbf{\Sigma})$. Then the scatter matrix $\mathbf{S} = \sum_{i=1}^{N}\mathbf{x}_i\mathbf{x}_i^T$ has a Wishart distribution: $\mathbf{S} \sim \text{Wi}(\mathbf{\Sigma}, 1)$. Hence $\mathbb{E}[\mathbf{S}] = N\mathbf{\Sigma}$. More generally, one can show that the mean and mode of $\text{Wi}(\mathbf{S}, \nu)$ are given by

$$\text{mean} = \nu\mathbf{S}, \ \text{mode} = (\nu - D - 1)\mathbf{S} \tag{4.163}$$

where the mode only exists if $\nu > D + 1$.

If $D = 1$, the Wishart reduces to the Gamma distribution:

$$\text{Wi}(\lambda|s^{-1}, \nu) = \text{Ga}(\lambda|\frac{\nu}{2}, \frac{s}{2}) \tag{4.164}$$

4.5.1 Inverse Wishart distribution

Recall that we showed (Exercise 2.10) that if $\lambda \sim \text{Ga}(a, b)$, then that $\frac{1}{\lambda} \sim \text{IG}(a, b)$. Similarly, if $\mathbf{\Sigma}^{-1} \sim \text{Wi}(\mathbf{S}, \nu)$ then $\mathbf{\Sigma} \sim \text{IW}(\mathbf{S}^{-1}, \nu + D + 1)$, where IW is the **inverse Wishart**, the multidimensional generalization of the inverse Gamma. It is defined as follows, for $\nu > D - 1$ and $\mathbf{S} \succ 0$:

$$\text{IW}(\mathbf{\Sigma}|\mathbf{S}, \nu) = \frac{1}{Z_{IW}}|\mathbf{\Sigma}|^{-(\nu+D+1)/2}\exp\left(-\frac{1}{2}\text{tr}(\mathbf{S}^{-1}\mathbf{\Sigma}^{-1})\right) \tag{4.165}$$

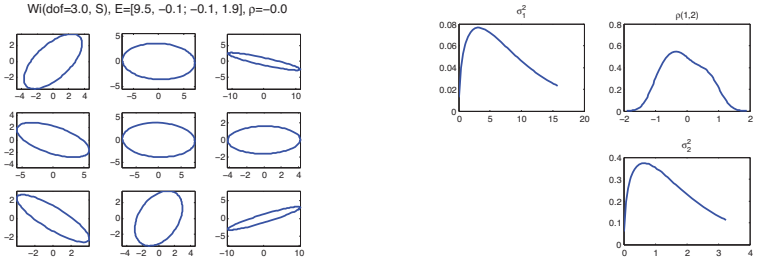
$$Z_{IW} = |\mathbf{S}|^{-\nu/2}2^{\nu D/2}\Gamma_D(\nu/2) \tag{4.166}$$

One can show that the distribution has these properties

$$\text{mean} = \frac{\mathbf{S}^{-1}}{\nu - D - 1}, \quad \text{mode} = \frac{\mathbf{S}^{-1}}{\nu + D + 1} \tag{4.167}$$

If $D = 1$, this reduces to the inverse Gamma:

$$\text{IW}(\sigma^2|S^{-1}, \nu) = \text{IG}(\sigma^2|\nu/2, S/2) \tag{4.168}$$



(a) (b)

Figure 4.16 Visualization of the Wishart distribution. Left: Some samples from the Wishart distribution, $\mathbf{\Sigma} \sim \text{Wi}(\mathbf{S}, \nu)$, where $\mathbf{S} = [3.1653, -0.0262; -0.0262, 0.6477]$ and $\nu = 3$. Right: Plots of the marginals (which are Gamma), and the approximate (sample-based) marginal on the correlation coefficient. If $\nu = 3$ there is a lot of uncertainty about the value of the correlation coefficient ρ (see the almost uniform distribution on $[-1, 1]$). The sampled matrices are highly variable, and some are nearly singular. As ν increases, the sampled matrices are more concentrated on the prior $\mathbf{S}$. Figure generated by `wiPlotDemo`.

4.5.2 Visualizing the Wishart distribution *

Since the Wishart is a distribution over matrices, it is hard to plot as a density function. However, we can easily sample from it, and in the 2d case, we can use the eigenvectors of the resulting matrix to define an ellipse, as explained in Section 4.1.2. See Figure 4.16 for some examples.

For higher dimensional matrices, we can plot marginals of the distribution. The diagonals of a Wishart distributed matrix have Gamma distributions, so are easy to plot. It is hard in general to work out the distribution of the off-diagonal elements, but we can sample matrices from the distribution, and then compute the distribution empirically. In particular, we can convert each sampled matrix to a correlation matrix, and thus compute a Monte Carlo approximation (Section 2.7) to the expected correlation coefficients:

$$\mathbb{E}\left[R_{ij}\right] \approx \frac{1}{S} \sum_{s=1}^{S} \mathbf{R}(\mathbf{\Sigma}^{(s)})_{ij} \tag{4.169}$$

where $\mathbf{\Sigma}^{(s)} \sim \text{Wi}(\mathbf{\Sigma}, \nu)$ and $\mathbf{R}(\mathbf{\Sigma})$ converts matrix $\mathbf{\Sigma}$ into a correlation matrix:

$$R_{ij} = \frac{\mathbf{\Sigma}_{ij}}{\sqrt{\mathbf{\Sigma}_{ii}\mathbf{\Sigma}_{jj}}} \tag{4.170}$$

We can then use kernel density estimation (Section 14.7.2) to produce a smooth approximation to the univariate density $\mathbb{E}\left[R_{ij}\right]$ for plotting purposes. See Figure 4.16 for some examples.

4.6 Inferring the parameters of an MVN

So far, we have discussed inference in a Gaussian assuming the parameters $\boldsymbol{\theta} = (\boldsymbol{\mu}, \mathbf{\Sigma})$ are known. We now discuss how to infer the parameters themselves. We will assume the data has

the form $\mathbf{x}_i \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for $i = 1 : N$ and is fully observed, so we have no missing data (see Section 11.6.1 for how to estimate parameters of an MVN in the presence of missing values). To simplify the presentation, we derive the posterior in three parts: first we compute $p(\boldsymbol{\mu}|\mathcal{D}, \boldsymbol{\Sigma})$; then we compute $p(\boldsymbol{\Sigma}|\mathcal{D}, \boldsymbol{\mu})$; finally we compute the joint $p(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathcal{D})$.

4.6.1 Posterior distribution of $\boldsymbol{\mu}$

We have discussed how to compute the MLE for $\boldsymbol{\mu}$; we now discuss how to compute its posterior, which is useful for modeling our uncertainty about its value.

The likelihood has the form

$$p(\mathcal{D}|\boldsymbol{\mu}) = \mathcal{N}(\bar{\mathbf{x}}|\boldsymbol{\mu}, \frac{1}{N}\boldsymbol{\Sigma}) \tag{4.171}$$

For simplicity, we will use a conjugate prior, which in this case is a Gaussian. In particular, if $p(\boldsymbol{\mu}) = \mathcal{N}(\boldsymbol{\mu}|\mathbf{m}_0, \mathbf{V}_0)$ then we can derive a Gaussian posterior for $\boldsymbol{\mu}$ based on the results in Section 4.4.2.2. We get

$$\begin{aligned} p(\boldsymbol{\mu}|\mathcal{D}, \boldsymbol{\Sigma}) &= \mathcal{N}(\boldsymbol{\mu}|\mathbf{m}_N, \mathbf{V}_N) && (4.172)\\ \mathbf{V}_N^{-1} &= \mathbf{V}_0^{-1} + N\boldsymbol{\Sigma}^{-1} && (4.173)\\ \mathbf{m}_N &= \mathbf{V}_N(\boldsymbol{\Sigma}^{-1}(N\bar{\mathbf{x}}) + \mathbf{V}_0^{-1}\mathbf{m}_0) && (4.174) \end{aligned}$$

This is exactly the same process as inferring the location of an object based on noisy radar "blips", except now we are inferring the mean of a distribution based on noisy samples. (To a Bayesian, there is no difference between uncertainty about parameters and uncertainty about anything else.)

We can model an uninformative prior by setting $\mathbf{V}_0 = \infty\mathbf{I}$. In this case we have $p(\boldsymbol{\mu}|\mathcal{D}, \boldsymbol{\Sigma}) = \mathcal{N}(\bar{\mathbf{x}}, \frac{1}{N}\boldsymbol{\Sigma})$, so the posterior mean is equal to the MLE. We also see that the posterior variance goes down as $1/N$, which is a standard result from frequentist statistics.

4.6.2 Posterior distribution of $\boldsymbol{\Sigma}$ *

We now discuss how to compute $p(\boldsymbol{\Sigma}|\mathcal{D}, \boldsymbol{\mu})$. The likelihood has the form

$$p(\mathcal{D}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \propto |\boldsymbol{\Sigma}|^{-\frac{N}{2}} \exp\left(-\frac{1}{2}\mathrm{tr}(\mathbf{S}_\mu\boldsymbol{\Sigma}^{-1})\right) \tag{4.175}$$

The corresponding conjugate prior is known as the inverse Wishart distribution (Section 4.5.1). Recall that this has the following pdf:

$$\mathrm{IW}(\boldsymbol{\Sigma}|\mathbf{S}_0^{-1}, \nu_0) \propto |\boldsymbol{\Sigma}|^{-(\nu_0+D+1)/2} \exp\left(-\frac{1}{2}\mathrm{tr}(\mathbf{S}_0\boldsymbol{\Sigma}^{-1})\right) \tag{4.176}$$

Here $\nu_0 > D - 1$ is the degrees of freedom (dof), and $\mathbf{S}_0$ is a symmetric pd matrix. We see that $\mathbf{S}_0^{-1}$ plays the role of the prior scatter matrix, and $N_0 \triangleq \nu_0 + D + 1$ controls the strength of the prior, and hence plays a role analogous to the sample size N.

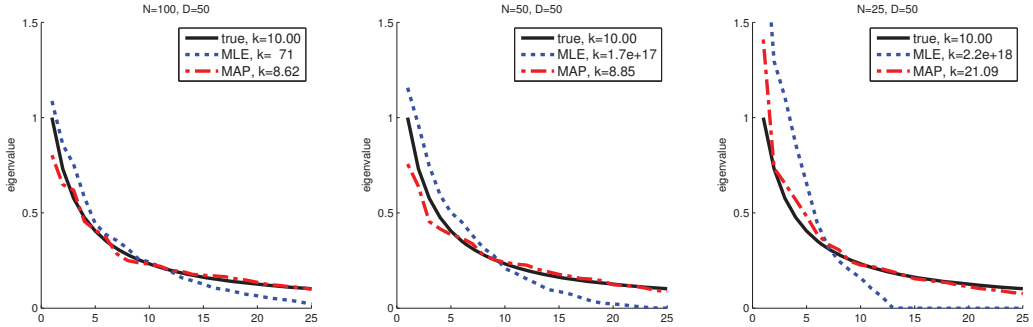


Figure 4.17 Estimating a covariance matrix in $D = 50$ dimensions using $N \in \{100, 50, 25\}$ samples. We plot the eigenvalues in descending order for the true covariance matrix (solid black), the MLE (dotted blue) and the MAP estimate (dashed red), using Equation 4.184 with $\lambda = 0.9$. We also list the condition number of each matrix in the legend. Based on Figure 1 of (Schaefer and Strimmer 2005). Figure generated by `shrinkcovDemo`.

Multiplying the likelihood and prior we find that the posterior is also inverse Wishart:

$$\begin{aligned} p(\boldsymbol{\Sigma}|\mathcal{D}, \boldsymbol{\mu}) &\propto |\boldsymbol{\Sigma}|^{-\frac{N}{2}} \exp\left(-\frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S}_\mu)\right) |\boldsymbol{\Sigma}|^{-(\nu_0+D+1)/2} \\ &\quad \exp\left(-\frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S}_0)\right) \quad (4.177) \\ &= |\boldsymbol{\Sigma}|^{-\frac{N+(\nu_0+D+1)}{2}} \exp\left(-\frac{1}{2}\mathrm{tr}\left[\boldsymbol{\Sigma}^{-1}(\mathbf{S}_\mu + \mathbf{S}_0)\right]\right) \quad (4.178) \\ &= \mathrm{IW}(\boldsymbol{\Sigma}|\mathbf{S}_N, \nu_N) \quad (4.179) \\ \nu_N &= \nu_0 + N \quad (4.180) \\ \mathbf{S}_N^{-1} &= \mathbf{S}_0 + \mathbf{S}_\mu \quad (4.181) \end{aligned}$$

In words, this says that the posterior strength ν_N is the prior strength ν_0 plus the number of observations N, and the posterior scatter matrix $\mathbf{S}_N$ is the prior scatter matrix $\mathbf{S}_0$ plus the data scatter matrix $\mathbf{S}_\mu$.

4.6.2.1 MAP estimation

We see from Equation 4.7 that $\hat{\boldsymbol{\Sigma}}_{mle}$ is a rank $\min(N, D)$ matrix. If $N < D$, this is not full rank, and hence will be uninvertible. And even if $N > D$, it may be the case that $\hat{\boldsymbol{\Sigma}}$ is ill-conditioned (meaning it is nearly singular).

To solve these problems, we can use the posterior mode (or mean). One can show (using techniques analogous to the derivation of the MLE) that the MAP estimate is given by

$$\hat{\boldsymbol{\Sigma}}_{map} = \frac{\mathbf{S}_N}{\nu_N + D + 1} = \frac{\mathbf{S}_0 + \mathbf{S}_\mu}{N_0 + N} \quad (4.182)$$

If we use an improper uniform prior, corresponding to $N_0 = 0$ and $\mathbf{S}_0 = \mathbf{0}$, we recover the MLE.

Let us now consider the use of a proper informative prior, which is necessary whenever D/N is large (say bigger than 0.1). Let $\boldsymbol{\mu} = \overline{\mathbf{x}}$, so $\mathbf{S}_\mu = \mathbf{S}_{\overline{x}}$. Then we can rewrite the MAP estimate as a convex combination of the prior mode and the MLE. To see this, let $\boldsymbol{\Sigma}_0 \triangleq \frac{\mathbf{S}_0}{N_0}$ be the prior mode. Then the posterior mode can be rewritten as

$$\hat{\boldsymbol{\Sigma}}_{map} = \frac{\mathbf{S}_0 + \mathbf{S}_{\overline{x}}}{N_0 + N} = \frac{N_0}{N_0 + N}\frac{\mathbf{S}_0}{N_0} + \frac{N}{N_0 + N}\frac{\mathbf{S}}{N} = \lambda\boldsymbol{\Sigma}_0 + (1-\lambda)\hat{\boldsymbol{\Sigma}}_{mle} \tag{4.183}$$

where $\lambda = \frac{N_0}{N_0+N}$, controls the amount of shrinkage towards the prior.

This begs the question: where do the parameters of the prior come from? It is common to set λ by cross validation. Alternatively, we can use the closed-form formula provided in (Ledoit and Wolf 2004b,a; Schaefer and Strimmer 2005), which is the optimal frequentist estimate if we use squared loss. This is arguably not the most natural loss function for covariance matrices (because it ignores the postive definite constraint), but it results in a simple estimator, which is implemented in the PMTK function `shrinkcov`. We discuss Bayesian ways of estimating λ later.

As for the prior covariance matrix, $\mathbf{S}_0$, it is common to use the following (data dependent) prior: $\mathbf{S}_0 = \mathrm{diag}(\hat{\boldsymbol{\Sigma}}_{mle})$. In this case, the MAP estimate is given by

$$\hat{\boldsymbol{\Sigma}}_{map}(i,j) = \begin{cases} \hat{\boldsymbol{\Sigma}}_{mle}(i,j) & \text{if } i = j \\ (1-\lambda)\hat{\boldsymbol{\Sigma}}_{mle}(i,j) & \text{otherwise} \end{cases} \tag{4.184}$$

Thus we see that the diagonal entries are equal to their ML estimates, and the off diagonal elements are "shrunk" somewhat towards 0. This technique is therefore called **shrinkage estimation**, or **regularized estimation**.

The benefits of MAP estimation are illustrated in Figure 4.17. We consider fitting a 50 dimensional Gaussian to $N = 100$, $N = 50$ and $N = 25$ data points. We see that the MAP estimate is always well-conditioned, unlike the MLE. In particular, we see that the **eigenvalue spectrum** of the MAP estimate is much closer to that of the true matrix than the MLE's. The eigenvectors, however, are unaffected.

The importance of regularizing the estimate of $\boldsymbol{\Sigma}$ will become apparent in later chapters, when we consider fitting covariance matrices to high dimensional data.

4.6.2.2 Univariate posterior

In the 1d case, the likelihood has the form

$$p(\mathcal{D}|\sigma^2) \propto (\sigma^2)^{-N/2}\exp\left(-\frac{1}{2\sigma^2}\sum_{i=1}^{N}(x_i-\mu)^2\right) \tag{4.185}$$

The standard conjugate prior is the inverse Gamma distribution, which is just the scalar version of the inverse Wishart:

$$\mathrm{IG}(\sigma^2|a_0,b_0) \propto (\sigma^2)^{-(a_0+1)}\exp(-\frac{b_0}{\sigma^2}) \tag{4.186}$$

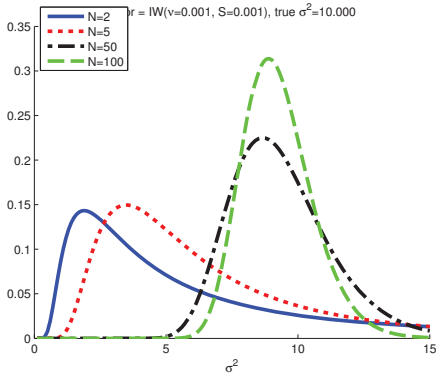


Figure 4.18 Sequential updating of the posterior for σ^2 starting from an uninformative prior. The data was generated from a Gaussian with known mean $\mu = 5$ and unknown variance $\sigma^2 = 10$. Figure generated by `gaussSeqUpdateSigma1D`.

Multiplying the likelihood and the prior, we see that the posterior is also IG:

$$p(\sigma^2|\mathcal{D}) = \text{IG}(\sigma^2|a_N, b_N) \tag{4.187}$$

$$a_N = a_0 + N/2 \tag{4.188}$$

$$b_N = b_0 + \frac{1}{2}\sum_{i=1}^{N}(x_i - \mu)^2 \tag{4.189}$$

See Figure 4.18 for an illustration.

The form of the posterior is not quite as pretty as the multivariate case, because of the factors of $\frac{1}{2}$. This arises because $\text{IW}(\sigma^2|s_0, \nu_0) = \text{IG}(\sigma^2|\frac{s_0}{2}, \frac{\nu_0}{2})$. Another problem with using the $\text{IG}(a_0, b_0)$ distribution is that the strength of the prior is encoded in both a_0 and b_0. To avoid both of these problems, it is common (in the statistics literature) to use an alternative parameterization of the IG distribution, known as the (scaled) **inverse chi-squared distribution**. This is defined as follows:

$$\chi^{-2}(\sigma^2|\nu_0, \sigma_0^2) = \text{IG}(\sigma^2|\frac{\nu_0}{2}, \frac{\nu_0\sigma_0^2}{2}) \propto (\sigma^2)^{-\nu_0/2-1}\exp(-\frac{\nu_0\sigma_0^2}{2\sigma^2}) \tag{4.190}$$

Here ν_0 controls the strength of the prior, and σ_0^2 encodes the value of the prior. With this prior, the posterior becomes

$$p(\sigma^2|\mathcal{D}, \mu) = \chi^{-2}(\sigma^2|\nu_N, \sigma_N^2) \tag{4.191}$$

$$\nu_N = \nu_0 + N \tag{4.192}$$

$$\sigma_N^2 = \frac{\nu_0\sigma_0^2 + \sum_{i=1}^{N}(x_i - \mu)^2}{\nu_N} \tag{4.193}$$

We see that the posterior dof ν_N is the prior dof ν_0 plus N, and the posterior sum of squares $\nu_N\sigma_N^2$ is the prior sum of squares $\nu_0\sigma_0^2$ plus the data sum of squares.

We can emulate an uninformative prior, $p(\sigma^2) \propto \sigma^{-2}$, by setting $\nu_0 = 0$, which makes intuitive sense (since it corresponds to a zero virtual sample size).

4.6.3 Posterior distribution of μ and Σ *

We now discuss how to compute $p(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathcal{D})$. These results are a bit complex, but will prove useful later on in this book. Feel free to skip this section on a first reading.

4.6.3.1 Likelihood

The likelihood is given by

$$p(\mathcal{D}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-ND/2}|\Sigma|^{-\frac{N}{2}} \exp\left(-\frac{1}{2}\sum_{i=1}^{N}(\mathbf{x}_i - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu})\right) \tag{4.194}$$

Now one can show that

$$\sum_{i=1}^{N}(\mathbf{x}_i - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\mathbf{x}_i - \boldsymbol{\mu}) = \mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S}_{\overline{x}}) + N(\overline{\mathbf{x}} - \boldsymbol{\mu})^T\boldsymbol{\Sigma}^{-1}(\overline{\mathbf{x}} - \boldsymbol{\mu}) \tag{4.195}$$

Hence we can rewrite the likelihood as follows:

$$p(\mathcal{D}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = (2\pi)^{-ND/2}|\Sigma|^{-\frac{N}{2}} \exp\left(-\frac{N}{2}(\boldsymbol{\mu} - \overline{\mathbf{x}})^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu} - \overline{\mathbf{x}})\right) \tag{4.196}$$

$$\exp\left(-\frac{N}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S}_{\overline{x}})\right) \tag{4.197}$$

We will use this form below.

4.6.3.2 Prior

The obvious prior to use is the following

$$p(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \mathcal{N}(\boldsymbol{\mu}|\mathbf{m}_0, \mathbf{V}_0)\mathrm{IW}(\boldsymbol{\Sigma}|\mathbf{S}_0, \nu_0) \tag{4.198}$$

Unfortunately, this is not conjugate to the likelihood. To see why, note that $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ appear together in a non-factorized way in the likelihood; hence they will also be coupled together in the posterior.

The above prior is sometimes called **semi-conjugate** or **conditionally conjugate**, since both conditionals, $p(\boldsymbol{\mu}|\boldsymbol{\Sigma})$ and $p(\boldsymbol{\Sigma}|\boldsymbol{\mu})$, are individually conjugate. To create a full conjugate prior, we need to use a prior where $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$ are dependent on each other. We will use a joint distribution of the form

$$p(\boldsymbol{\mu}, \boldsymbol{\Sigma}) = p(\boldsymbol{\Sigma})p(\boldsymbol{\mu}|\boldsymbol{\Sigma}) \tag{4.199}$$

Looking at the form of the likelihood equation, Equation 4.197, we see that a natural conjugate

prior has the form of a **Normal-inverse-wishart** or **NIW** distribution, defined as follows:

$$\mathrm{NIW}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathbf{m}_0, \kappa_0, \nu_0, \mathbf{S}_0) \triangleq \tag{4.200}$$

$$\mathcal{N}(\boldsymbol{\mu}|\mathbf{m}_0, \frac{1}{\kappa_0}\boldsymbol{\Sigma}) \times \mathrm{IW}(\boldsymbol{\Sigma}|\mathbf{S}_0, \nu_0) \tag{4.201}$$

$$= \frac{1}{Z_{NIW}}|\boldsymbol{\Sigma}|^{-\frac{1}{2}} \exp\left(-\frac{\kappa_0}{2}(\boldsymbol{\mu}-\mathbf{m}_0)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}-\mathbf{m}_0)\right) \tag{4.202}$$

$$\times|\boldsymbol{\Sigma}|^{-\frac{\nu_0+D+1}{2}} \exp\left(-\frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S}_0)\right) \tag{4.203}$$

$$= \frac{1}{Z_{NIW}}|\boldsymbol{\Sigma}|^{-\frac{\nu_0+D+2}{2}} \tag{4.204}$$

$$\times \exp\left(-\frac{\kappa_0}{2}(\boldsymbol{\mu}-\mathbf{m}_0)^T\boldsymbol{\Sigma}^{-1}(\boldsymbol{\mu}-\mathbf{m}_0) - \frac{1}{2}\mathrm{tr}(\boldsymbol{\Sigma}^{-1}\mathbf{S}_0)\right) \tag{4.205}$$

$$Z_{NIW} = 2^{\nu_0 D/2}\Gamma_D(\nu_0/2)(2\pi/\kappa_0)^{D/2}|\mathbf{S}_0|^{-\nu_0/2} \tag{4.206}$$

where $\Gamma_D(a)$ is the multivariate Gamma function.

The parameters of the NIW can be interpreted as follows: $\mathbf{m}_0$ is our prior mean for $\boldsymbol{\mu}$, and κ_0 is how strongly we believe this prior; and $\mathbf{S}_0$ is (proportional to) our prior mean for $\boldsymbol{\Sigma}$, and ν_0 is how strongly we believe this prior.[3]

One can show (Minka 2000f) that the (improper) uninformative prior has the form

$$\lim_{k\to 0}\mathcal{N}(\mu|\mathbf{m}_0, \boldsymbol{\Sigma}/k)\mathrm{IW}(\boldsymbol{\Sigma}|\mathbf{S}_0, k) \propto |2\pi\boldsymbol{\Sigma}|^{-\frac{1}{2}}|\boldsymbol{\Sigma}|^{-(D+1)/2} \tag{4.207}$$

$$\propto |\boldsymbol{\Sigma}|^{-(\frac{D}{2}+1)} \propto \mathrm{NIW}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathbf{0}, 0, 0, 0\mathbf{I}) \tag{4.208}$$

In practice, it is often better to use a weakly informative data-dependent prior. A common choice (see e.g., (Chipman et al. 2001, p81), (Fraley and Raftery 2007, p6)) is to use $\mathbf{S}_0 = \mathrm{diag}(\mathbf{S}_{\overline{x}})/N$, and $\nu_0 = D+2$, to ensure $\mathbb{E}[\boldsymbol{\Sigma}] = \mathbf{S}_0$, and to set $\boldsymbol{\mu}_0 = \overline{\mathbf{x}}$ and κ_0 to some small number, such as 0.01.

3. Although this prior has four parameters, there are really only three free parameters, since our uncertainty in the mean is proportional to the variance. In particular, if we believe that the variance is large, then our uncertainty in μ must be large too. This makes sense intuitively, since if the data has large spread, it may be hard to pin down its mean. See also Exercise 9.1, where we will see the three free parameters more explicitly. If we want separate "control" over our confidence in $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$, we must use a semi-conjugate prior.

4.6.3.3 Posterior

The posterior can be shown (Exercise 4.11) to be NIW with updated parameters:

$$p(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathcal{D}) = \text{NIW}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathbf{m}_N, \kappa_N, \nu_N, \mathbf{S}_N) \tag{4.209}$$

$$\mathbf{m}_N = \frac{\kappa_0 \mathbf{m}_0 + N\overline{\mathbf{x}}}{\kappa_N} = \frac{\kappa_0}{\kappa_0 + N}\mathbf{m}_0 + \frac{N}{\kappa_0 + N}\overline{\mathbf{x}} \tag{4.210}$$

$$\kappa_N = \kappa_0 + N \tag{4.211}$$

$$\nu_N = \nu_0 + N \tag{4.212}$$

$$\mathbf{S}_N = \mathbf{S}_0 + \mathbf{S}_{\overline{x}} + \frac{\kappa_0 N}{\kappa_0 + N}(\overline{\mathbf{x}} - \mathbf{m}_0)(\overline{\mathbf{x}} - \mathbf{m}_0)^T \tag{4.213}$$

$$= \mathbf{S}_0 + \mathbf{S} + \kappa_0 \mathbf{m}_0 \mathbf{m}_0^T - \kappa_N \mathbf{m}_N \mathbf{m}_N^T \tag{4.214}$$

where we have defined $\mathbf{S} \triangleq \sum_{i=1}^N \mathbf{x}_i \mathbf{x}_i^T$ as the uncentered sum-of-squares matrix (this is easier to update incrementally than the centered version).

This result is actually quite intuitive: the posterior mean is a convex combination of the prior mean and the MLE, with "strength" $\kappa_0 + N$; and the posterior scatter matrix $\mathbf{S}_N$ is the prior scatter matrix $\mathbf{S}_0$ plus the empirical scatter matrix $\mathbf{S}_{\overline{x}}$ plus an extra term due to the uncertainty in the mean (which creates its own virtual scatter matrix).

4.6.3.4 Posterior mode

The mode of the joint distribution has the following form:

$$\text{argmax}\, p(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathcal{D}) = (\mathbf{m}_N, \frac{\mathbf{S}_N}{\nu_N + D + 2}) \tag{4.215}$$

If we set $\kappa_0 = 0$, this reduces to

$$\text{argmax}\, p(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathcal{D}) = (\overline{\mathbf{x}}, \frac{\mathbf{S}_0 + \mathbf{S}_{\overline{x}}}{\nu_0 + N + D + 2}) \tag{4.216}$$

The corresponding estimate $\hat{\boldsymbol{\Sigma}}$ is almost the same as Equation 4.183, but differs by 1 in the denominator, because this is the mode of the joint, not the mode of the marginal.

4.6.3.5 Posterior marginals

The posterior marginal for $\boldsymbol{\Sigma}$ is simply

$$p(\boldsymbol{\Sigma}|\mathcal{D}) = \int p(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathcal{D}) d\boldsymbol{\mu} = \text{IW}(\boldsymbol{\Sigma}|\mathbf{S}_N, \nu_N) \tag{4.217}$$

The mode and mean of this marginal are given by

$$\hat{\boldsymbol{\Sigma}}_{map} = \frac{\mathbf{S}_N}{\nu_N + D + 1}, \quad \mathbb{E}[\boldsymbol{\Sigma}] = \frac{\mathbf{S}_N}{\nu_N - D - 1} \tag{4.218}$$

One can show that the posterior marginal for $\boldsymbol{\mu}$ has a multivariate Student T distribution:

$$p(\boldsymbol{\mu}|\mathcal{D}) = \int p(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathcal{D}) d\boldsymbol{\Sigma} = \mathcal{T}(\boldsymbol{\mu}|\mathbf{m}_N, \frac{1}{\kappa_N(\nu_N - D + 1)}\mathbf{S}_N, \nu_N - D + 1) \tag{4.219}$$

This follows from the fact that the Student distribution can be represented as a scaled mixture of Gaussians (see Equation 11.61).

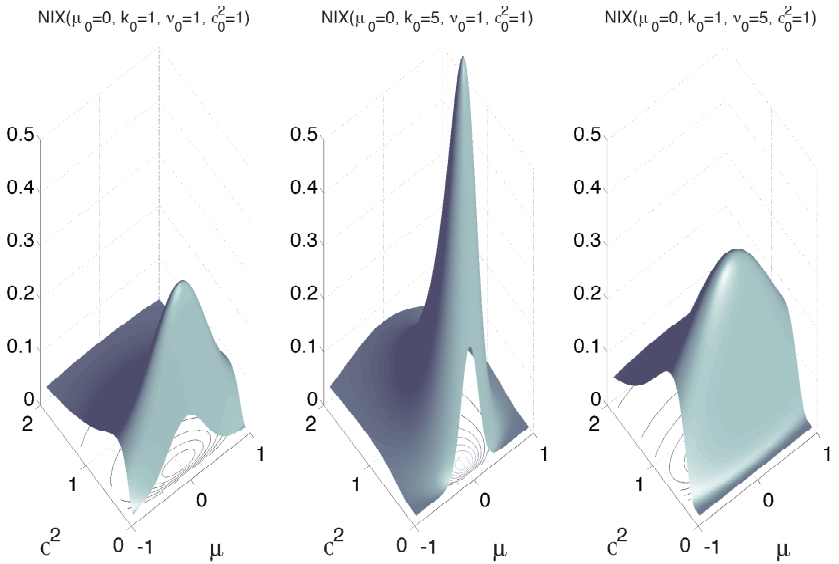


Figure 4.19 The $NI\chi^2(m_0, \kappa_0, \nu_0, \sigma_0^2)$ distribution. m_0 is the prior mean and κ_0 is how strongly we believe this; σ_0^2 is the prior variance and ν_0 is how strongly we believe this. (a) $m_0 = 0, \kappa_0 = 1, \nu_0 = 1, \sigma_0^2 = 1$. Notice that the contour plot (underneath the surface) is shaped like a "squashed egg". (b) We increase the strength of our belief in the mean, so it gets narrower: $m_0 = 0, \kappa_0 = 5, \nu_0 = 1, \sigma_0^2 = 1$. (c) We increase the strength of our belief in the variance, so it gets narrower: $m_0 = 0, \kappa_0 = 1, \nu_0 = 5, \sigma_0^2 = 1$. Figure generated by `NIXdemo2`.

4.6.3.6 Posterior predictive

The posterior predictive is given by

$$p(\mathbf{x}|\mathcal{D}) = \frac{p(\mathbf{x}, \mathcal{D})}{p(\mathcal{D})} \tag{4.220}$$

so it can be easily evaluated in terms of a ratio of marginal likelihoods.

It turns out that this ratio has the form of a multivariate Student-T distribution:

$$p(\mathbf{x}|\mathcal{D}) = \int\int \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) \mathrm{NIW}(\boldsymbol{\mu}, \boldsymbol{\Sigma}|\mathbf{m}_N, \kappa_N, \nu_N, \mathbf{S}_N) d\boldsymbol{\mu} d\boldsymbol{\Sigma} \tag{4.221}$$

$$= \mathcal{T}(\mathbf{x}|\mathbf{m}_N, \frac{\kappa_N + 1}{\kappa_N(\nu_N - D + 1)}\mathbf{S}_N, \nu_N - D + 1) \tag{4.222}$$

The Student-T has wider tails than a Gaussian, which takes into account the fact that $\boldsymbol{\Sigma}$ is unknown. However, this rapidly becomes Gaussian-like.

4.6.3.7 Posterior for scalar data

We now specialise the above results to the case where x_i is 1d. These results are widely used in the statistics literature. As in Section 4.6.2.2, it is conventional not to use the normal inverse

Wishart, but to use the **normal inverse chi-squared** or **NIX** distribution, defined by

$$NI\chi^2(\mu, \sigma^2|m_0, \kappa_0, \nu_0, \sigma_0^2) \triangleq \mathcal{N}(\mu|m_0, \sigma^2/\kappa_0)\ \chi^{-2}(\sigma^2|\nu_0, \sigma_0^2) \tag{4.223}$$

$$\propto (\frac{1}{\sigma^2})^{(\nu_0+3)/2} \exp\left(-\frac{\nu_0\sigma_0^2 + \kappa_0(\mu - m_0)^2}{2\sigma^2}\right) \tag{4.224}$$

See Figure 4.19 for some plots. Along the μ axis, the distribution is shaped like a Gaussian, and along the σ^2 axis, the distribution is shaped like a χ^{-2}; the contours of the joint density have a "squashed egg" appearance. Interestingly, we see that the contours for μ are more peaked for small values of σ^2, which makes sense, since if the data is low variance, we will be able to estimate its mean more reliably.

One can show that the posterior is given by

$$p(\mu, \sigma^2|\mathcal{D}) = NI\chi^2(\mu, \sigma^2|m_N, \kappa_N, \nu_N, \sigma_N^2) \tag{4.225}$$

$$m_N = \frac{\kappa_0 m_0 + N\overline{x}}{\kappa_N} \tag{4.226}$$

$$\kappa_N = \kappa_0 + N \tag{4.227}$$

$$\nu_N = \nu_0 + N \tag{4.228}$$

$$\nu_N\sigma_N^2 = \nu_0\sigma_0^2 + \sum_{i=1}^{N}(x_i - \overline{x})^2 + \frac{N\kappa_0}{\kappa_0 + N}(m_0 - \overline{x})^2 \tag{4.229}$$

The posterior marginal for σ^2 is just

$$p(\sigma^2|\mathcal{D}) = \int p(\mu, \sigma^2|\mathcal{D})d\mu = \chi^{-2}(\sigma^2|\nu_N, \sigma_N^2) \tag{4.230}$$

with the posterior mean given by $\mathbb{E}\left[\sigma^2|\mathcal{D}\right] = \frac{\nu_N}{\nu_N - 2}\sigma_N^2$.

The posterior marginal for μ has a Student T distribution, which follows from the scale mixture representation of the student:

$$p(\mu|\mathcal{D}) = \int p(\mu, \sigma^2|D)d\sigma^2 = \mathcal{T}(\mu|m_N, \sigma_N^2/\kappa_N, \nu_N) \tag{4.231}$$

with the posterior mean given by $\mathbb{E}\left[\mu|\mathcal{D}\right] = m_N$.

Let us see how these results look if we use the following uninformative prior:

$$p(\mu, \sigma^2) \propto p(\mu)p(\sigma^2) \propto \sigma^{-2} \propto NI\chi^2(\mu, \sigma^2|\mu_0 = 0, \kappa_0 = 0, \nu_0 = -1, \sigma_0^2 = 0) \tag{4.232}$$

With this prior, the posterior has the form

$$p(\mu, \sigma^2|\mathcal{D}) = NI\chi^2(\mu, \sigma^2|m_N = \overline{x}, \kappa_N = N, \nu_N = N - 1, \sigma_N^2 = s^2) \tag{4.233}$$

where

$$s^2 \triangleq \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2 = \frac{N}{N-1}\hat{\sigma}_{mle}^2 \tag{4.234}$$

is the the **sample standard deviation**. (In Section 6.4.2, we show that this is an unbiased estimate of the variance.) Hence the marginal posterior for the mean is given by

$$p(\mu|\mathcal{D}) = \mathcal{T}(\mu|\overline{x}, \frac{s^2}{N}, N - 1) \tag{4.235}$$

and the posterior variance of μ is

$$\text{var}\left[\mu|\mathcal{D}\right] = \frac{\nu_N}{\nu_N - 2}\sigma_N^2 = \frac{N-1}{N-3}\frac{s^2}{N} \rightarrow \frac{s^2}{N} \tag{4.236}$$

The square root of this is called the **standard error of the mean**:

$$\sqrt{\text{var}\left[\mu|\mathcal{D}\right]} \approx \frac{s}{\sqrt{N}} \tag{4.237}$$

Thus an approximate 95% posterior **credible interval** for the mean is

$$I_{.95}(\mu|\mathcal{D}) = \overline{x} \pm 2\frac{s}{\sqrt{N}} \tag{4.238}$$

(Bayesian credible intervals are discussed in more detail in Section 5.2.2; they are contrasted with frequentist confidence intervals in Section 6.6.1.)

4.6.3.8 Bayesian t-test

Suppose we want to test the hypothesis that $\mu \neq \mu_0$ for some known value μ_0 (often 0), given values $x_i \sim \mathcal{N}(\mu, \sigma^2)$. This is called a two-sided, one-sample **t-test**. A simple way to perform such a test is just to check if $\mu_0 \in I_{0.95}(\mu|\mathcal{D})$. If it is not, then we can be 95% sure that $\mu \neq \mu_0$.[4] A more common scenario is when we want to test if two paired samples have the same mean. More precisely, suppose $y_i \sim \mathcal{N}(\mu_1, \sigma^2)$ and $z_i \sim \mathcal{N}(\mu_2, \sigma^2)$. We want to determine if $\mu = \mu_1 - \mu_2 > 0$, using $x_i = y_i - z_i$ as our data. We can evaluate this quantity as follows:

$$p(\mu > \mu_0|\mathcal{D}) = \int_{\mu_0}^{\infty} p(\mu|\mathcal{D})d\mu \tag{4.239}$$

This is called a one-sided, **paired t-test**. (For a similar approach to unpaired tests, comparing the difference in binomial proportions, see Section 5.2.3.)

To calculate the posterior, we must specify a prior. Suppose we use an uninformative prior. As we showed above, we find that the posterior marginal on μ has the form

$$p(\mu|\mathcal{D}) = \mathcal{T}(\mu|\overline{x}, \frac{s^2}{N}, N-1) \tag{4.240}$$

Now let us define the following **t statistic**:

$$t \triangleq \frac{\overline{x} - \mu_0}{s/\sqrt{N}} \tag{4.241}$$

where the denominator is the standard error of the mean. We see that

$$p(\mu|\mathcal{D}) = 1 - F_{N-1}(t) \tag{4.242}$$

where $F_\nu(t)$ is the cdf of the standard Student t distribution $\mathcal{T}(0, 1, \nu)$.

4. A more complex approach is to perform Bayesian model comparison. That is, we compute the Bayes factor (described in Section 5.3.3) $p(\mathcal{D}|H_0)/p(\mathcal{D}|H_1)$, where H_0 is the point null hypothesis that $\mu = \mu_0$, and H_1 is the alternative hypothesis that $\mu \neq \mu_0$. See (Gonen et al. 2005; Rouder et al. 2009) for details.

4.6.3.9 Connection with frequentist statistics *

If we use an uninformative prior, it turns out that the above Bayesian analysis gives the same result as derived using frequentist methods. (We discuss frequentist statistics in Chapter 6.) Specifically, from the above results, we see that

$$\frac{\mu - \overline{x}}{\sqrt{s/N}} | \mathcal{D} \sim t_{N-1} \tag{4.243}$$

This has the same form as the sampling distribution of the MLE:

$$\frac{\mu - \overline{X}}{\sqrt{s/N}} | \mu \sim t_{N-1} \tag{4.244}$$

The reason is that the Student distribution is symmetric in its first two arguments, so $\mathcal{T}(\overline{x}|\mu, \sigma^2, \nu) = \mathcal{T}(\mu|\overline{x}, \sigma^2, \nu)$; hence statements about the posterior for μ have the same form as statements about the sampling distribution of $\overline{x}$. Consequently, the (one-sided) p-value (defined in Section 6.6.2) returned by a frequentist test is the same as $p(\mu > \mu_0|\mathcal{D})$ returned by the Bayesian method. See `bayesTtestDemo` for an example.

Despite the superficial similarity, these two results have a different interpretation: in the Bayesian approach, μ is unknown and $\overline{x}$ is fixed, whereas in the frequentist approach, $\overline{X}$ is unknown and μ is fixed. More equivalences between frequentist and Bayesian inference in simple models using uninformative priors can be found in (Box and Tiao 1973). See also Section 7.6.3.3.

4.6.4 Sensor fusion with unknown precisions *

In this section, we apply the results in Section 4.6.3 to the problem of sensor fusion in the case where the precision of each measurement device is unknown. This generalizes the results of Section 4.4.2.2, where the measurement model was assumed to be Gaussian with known precision. The unknown precision case turns out to give qualitatively different results, yielding a potentially multi-modal posterior as we will see. Our presentation is based on (Minka 2001e).

Suppose we want to pool data from multiple sources to estimate some quantity $\mu \in \mathbb{R}$, but the reliability of the sources is unknown. Specifically, suppose we have two different measurement devices, x and y, with different precisions: $x_i|\mu \sim \mathcal{N}(\mu, \lambda_x^{-1})$ and $y_i|\mu \sim \mathcal{N}(\mu, \lambda_y^{-1})$. We make two independent measurements with each device, which turn out to be

$$x_1 = 1.1, x_2 = 1.9, y_1 = 2.9, y_2 = 4.1 \tag{4.245}$$

We will use a non-informative prior for μ, $p(\mu) \propto 1$, which we can emulate using an infinitely broad Gaussian, $p(\mu) = \mathcal{N}(\mu|m_0 = 0, \lambda_0^{-1} = \infty)$. If the λ_x and λ_y terms were known, then the posterior would be Gaussian:

$$p(\mu|\mathcal{D}, \lambda_x, \lambda_y) = \mathcal{N}(\mu|m_N, \lambda_N^{-1}) \tag{4.246}$$

$$\lambda_N = \lambda_0 + N_x\lambda_x + N_y\lambda_y \tag{4.247}$$

$$m_N = \frac{\lambda_x N_x \overline{x} + \lambda_y N_y \overline{y}}{N_x\lambda_x + N_y\lambda_y} \tag{4.248}$$

where $N_x = 2$ is the number of x measurements, $N_y = 2$ is the number of y measurements, $\overline{x} = \frac{1}{N_x}\sum_{i=1}^{N_x} x_i = 1.5$ and $\overline{y} = \frac{1}{N_y}\sum_{i=1}^{N_y} y_i = 3.5$. This result follows because the posterior precision is the sum of the measurement precisions, and the posterior mean is a weighted sum of the prior mean (which is 0) and the data means.

However, the measurement precisions are not known. Initially we will estimate them by maximum likelihood. The log-likelihood is given by

$$\ell(\mu, \lambda_x, \lambda_y) = \log \lambda_x - \frac{\lambda_x}{2}\sum_i (x_i - \mu)^2 + \log \lambda_y - \frac{\lambda_y}{2}\sum_i (y_i - \mu)^2 \tag{4.249}$$

The MLE is obtained by solving the following simultaneous equations:

$$\frac{\partial \ell}{\partial \mu} = \lambda_x N_x(\overline{x} - \mu) + \lambda_y N_y(\overline{y} - \mu) = 0 \tag{4.250}$$

$$\frac{\partial \ell}{\partial \lambda_x} = \frac{1}{\lambda_x} - \frac{1}{N_x}\sum_{i=1}^{N_x}(x_i - \mu)^2 = 0 \tag{4.251}$$

$$\frac{\partial \ell}{\partial \lambda_y} = \frac{1}{\lambda_y} - \frac{1}{N_y}\sum_{i=1}^{N_y}(y_i - \mu)^2 = 0 \tag{4.252}$$

This gives

$$\hat{\mu} = \frac{N_x \hat{\lambda}_x \overline{x} + N_y \hat{\lambda}_y \overline{y}}{N_x \hat{\lambda}_x + N_y \hat{\lambda}_y} \tag{4.253}$$

$$1/\hat{\lambda}_x = \frac{1}{N_x}\sum_i (x_i - \hat{\mu})^2 \tag{4.254}$$

$$1/\hat{\lambda}_y = \frac{1}{N_y}\sum_i (y_i - \hat{\mu})^2 \tag{4.255}$$

We notice that the MLE for μ has the same form as the posterior mean, m_N.

We can solve these equations by fixed point iteration. Let us initialize by estimating $\lambda_x = 1/s_x^2$ and $\lambda_y = 1/s_y^2$, where $s_x^2 = \frac{1}{N_x}\sum_{i=1}^{N_x}(x_i - \overline{x})^2 = 0.16$ and $s_y^2 = \frac{1}{N_y}\sum_{i=1}^{N_y}(y_i - \overline{y})^2 = 0.36$. Using this, we get $\hat{\mu} = 2.1154$, so $p(\mu|\mathcal{D}, \hat{\lambda}_x, \hat{\lambda}_y) = \mathcal{N}(\mu|2.1154, 0.0554)$. If we now iterate, we converge to $\hat{\lambda}_x = 1/0.1662$, $\hat{\lambda}_y = 1/4.0509$, $p(\mu|\mathcal{D}, \hat{\lambda}_x, \hat{\lambda}_y) = \mathcal{N}(\mu|1.5788, 0.0798)$.

The plug-in approximation to the posterior is plotted in Figure 4.20(a). This weights each sensor according to its estimated precision. Since sensor y was estimated to be much less reliable than sensor x, we have $\mathbb{E}\left[\mu|\mathcal{D}, \hat{\lambda}_x, \hat{\lambda}_y\right] \approx \overline{x}$, so we effectively ignore the y sensor.

Now we will adopt a Bayesian approach and integrate out the unknown precisions, rather than trying to estimate them. That is, we compute

$$p(\mu|\mathcal{D}) \propto p(\mu)\left[\int p(\mathcal{D}_x|\mu, \lambda_x)p(\lambda_x|\mu)d\lambda_x\right]\left[\int p(\mathcal{D}_y|\mu, \lambda_y)p(\lambda_y|\mu)d\lambda_y\right] \tag{4.256}$$

We will use uninformative Jeffrey's priors, $p(\mu) \propto 1$, $p(\lambda_x|\mu) \propto 1/\lambda_x$ and $p(\lambda_y|\mu) \propto 1/\lambda_y$.

Since the x and y terms are symmetric, we will just focus on one of them. The key integral is

$$I = \int p(\mathcal{D}_x|\mu, \lambda_x)p(\lambda_x|\mu)d\lambda_x \quad \propto \quad \int \lambda_x^{-1}(N_x\lambda_x)^{N_x/2} \tag{4.257}$$

$$\exp\left(-\frac{N_x}{2}\lambda_x(\overline{x}-\mu)^2 - \frac{N_x}{2}s_x^2\lambda_x\right)d\lambda_x \tag{4.258}$$

Exploiting the fact that $N_x = 2$ this simplifies to

$$I = \int \lambda_x^{-1}\lambda_x^1 \exp(-\lambda_x[(\overline{x}-\mu)^2 + s_x^2])d\lambda_x \tag{4.259}$$

We recognize this as proportional to the integral of an unnormalized Gamma density

$$\mathrm{Ga}(\lambda|a,b) \propto \lambda^{a-1}e^{-\lambda b} \tag{4.260}$$

where $a = 1$ and $b = (\overline{x}-\mu)^2 + s_x^2$. Hence the integral is proportional to the normalizing constant of the Gamma distribution, $\Gamma(a)b^{-a}$, so we get

$$I \propto \int p(\mathcal{D}_x|\mu, \lambda_x)p(\lambda_x|\mu)d\lambda_x \quad \propto \quad \left((\overline{x}-\mu)^2 + s_x^2\right)^{-1} \tag{4.261}$$

and the posterior becomes

$$p(\mu|\mathcal{D}) \propto \frac{1}{(\overline{x}-\mu)^2 + s_x^2} \quad \frac{1}{(\overline{y}-\mu)^2 + s_y^2} \tag{4.262}$$

The exact posterior is plotted in Figure 4.20(b). We see that it has two modes, one near $\overline{x} = 1.5$ and one near $\overline{y} = 3.5$. These correspond to the beliefs that the x sensor is more reliable than the y one, and vice versa. The weight of the first mode is larger, since the data from the x sensor agree more with each other, so it seems slightly more likely that the x sensor is the reliable one. (They obviously cannot both be reliable, since they disagree on the values that they are reporting.) However, the Bayesian solution keeps open the possibility that the y sensor is the more reliable one; from two measurements, we cannot tell, and choosing just the x sensor, as the plug-in approximation does, results in over confidence (a posterior that is too narrow).

Exercises

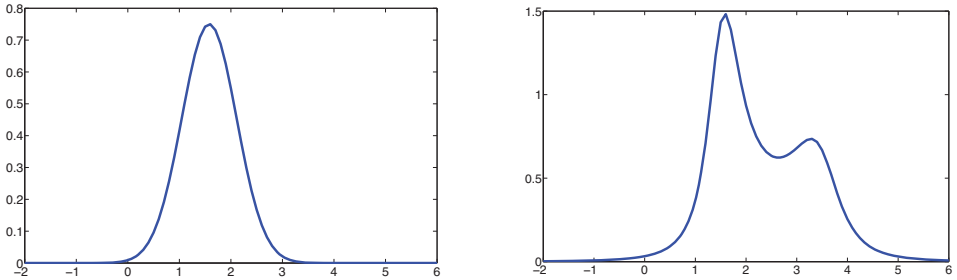
Exercise 4.1 Uncorrelated does not imply independent

Let $X \sim U(-1, 1)$ and $Y = X^2$. Clearly Y is dependent on X (in fact, Y is uniquely determined by X). However, show that $\rho(X, Y) = 0$. Hint: if $X \sim U(a, b)$ then $E[X] = (a + b)/2$ and $\mathrm{var}[X] = (b - a)^2/12$.

Exercise 4.2 Uncorrelated and Gaussian does not imply independent unless *jointly* Gaussian

Let $X \sim \mathcal{N}(0, 1)$ and $Y = WX$, where $p(W = -1) = p(W = 1) = 0.5$. It is clear that X and Y are not independent, since Y is a function of X.

a. Show $Y \sim \mathcal{N}(0, 1)$.



(a) (b)

Figure 4.20 Posterior for μ. (a) Plug-in approximation. (b) Exact posterior. Figure generated by `sensorFusionUnknownPrec`.

b. Show $\text{cov}\,[X, Y] = 0$. Thus X and Y are uncorrelated but dependent, even though they are Gaussian. Hint: use the definition of covariance

$$\text{cov}\,[X, Y] = \mathbb{E}\,[XY] - \mathbb{E}\,[X]\,\mathbb{E}\,[Y] \tag{4.263}$$

and the **rule of iterated expectation**

$$\mathbb{E}\,[XY] = \mathbb{E}\,[\mathbb{E}\,[XY|W]] \tag{4.264}$$

Exercise 4.3 Correlation coefficient is between -1 and +1

Prove that $-1 \leq \rho(X, Y) \leq 1$

Exercise 4.4 Correlation coefficient for linearly related variables is ± 1

Show that, if $Y = aX + b$ for some parameters $a > 0$ and b, then $\rho(X, Y) = 1$. Similarly show that if $a < 0$, then $\rho(X, Y) = -1$.

Exercise 4.5 Normalization constant for a multidimensional Gaussian

Prove that the normalization constant for a d-dimensional Gaussian is given by

$$(2\pi)^{d/2}|\boldsymbol{\Sigma}|^{\frac{1}{2}} = \int \exp(-\frac{1}{2}(\mathbf{x} - \mu)^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}))d\mathbf{x} \tag{4.265}$$

Hint: diagonalize $\boldsymbol{\Sigma}$ and use the fact that $|\boldsymbol{\Sigma}| = \prod_i \lambda_i$ to write the joint pdf as a product of d one-dimensional Gaussians in a transformed coordinate system. (You will need the change of variables formula.) Finally, use the normalization constant for univariate Gaussians.

Exercise 4.6 Bivariate Gaussian

Let $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ where $\mathbf{x} \in \mathbb{R}^2$ and

$$\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & \rho\sigma_1\sigma_2 \\ \rho\sigma_1\sigma_2 & \sigma_2^2 \end{pmatrix} \tag{4.266}$$

where ρ is the correlation coefficient. Show that the pdf is given by

$$p(x_1, x_2) = \frac{1}{2\pi\sigma_1\sigma_2\sqrt{1-\rho^2}} \tag{4.267}$$

$$\exp\left(-\frac{1}{2(1-\rho^2)}\left(\frac{(x_1-\mu_1)^2}{\sigma_1^2} + \frac{(x_2-\mu_2)^2}{\sigma_2^2} - 2\rho\frac{(x_1-\mu_1)}{\sigma_1}\frac{(x_2-\mu_2)}{\sigma_2}\right)\right) \tag{4.268}$$

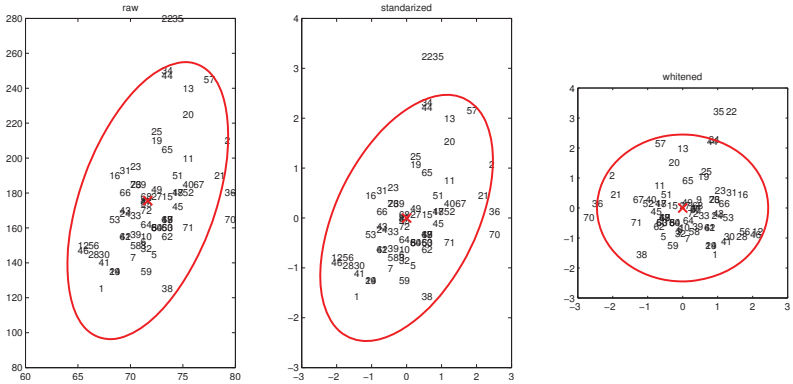


Figure 4.21 (a) Height/weight data for the men. (b) Standardized. (c) Whitened.

Exercise 4.7 Conditioning a bivariate Gaussian

Consider a bivariate Gaussian distribution $p(x_1, x_2) = \mathcal{N}(x|\mu, \Sigma)$ where

$$\Sigma = \begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{21} & \sigma_2^2 \end{pmatrix} = \sigma_1\sigma_2 \begin{pmatrix} \frac{\sigma_1}{\sigma_2} & \rho \\ \rho & \frac{\sigma_2}{\sigma_1} \end{pmatrix} \tag{4.269}$$

where the correlation coefficient is given by

$$\rho \triangleq \frac{\sigma_{12}}{\sigma_1\sigma_2} \tag{4.270}$$

a. What is $P(X_2|x_1)$? Simplify your answer by expressing it in terms of ρ, σ_2, σ_1, μ_1,μ_2 and x_1.

b. Assume $\sigma_1 = \sigma_2 = 1$. What is $P(X_2|x_1)$ now?

Exercise 4.8 Whitening vs standardizing

a. Load the height/weight data using `rawdata = dlmread('heightWeightData.txt')`. The first column is the class label (1=male, 2=female), the second column is height, the third weight. Extract the height/weight data corresponding to the males. Fit a 2d Gaussian to the male data, using the empirical mean and covariance. Plot your Gaussian as an ellipse (use `gaussPlot2d`), superimposing on your scatter plot. It should look like Figure 4.21(a), where have labeled each datapoint by its index. Turn in your figure and code.

b. **Standardizing** the data means ensuring the empirical variance along each dimension is 1. This can be done by computing $\frac{x_{ij}-\overline{x}_j}{\sigma_j}$, where σ_j is the empirical std of dimension j. Standardize the data and replot. It should look like Figure 4.21(b). (Use `axis('equal')`.) Turn in your figure and code.

c. **Whitening** or **sphereing** the data means ensuring its empirical covariance matrix is proportional to $\mathbf{I}$, so the data is uncorrelated and of equal variance along each dimension. This can be done by computing $\mathbf{\Lambda}^{-\frac{1}{2}}\mathbf{U}^T\mathbf{x}$ for each data vector $\mathbf{x}$, where $\mathbf{U}$ are the eigenvectors and $\mathbf{\Lambda}$ the eigenvalues of $\mathbf{X}$. Whiten the data and replot. It should look like Figure 4.21(c). Note that whitening rotates the data, so people move to counter-intuitive locations in the new coordinate system (see e.g., person 2, who moves from the right hand side to the left).

Exercise 4.9 Sensor fusion with known variances in 1d

Suppose we have two sensors with known (and different) variances v_1 and v_2, but unknown (and the same) mean μ. Suppose we observe n_1 observations $y_i^{(1)} \sim \mathcal{N}(\mu, v_1)$ from the first sensor and n_2 observations

$y_i^{(2)} \sim \mathcal{N}(\mu, v_2)$ from the second sensor. (For example, suppose μ is the true temperature outside, and sensor 1 is a precise (low variance) digital thermosensing device, and sensor 2 is an imprecise (high variance) mercury thermometer.) Let $\mathcal{D}$ represent all the data from both sensors. What is the posterior $p(\mu|\mathcal{D})$, assuming a non-informative prior for μ (which we can simulate using a Gaussian with a precision of 0)? Give an explicit expression for the posterior mean and variance.

Exercise 4.10 Derivation of information form formulae for marginalizing and conditioning

Derive the information form results of Section 4.3.1.

Exercise 4.11 Derivation of the NIW posterior

Derive Equation 4.209. Hint: one can show that

$$N(\overline{\mathbf{x}} - \boldsymbol{\mu})(\overline{\mathbf{x}} - \boldsymbol{\mu})^T + \kappa_0(\boldsymbol{\mu} - \mathbf{m}_0)(\boldsymbol{\mu} - \mathbf{m}_0)^T \tag{4.271}$$

$$= \kappa_N(\boldsymbol{\mu} - \mathbf{m}_N)(\boldsymbol{\mu} - \mathbf{m}_N)^T + \frac{\kappa_0 N}{\kappa_N}(\overline{\mathbf{x}} - \mathbf{m}_0)(\overline{\mathbf{x}} - \mathbf{m}_0)^T \tag{4.272}$$

This is a matrix generalization of an operation called **completing the square**.[5]

Derive the corresponding result for the normal-Wishart model.

Exercise 4.12 BIC for Gaussians

(Source: Jaakkola.)

The Bayesian information criterion (BIC) is a penalized log-likelihood function that can be used for model selection (see Section 5.3.2.4). It is defined as

$$BIC = \log p(\mathcal{D}|\hat{\boldsymbol{\theta}}_{ML}) - \frac{d}{2}\log(N) \tag{4.273}$$

where d is the number of free parameters in the model and N is the number of samples. In this question, we will see how to use this to choose between a full covariance Gaussian and a Gaussian with a diagonal covariance. Obviously a full covariance Gaussian has higher likelihood, but it may not be "worth" the extra parameters if the improvement over a diagonal covariance matrix is too small. So we use the BIC score to choose the model.

Following Section 4.1.3, we can write

$$\log p(\mathcal{D}|\hat{\boldsymbol{\Sigma}}, \hat{\boldsymbol{\mu}}) = -\frac{N}{2}\mathrm{tr}\left(\hat{\boldsymbol{\Sigma}}^{-1}\hat{\mathbf{S}}\right) - \frac{N}{2}\log(|\hat{\boldsymbol{\Sigma}}|) \tag{4.274}$$

$$\hat{\mathbf{S}} = \frac{1}{N}\sum_{i=1}^{N}(\mathbf{x}_i - \overline{\mathbf{x}})(\mathbf{x}_i - \overline{\mathbf{x}})^T \tag{4.275}$$

where $\hat{\mathbf{S}}$ is the scatter matrix (empirical covariance), the trace of a matrix is the sum of its diagonals, and we have used the trace trick.

a. Derive the BIC score for a Gaussian in D dimensions with full covariance matrix. Simplify your answer as much as possible, exploiting the form of the MLE. Be sure to specify the number of free parameters d.

b. Derive the BIC score for a Gaussian in D dimensions with a *diagonal* covariance matrix. Be sure to specify the number of free parameters d. Hint: for the digaonal case, the ML estimate of $\boldsymbol{\Sigma}$ is the same as $\hat{\boldsymbol{\Sigma}}_{ML}$ except the off-diagonal terms are zero:

$$\hat{\boldsymbol{\Sigma}}_{diag} = \mathrm{diag}(\hat{\boldsymbol{\Sigma}}_{ML}(1,1), \ldots, \hat{\boldsymbol{\Sigma}}_{ML}(D,D)) \tag{4.276}$$

5. In the scalar case, completing the square means rewriting $c_2x^2 + c_1x + c_0$ as $-a(x-b)^2 + w$ where $a = -c_2$, $b = \frac{c_1}{2c_2}$ and $w = \frac{c_1^2}{4c_2} + c_0$.

Exercise 4.13 Gaussian posterior credible interval

(Source: DeGroot.)

Let $X \sim \mathcal{N}(\mu, \sigma^2 = 4)$ where μ is unknown but has prior $\mu \sim \mathcal{N}(\mu_0, \sigma_0^2 = 9)$. The posterior after seeing n samples is $\mu \sim \mathcal{N}(\mu_n, \sigma_n^2)$. (This is called a credible interval, and is the Bayesian analog of a confidence interval.) How big does n have to be to ensure

$$p(\ell \leq \mu_n \leq u|D) \geq 0.95 \tag{4.277}$$

where (ℓ, u) is an interval (centered on μ_n) of width 1 and D is the data. Hint: recall that 95% of the probability mass of a Gaussian is within $\pm 1.96\sigma$ of the mean.

Exercise 4.14 MAP estimation for 1D Gaussians

(Source: Jaakkola.)

Consider samples $x_1, \ldots, x_n$ from a Gaussian random variable with known variance σ^2 and unknown mean μ. We further assume a prior distribution (also Gaussian) over the mean, $\mu \sim \mathcal{N}(m, s^2)$, with fixed mean m and fixed variance s^2. Thus the only unknown is μ.

a. Calculate the MAP estimate $\hat{\mu}_{MAP}$. You can state the result without proof. Alternatively, with a lot more work, you can compute derivatives of the log posterior, set to zero and solve.
b. Show that as the number of samples n increase, the MAP estimate converges to the maximum likelihood estimate.
c. Suppose n is small and fixed. What does the MAP estimator converge to if we increase the prior variance s^2?
d. Suppose n is small and fixed. What does the MAP estimator converge to if we decrease the prior variance s^2?

Exercise 4.15 Sequential (recursive) updating of $\hat{\mathbf{\Sigma}}$

(Source: (Duda et al. 2001, Q3.35,3.36).)

The unbiased estimates for the covariance of a d-dimensional Gaussian based on n samples is given by

$$\hat{\mathbf{\Sigma}} = \mathbf{C}_n \quad = \quad \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{x}_i - \mathbf{m}_n)(\mathbf{x}_i - \mathbf{m}_n)^T \tag{4.278}$$

It is clear that it takes $O(nd^2)$ time to compute $\mathbf{C}_n$. If the data points arrive one at a time, it is more efficient to incrementally update these estimates than to recompute from scratch.

a. Show that the covariance can be sequentially udpated as follows

$$\mathbf{C}_{n+1} = \frac{n-1}{n} \mathbf{C}_n + \frac{1}{n+1} (\mathbf{x}_{n+1} - \mathbf{m}_n)(\mathbf{x}_{n+1} - \mathbf{m}_n)^T \tag{4.279}$$

b. How much time does it take per sequential update? (Use big-O notation.)
c. Show that we can sequentially update the precision matrix using

$$\mathbf{C}_{n+1}^{-1} = \frac{n}{n-1} \left[\mathbf{C}_n^{-1} - \frac{\mathbf{C}_n^{-1} (\mathbf{x}_{n+1} - \mathbf{m}_n)(\mathbf{x}_{n+1} - \mathbf{m}_n)^T \mathbf{C}_n^{-1}}{\frac{n^2-1}{n} + (\mathbf{x}_{n+1} - \mathbf{m}_n)^T \mathbf{C}_n^{-1} (\mathbf{x}_{n+1} - \mathbf{m}_n)} \right] \tag{4.280}$$

Hint: notice that the update to $\mathbf{C}_{n+1}$ consists of adding a rank-one matrix, namely $\mathbf{u}\mathbf{u}^T$, where $\mathbf{u} = \mathbf{x}_{n+1} - \mathbf{m}_n$. Use the matrix inversion lemma for rank-one updates (Equation 4.111), which we repeat here for convenience:

$$(\mathbf{E} + \mathbf{u}\mathbf{v}^T)^{-1} \quad = \quad \mathbf{E}^{-1} - \frac{\mathbf{E}^{-1}\mathbf{u}\mathbf{v}^T\mathbf{E}^{-1}}{1 + \mathbf{v}^T\mathbf{E}^{-1}\mathbf{u}} \tag{4.281}$$

d. What is the time complexity per update?

Exercise 4.16 Likelihood ratio for Gaussians

Source: Source: Alpaydin p103 ex 4. Consider a binary classifier where the K class conditional densities are MVN $p(x|y=j) = \mathcal{N}(x|\mu_j, \Sigma_j)$. By Bayes rule, we have

$$\log \frac{p(y=1|x)}{p(y=0|x)} = \log \frac{p(x|y=1)}{p(x|y=0)} + \log \frac{p(y=1)}{p(y=0)} \tag{4.282}$$

In other words, the log posterior ratio is the log likelihood ratio plus the log prior ratio. For each of the 4 cases in the table below, derive an expression for the log likelihood ratio $\log \frac{p(x|y=1)}{p(x|y=0)}$, simplifying as much as possible.

Form of Σ_j	Cov	Num parameters
Arbitrary	Σ_j	$Kd(d+1)/2$
Shared	$\Sigma_j = \Sigma$	$d(d+1)/2$
Shared, axis-aligned	$\Sigma_j = \Sigma$ with $\Sigma_{ij} = 0$ for $i \neq j$	d
Shared, spherical	$\Sigma_j = \sigma^2 I$	1

Exercise 4.17 LDA/QDA on height/weight data

The function `discrimAnalysisHeightWeightDemo` fits an LDA and QDA model to the height/weight data. Compute the misclassification rate of both of these models on the training set. Turn in your numbers and code.

Exercise 4.18 Naive Bayes with mixed features

Consider a 3 class naive Bayes classifier with one binary feature and one Gaussian feature:

$$y \sim \text{Mu}(y|\boldsymbol{\pi}, 1),\ x_1|y=c \sim \text{Ber}(x_1|\theta_c),\ x_2|y=c \sim \mathcal{N}(x_2|\mu_c, \sigma_c^2) \tag{4.283}$$

Let the parameter vectors be as follows:

$$\boldsymbol{\pi} = (0.5, 0.25, 0.25),\ \boldsymbol{\theta} = (0.5, 0.5, 0.5),\ \boldsymbol{\mu} = (-1, 0, 1),\ \boldsymbol{\sigma}^2 = (1, 1, 1) \tag{4.284}$$

a. Compute $p(y|x_1 = 0, x_2 = 0)$ (the result should be a vector of 3 numbers that sums to 1).

b. Compute $p(y|x_1 = 0)$.

c. Compute $p(y|x_2 = 0)$.

d. Explain any interesting patterns you see in your results. Hint: look at the parameter vector θ.

Exercise 4.19 Decision boundary for LDA with semi tied covariances

Consider a generative classifier with class conditional densities of the form $\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$. In LDA, we assume $\boldsymbol{\Sigma}_c = \boldsymbol{\Sigma}$, and in QDA, each $\boldsymbol{\Sigma}_c$ is arbitrary. Here we consider the 2 class case in which $\boldsymbol{\Sigma}_1 = k\boldsymbol{\Sigma}_0$, for $k > 1$. That is, the Gaussian ellipsoids have the same "shape", but the one for class 1 is "wider". Derive an expression for $p(y = 1|\mathbf{x}, \boldsymbol{\theta})$, simplifying as much as possible. Give a geometric interpretation of your result, if possible.

Exercise 4.20 Logistic regression vs LDA/QDA

(Source: Jaakkola.) Suppose we train the following binary classifiers via maximum likelihood.

a. GaussI: A generative classifier, where the class conditional densities are Gaussian, with both covariance matrices set to $\mathbf{I}$ (identity matrix), i.e., $p(\mathbf{x}|y=c) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_c, \mathbf{I})$. We assume $p(y)$ is uniform.

b. GaussX: as for GaussI, but the covariance matrices are unconstrained, i.e., $p(\mathbf{x}|y=c) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$.

c. LinLog: A logistic regression model with linear features.

d. QuadLog: A logistic regression model, using linear and quadratic features (i.e., polynomial basis function expansion of degree 2).

After training we compute the performance of each model M on the training set as follows:

$$L(M) = \frac{1}{n}\sum_{i=1}^{n} \log p(y_i|\mathbf{x}_i, \hat{\boldsymbol{\theta}}, M) \tag{4.285}$$

(Note that this is the *conditional* log-likelihood $p(y|\mathbf{x}, \hat{\boldsymbol{\theta}})$ and not the joint log-likelihood $p(y, \mathbf{x}|\hat{\boldsymbol{\theta}})$.) We now want to compare the performance of each model. We will write $L(M) \leq L(M')$ if model M *must* have lower (or equal) log likelihood (on the training set) than M', for any training set (in other words, M is worse than M', at least as far as training set logprob is concerned). For each of the following model pairs, state whether $L(M) \leq L(M')$, $L(M) \geq L(M')$, or whether no such statement can be made (i.e., M might sometimes be better than M' and sometimes worse); also, for each question, briefly (1-2 sentences) explain why.

a. GaussI, LinLog.

b. GaussX, QuadLog.

c. LinLog, QuadLog.

d. GaussI, QuadLog.

e. Now suppose we measure performance in terms of the average misclassification rate on the training set:

$$R(M) = \frac{1}{n}\sum_{i=1}^{n} I(y_i \neq \hat{y}(\mathbf{x}_i)) \tag{4.286}$$

Is it true in general that $L(M) > L(M')$ implies that $R(M) < R(M')$? Explain why or why not.

Exercise 4.21 Gaussian decision boundaries

(Source: (Duda et al. 2001, Q3.7).) Let $p(x|y = j) = \mathcal{N}(x|\mu_j, \sigma_j)$ where $j = 1, 2$ and $\mu_1 = 0, \sigma_1^2 = 1, \mu_2 = 1, \sigma_2^2 = 10^6$. Let the class priors be equal, $p(y = 1) = p(y = 2) = 0.5$.

a. Find the decision region

$$R_1 = \{x : p(x|\mu_1, \sigma_1) \geq p(x|\mu_2, \sigma_2)\} \tag{4.287}$$

Sketch the result. Hint: draw the curves and find where they intersect. Find *both* solutions of the equation

$$p(x|\mu_1, \sigma_1) = p(x|\mu_2, \sigma_2) \tag{4.288}$$

Hint: recall that to solve a quadratic equation $ax^2 + bx + c = 0$, we use

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{4.289}$$

b. Now suppose $\sigma_2 = 1$ (and all other parameters remain the same). What is R_1 in this case?

Exercise 4.22 QDA with 3 classes

Consider a three category classification problem. Let the prior probabilites:

$$P(Y=1) = P(Y=2) = P(Y=3) = 1/3 \tag{4.290}$$

The class-conditional densities are multivariate normal densities with parameters:

$$\mu_1 = [0,0]^T, \mu_2 = [1,1]^T, \mu_3 = [-1,1]^T \tag{4.291}$$

$$\Sigma_1 = \begin{bmatrix} 0.7 & 0 \\ 0 & 0.7 \end{bmatrix}, \Sigma_2 = \begin{bmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix}, \Sigma_3 = \begin{bmatrix} 0.8 & 0.2 \\ 0.2 & 0.8 \end{bmatrix} \tag{4.292}$$

Classify the following points:

a. $\mathbf{x} = [-0.5, 0.5]$
b. $\mathbf{x} = [0.5, 0.5]$

Exercise 4.23 Scalar QDA

[Note: you can solve this exercise by hand or using a computer (matlab, R, whatever). In either case, show your work.] Consider the following training set of heights x (in inches) and gender y (male/female) of some US college students: $\mathbf{x} = (67, 79, 71, 68, 67, 60)$, $\mathbf{y} = (m, m, m, f, f, f)$.

a. Fit a Bayes classifier to this data, using maximum likelihood estimation, i.e., estimate the parameters of the class conditional likelihoods

$$p(x|y=c) = \mathcal{N}(x; \mu_c, \sigma_c) \tag{4.293}$$

and the class prior

$$p(y=c) = \pi_c \tag{4.294}$$

What are your values of μ_c, σ_c, π_c for $c = m, f$? Show your work (so you can get partial credit if you make an arithmetic error).

b. Compute $p(y = m|x, \hat{\theta})$, where $x = 72$, and $\hat{\theta}$ are the MLE parameters. (This is called a plug-in prediction.)

c. What would be a simple way to extend this technique if you had multiple attributes per person, such as height and weight? Write down your proposed model as an equation.

5 *Bayesian statistics*

5.1 Introduction

We have now seen a variety of different probability models, and we have discussed how to fit them to data, i.e., we have discussed how to compute MAP parameter estimates $\hat{\boldsymbol{\theta}} = \operatorname{argmax} p(\boldsymbol{\theta}|\mathcal{D})$, using a variety of different priors. We have also discussed how to compute the full posterior $p(\boldsymbol{\theta}|\mathcal{D})$, as well as the posterior predictive density, $p(\mathbf{x}|\mathcal{D})$, for certain special cases (and in later chapters, we will discuss algorithms for the general case).

Using the posterior distribution to summarize everything we know about a set of unknown variables is at the core of **Bayesian statistics**. In this chapter, we discuss this approach to statistics in more detail. In Chapter 6, we discuss an alternative approach to statistics known as frequentist or classical statistics.

5.2 Summarizing posterior distributions

The posterior $p(\boldsymbol{\theta}|\mathcal{D})$ summarizes everything we know about the unknown quantities $\boldsymbol{\theta}$. In this section, we discuss some simple quantities that can be derived from a probability distribution, such as a posterior. These summary statistics are often easier to understand and visualize than the full joint.

5.2.1 MAP estimation

We can easily compute a **point estimate** of an unknown quantity by computing the posterior mean, median or mode. In Section 5.7, we discuss how to use decision theory to choose between these methods. Typically the posterior mean or median is the most appropriate choice for a real-valued quantity, and the vector of posterior marginals is the best choice for a discrete quantity. However, the posterior mode, aka the MAP estimate, is the most popular choice because it reduces to an optimization problem, for which efficient algorithms often exist. Futhermore, MAP estimation can be interpreted in non-Bayesian terms, by thinking of the log prior as a regularizer (see Section 6.5 for more details).

Although this approach is computationally appealing, it is important to point out that there are various drawbacks to MAP estimation, which we briefly discuss below. This will provide motivation for the more thoroughly Bayesian approach which we will study later in this chapter (and elsewhere in this book).

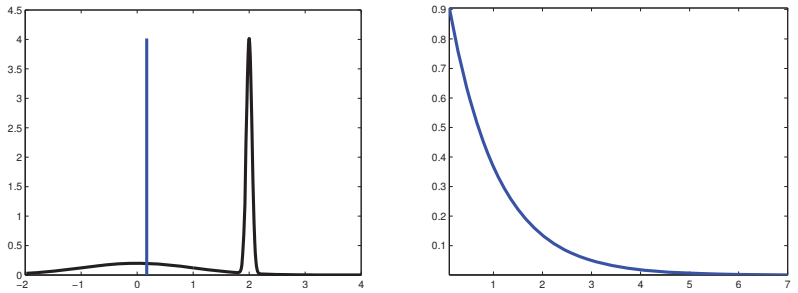


Figure 5.1 (a) A bimodal distribution in which the mode is very untypical of the distribution. The thin blue vertical line is the mean, which is arguably a better summary of the distribution, since it is near the majority of the probability mass. Figure generated by `bimodalDemo`. (b) A skewed distribution in which the mode is quite different from the mean. Figure generated by `gammaPlotDemo`.

5.2.1.1 No measure of uncertainty

The most obvious drawback of MAP estimation, and indeed of any other **point estimate** such as the posterior mean or median, is that it does not provide any measure of uncertainty. In many applications, it is important to know how much one can trust a given estimate. We can derive such confidence measures from the posterior, as we discuss in Section 5.2.2.

5.2.1.2 Plugging in the MAP estimate can result in overfitting

In machine learning, we often care more about predictive accuracy than in interpreting the parameters of our models. However, if we don't model the uncertainty in our parameters, then our predictive distribution will be overconfident. We saw several examples of this in Chapter 3, and we will see more examples later. Overconfidence in predictions is particularly problematic in situations where we may be risk averse; see Section 5.7 for details.

5.2.1.3 The mode is an untypical point

Choosing the mode as a summary of a posterior distribution is often a very poor choice, since the mode is usually quite untypical of the distribution, unlike the mean or median. This is illustrated in Figure 5.1(a) for a 1d continuous space. The basic problem is that the mode is a point of measure zero, whereas the mean and median take the volume of the space into account. Another example is shown in Figure 5.1(b): here the mode is 0, but the mean is non-zero. Such skewed distributions often arise when inferring variance parameters, especially in hierarchical models. In such cases the MAP estimate (and hence the MLE) is obviously a very bad estimate.

How should we summarize a posterior if the mode is not a good choice? The answer is to use decision theory, which we discuss in Section 5.7. The basic idea is to specify a loss function, where $L(\theta, \hat{\theta})$ is the loss you incur if the truth is θ and your estimate is $\hat{\theta}$. If we use 0-1 loss, $L(\theta, \hat{\theta}) = \mathbb{I}(\theta \neq \hat{\theta})$, then the optimal estimate is the posterior mode. 0-1 loss means you only get "points" if you make no errors, otherwise you get nothing: there is no "partial credit" under

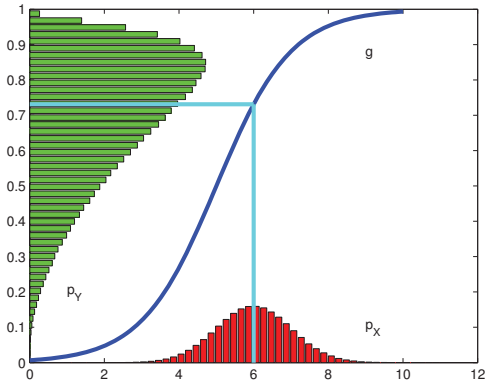


Figure 5.2 Example of the transformation of a density under a nonlinear transform. Note how the mode of the transformed distribution is not the transform of the original mode. Based on Exercise 1.4 of (Bishop 2006b). Figure generated by `bayesChangeOfVar`.

this loss function! For continuous-valued quantities, we often prefer to use squared error loss, $L(\theta, \hat{\theta}) = (\theta - \hat{\theta})^2$; the corresponding optimal estimator is then the posterior mean, as we show in Section 5.7. Or we can use a more robust loss function, $L(\theta, \hat{\theta}) = |\theta - \hat{\theta}|$, which gives rise to the posterior median.

5.2.1.4 MAP estimation is not invariant to reparameterization *

A more subtle problem with MAP estimation is that the result we get depends on how we parameterize the probability distribution. Changing from one representation to another equivalent representation changes the result, which is not very desirable, since the units of measurement are arbitrary (e.g., when measuring distance, we can use centimetres or inches).

To understand the problem, suppose we compute the posterior for x. If we define $y = f(x)$, the distribution for y is given by Equation 2.87, which we repeat here for convenience:

$$p_y(y) = p_x(x)\left|\frac{dx}{dy}\right| \tag{5.1}$$

The $|\frac{dx}{dy}|$ term is called the Jacobian, and it measures the change in size of a unit volume passed through f. Let $\hat{x} = \text{argmax}_x\, p_x(x)$ be the MAP estimate for x. In general it is not the case that $\hat{y} = \text{argmax}_y\, p_y(y)$ is given by $f(\hat{x})$. For example, let $x \sim \mathcal{N}(6, 1)$ and $y = f(x)$, where

$$f(x) = \frac{1}{1 + \exp(-x + 5)} \tag{5.2}$$

We can derive the distribution of y using Monte Carlo simulation (see Section 2.7.1). The result is shown in Figure 5.2. We see that the original Gaussian has become "squashed" by the sigmoid nonlinearity. In particular, we see that the mode of the transformed distribution is not equal to the transform of the original mode.

To see how this problem arises in the context of MAP estimation, consider the following example, due to Michael Jordan. The Bernoulli distribution is typically parameterized by its mean μ, so $p(y = 1|\mu) = \mu$, where $y \in \{0, 1\}$. Suppose we have a uniform prior on the unit interval: $p_\mu(\mu) = 1\ \mathbb{I}(0 \leq \mu \leq 1)$. If there is no data, the MAP estimate is just the mode of the prior, which can be anywhere between 0 and 1. We will now show that different parameterizations can pick different points in this interval arbitrarily.

First let $\theta = \sqrt{\mu}$ so $\mu = \theta^2$. The new prior is

$$p_\theta(\theta) = p_\mu(\mu)\left|\frac{d\mu}{d\theta}\right| = 2\theta \tag{5.3}$$

for $\theta \in [0, 1]$ so the new mode is

$$\hat{\theta}_{MAP} = \arg\max_{\theta \in [0,1]} 2\theta = 1 \tag{5.4}$$

Now let $\phi = 1 - \sqrt{1 - \mu}$. The new prior is

$$p_\phi(\phi) = p_\mu(\mu)\left|\frac{d\mu}{d\phi}\right| = 2(1 - \phi) \tag{5.5}$$

for $\phi \in [0, 1]$, so the new mode is

$$\hat{\phi}_{MAP} = \arg\max_{\phi \in [0,1]} 2 - 2\phi = 0 \tag{5.6}$$

Thus the MAP estimate depends on the parameterization. The MLE does not suffer from this since the likelihood is a function, not a probability density. Bayesian inference does not suffer from this problem either, since the change of measure is taken into account when integrating over the parameter space.

One solution to the problem is to optimize the following objective function:

$$\hat{\boldsymbol{\theta}} = \operatorname*{argmax}_{\boldsymbol{\theta}} p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})|\mathbf{I}(\boldsymbol{\theta})|^{-\frac{1}{2}} \tag{5.7}$$

Here $\mathbf{I}(\boldsymbol{\theta})$ is the Fisher information matrix associated with $p(\mathbf{x}|\boldsymbol{\theta})$ (see Section 6.2.2). This estimate is parameterization independent, for reasons explained in (Jermyn 2005; Druilhet and Marin 2007). Unfortunately, optimizing Equation 5.7 is often difficult, which minimizes the appeal of the whole approach.

5.2.2 Credible intervals

In addition to point estimates, we often want a measure of confidence. A standard measure of confidence in some (scalar) quantity θ is the "width" of its posterior distribution. This can be measured using a $100(1 - \alpha)$% **credible interval**, which is a (contiguous) region $C = (\ell, u)$ (standing for lower and upper) which contains $1 - \alpha$ of the posterior probability mass, i.e.,

$$C_\alpha(\mathcal{D}) = (\ell, u) : P(\ell \leq \theta \leq u|\mathcal{D}) = 1 - \alpha \tag{5.8}$$

There may be many such intervals, so we choose one such that there is $(1 - \alpha)/2$ mass in each tail; this is called a **central interval**.

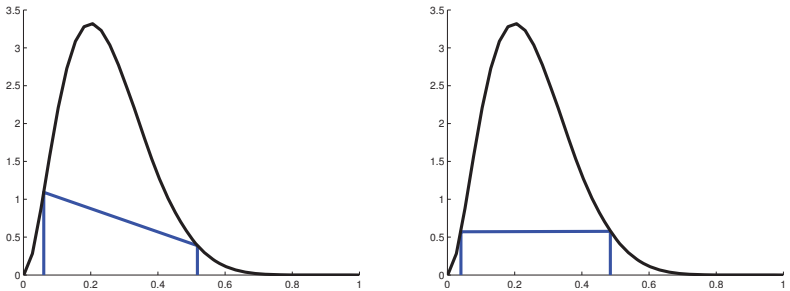


Figure 5.3 (a) Central interval and (b) HPD region for a Beta(3,9) posterior. The CI is (0.06, 0.52) and the HPD is (0.04, 0.48). Based on Figure 3.6 of (Hoff 2009). Figure generated by `betaHPD`.

If the posterior has a known functional form, we can compute the posterior central interval using $\ell = F^{-1}(\alpha/2)$ and $u = F^{-1}(1-\alpha/2)$, where F is the cdf of the posterior. For example, if the posterior is Gaussian, $p(\theta|\mathcal{D}) = \mathcal{N}(0,1)$, and $\alpha = 0.05$, then we have $\ell = \Phi(\alpha/2) = -1.96$, and $u = \Phi(1-\alpha/2) = 1.96$, where Φ denotes the cdf of the Gaussian. This is illustrated in Figure 2.3(c). This justifies the common practice of quoting a credible interval in the form of $\mu \pm 2\sigma$, where μ represents the posterior mean, σ represents the posterior standard deviation, and 2 is a good approximation to 1.96.

Of course, the posterior is not always Gaussian. For example, in our coin example, if we use a uniform prior and we observe $N_1 = 47$ heads out of $N = 100$ trials, then the posterior is a beta distribution, $p(\theta|\mathcal{D}) = \text{Beta}(48, 54)$. We find the 95% posterior credible interval is $(0.3749, 0.5673)$ (see `betaCredibleInt` for the one line of Matlab code we used to compute this).

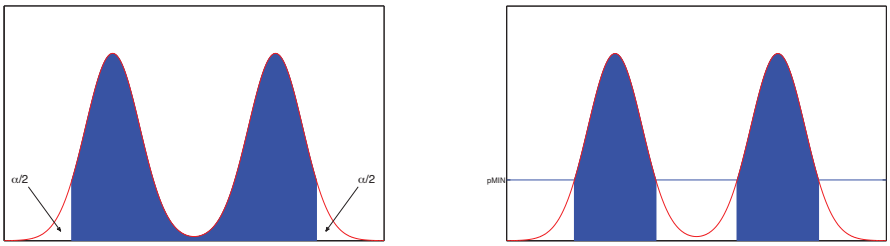
If we don't know the functional form, but we can draw samples from the posterior, then we can use a Monte Carlo approximation to the posterior quantiles: we simply sort the S samples, and find the one that occurs at location α/S along the sorted list. As $S \to \infty$, this converges to the true quantile. See `mcQuantileDemo` for a demo.

People often confuse Bayesian credible intervals with frequentist confidence intervals. However, they are not the same thing, as we discuss in Section 6.6.1. In general, credible intervals are usually what people want to compute, but confidence intervals are usually what they actually compute, because most people are taught frequentist statistics but not Bayesian statistics. Fortunately, the mechanics of computing a credible interval is just as easy as computing a confidence interval (see e.g., `betaCredibleInt` for how to do it in Matlab).

5.2.2.1 Highest posterior density regions *

A problem with central intervals is that there might be points outside the CI which have higher probability density. This is illustrated in Figure 5.3(a), where we see that points outside the left-most CI boundary have higher density than those just inside the right-most CI boundary.

This motivates an alternative quantity known as the **highest posterior density** or **HPD** region. This is defined as the (set of) most probable points that in total constitute $100(1-\alpha)\%$ of the



(a) (b)

Figure 5.4 (a) Central interval and (b) HPD region for a hypothetical multimodal posterior. Based on Figure 2.2 of (Gelman et al. 2004). Figure generated by `postDensityIntervals`.

probability mass. More formally, we find the threshold p^* on the pdf such that

$$1 - \alpha = \int_{\theta: p(\theta|\mathcal{D}) > p^*} p(\theta|\mathcal{D}) d\theta \tag{5.9}$$

and then define the HPD as

$$C_\alpha(\mathcal{D}) = \{\theta : p(\theta|\mathcal{D}) \geq p^*\} \tag{5.10}$$

In 1d, the HPD region is sometimes called a **highest density interval** or **HDI**. For example, Figure 5.3(b) shows the 95% HDI of a $\text{Beta}(3, 9)$ distribution, which is $(0.04, 0.48)$. We see that this is narrower than the CI, even though it still contains 95% of the mass; furthermore, every point inside of it has higher density than every point outside of it.

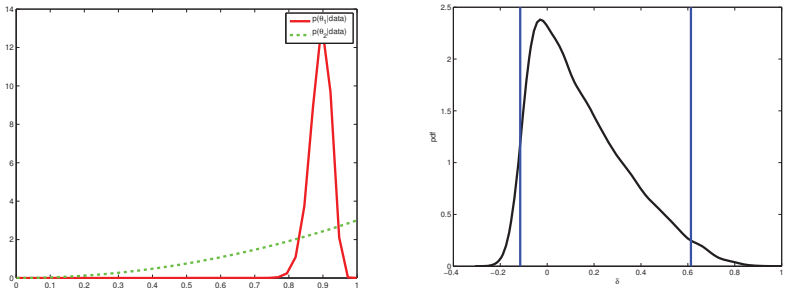
For a unimodal distribution, the HDI will be the narrowest interval around the mode containing 95% of the mass. To see this, imagine "water filling" in reverse, where we lower the level until 95% of the mass is revealed, and only 5% is submerged. This gives a simple algorithm for computing HDIs in the 1d case: simply search over points such that the interval contains 95% of the mass and has minimal width. This can be done by 1d numerical optimization if we know the inverse CDF of the distribution, or by search over the sorted data points if we have a bag of samples (see `betaHPD` for a demo).

If the posterior is multimodal, the HDI may not even be a connected region: see Figure 5.4(b) for an example. However, summarizing multimodal posteriors is always difficult.

5.2.3 Inference for a difference in proportions

Sometimes we have multiple parameters, and we are interested in computing the posterior distribution of some function of these parameters. For example, suppose you are about to buy something from Amazon.com, and there are two sellers offering it for the same price. Seller 1 has 90 positive reviews and 10 negative reviews. Seller 2 has 2 positive reviews and 0 negative reviews. Who should you buy from?[1]

1. This example is from `www.johndcook.com/blog/2011/09/27/bayesian-amazon`. See also `lingpipe-blog.com/2009/10/13/bayesian-counterpart-to-fisher-exact-test-on-contingency-tables`.



(a) (b)

Figure 5.5 (a) Exact posteriors $p(\theta_i|\mathcal{D}_i)$. (b) Monte Carlo approximation to $p(\delta|\mathcal{D})$. We use kernel density estimation to get a smooth plot. The vertical lines enclose the 95% central interval. Figure generated by `amazonSellerDemo`,

On the face of it, you should pick seller 2, but we cannot be very confident that seller 2 is better since it has had so few reviews. In this section, we sketch a Bayesian analysis of this problem. Similar methodology can be used to compare rates or proportions across groups for a variety of other settings.

Let θ_1 and θ_2 be the unknown reliabilities of the two sellers. Since we don't know much about them, we'll endow them both with uniform priors, $\theta_i \sim \text{Beta}(1,1)$. The posteriors are $p(\theta_1|\mathcal{D}_1) = \text{Beta}(91,11)$ and $p(\theta_2|\mathcal{D}_2) = \text{Beta}(3,1)$.

We want to compute $p(\theta_1 > \theta_2|\mathcal{D})$. For convenience, let us define $\delta = \theta_1 - \theta_2$ as the difference in the rates. (Alternatively we might want to work in terms of the log-odds ratio.) We can compute the desired quantity using numerical integration:

$$p(\delta > 0|\mathcal{D}) = \int_0^1 \int_0^1 \mathbb{I}(\theta_1 > \theta_2)\text{Beta}(\theta_1|y_1+1, N_1-y_1+1) \text{Beta}(\theta_2|y_2+1, N_2-y_2+1)d\theta_1 d\theta_2 \quad (5.11)$$

We find $p(\delta > 0|\mathcal{D}) = 0.710$, which means you are better off buying from seller 1! See `amazonSellerDemo` for the code. (It is also possible to solve the integral analytically (Cook 2005).)

A simpler way to solve the problem is to approximate the posterior $p(\delta|\mathcal{D})$ by Monte Carlo sampling. This is easy, since θ_1 and θ_2 are independent in the posterior, and both have beta distributions, which can be sampled from using standard methods. The distributions $p(\theta_i|\mathcal{D}_i)$ are shown in Figure 5.5(a), and a MC approximation to $p(\delta|\mathcal{D})$, together with a 95% HPD, is shown Figure 5.5(b). An MC approximation to $p(\delta > 0|\mathcal{D})$ is obtained by counting the fraction of samples where $\theta_1 > \theta_2$; this turns out to be 0.718, which is very close to the exact value. (See `amazonSellerDemo` for the code.)

5.3 Bayesian model selection

In Figure 1.18, we saw that using too high a degree polynomial results in overfitting, and using too low a degree results in underfitting. Similarly, in Figure 7.8(a), we saw that using too small

a regularization parameter results in overfitting, and too large a value results in underfitting. In general, when faced with a set of models (i.e., families of parametric distributions) of different complexity, how should we choose the best one? This is called the **model selection** problem.

One approach is to use cross-validation to estimate the generalization error of all the candiate models, and then to pick the model that seems the best. However, this requires fitting each model K times, where K is the number of CV folds. A more efficient approach is to compute the posterior over models,

$$p(m|\mathcal{D}) = \frac{p(\mathcal{D}|m)p(m)}{\sum_{m \in \mathcal{M}} p(m, \mathcal{D})} \tag{5.12}$$

From this, we can easily compute the MAP model, $\hat{m} = \operatorname{argmax} p(m|\mathcal{D})$. This is called **Bayesian model selection**.

If we use a uniform prior over models, $p(m) \propto 1$, this amounts to picking the model which maximizes

$$p(\mathcal{D}|m) = \int p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}|m)d\boldsymbol{\theta} \tag{5.13}$$

This quantity is called the **marginal likelihood**, the **integrated likelihood**, or the **evidence** for model m. The details on how to perform this integral will be discussed in Section 5.3.2. But first we give an intuitive interpretation of what this quantity means.

5.3.1 Bayesian Occam's razor

One might think that using $p(\mathcal{D}|m)$ to select models would always favor the model with the most parameters. This is true if we use $p(\mathcal{D}|\hat{\boldsymbol{\theta}}_m)$ to select models, where $\hat{\boldsymbol{\theta}}_m$ is the MLE or MAP estimate of the parameters for model m, because models with more parameters will fit the data better, and hence achieve higher likelihood. However, if we integrate out the parameters, rather than maximizing them, we are automatically protected from overfitting: models with more parameters do not necessarily have higher *marginal* likelihood. This is called the **Bayesian Occam's razor** effect (MacKay 1995b; Murray and Ghahramani 2005), named after the principle known as **Occam's razor**, which says one should pick the simplest model that adequately explains the data.

One way to understand the Bayesian Occam's razor is to notice that the marginal likelihood can be rewritten as follows, based on the chain rule of probability (Equation 2.5):

$$p(\mathcal{D}) = p(y_1)p(y_2|y_1)p(y_3|y_{1:2}) \ldots p(y_N|y_{1:N-1}) \tag{5.14}$$

where we have dropped the conditioning on $\mathbf{x}$ for brevity. This is similar to a leave-one-out cross-validation estimate (Section 1.4.8) of the likelihood, since we predict each future point given all the previous ones. (Of course, the order of the data does not matter in the above expression.) If a model is too complex, it will overfit the "early" examples and will then predict the remaining ones poorly.

Another way to understand the Bayesian Occam's razor effect is to note that probabilities must sum to one. Hence $\sum_{\mathcal{D}'} p(\mathcal{D}'|m) = 1$, where the sum is over all possible data sets. Complex models, which can predict many things, must spread their probability mass thinly, and hence will not obtain as large a probability for any given data set as simpler models. This is sometimes

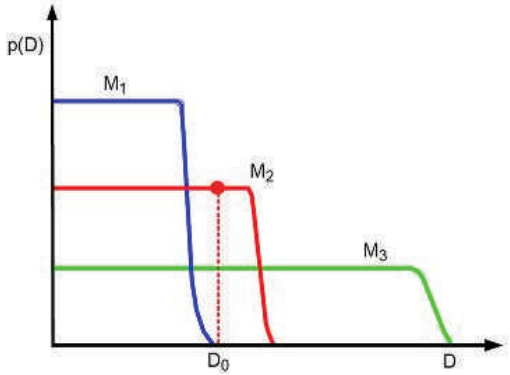


Figure 5.6 A schematic illustration of the Bayesian Occam's razor. The broad (green) curve corresponds to a complex model, the narrow (blue) curve to a simple model, and the middle (red) curve is just right. Based on Figure 3.13 of (Bishop 2006a). See also (Murray and Ghahramani 2005, Figure 2) for a similar plot produced on real data.

called the **conservation of probability mass** principle, and is illustrated in Figure 5.6. On the horizontal axis we plot all possible data sets in order of increasing complexity (measured in some abstract sense). On the vertical axis we plot the predictions of 3 possible models: a simple one, M_1; a medium one, M_2; and a complex one, M_3. We also indicate the actually observed data $\mathcal{D}_0$ by a vertical line. Model 1 is too simple and assigns low probability to $\mathcal{D}_0$. Model 3 also assigns $\mathcal{D}_0$ relatively low probability, because it can predict many data sets, and hence it spreads its probability quite widely and thinly. Model 2 is "just right": it predicts the observed data with a reasonable degree of confidence, but does not predict too many other things. Hence model 2 is the most probable model.

As a concrete example of the Bayesian Occam's razor, consider the data in Figure 5.7. We plot polynomials of degrees 1, 2 and 3 fit to $N = 5$ data points. It also shows the posterior over models, where we use a Gaussian prior (see Section 7.6 for details). There is not enough data to justify a complex model, so the MAP model is $d = 1$. Figure 5.8 shows what happens when $N = 30$. Now it is clear that $d = 2$ is the right model (the data was in fact generated from a quadratic).

As another example, Figure 7.8(c) plots $\log p(\mathcal{D}|\lambda)$ vs $\log(\lambda)$, for the polynomial ridge regression model, where λ ranges over the same set of values used in the CV experiment. We see that the maximum evidence occurs at roughly the same point as the minimum of the test MSE, which also corresponds to the point chosen by CV.

When using the Bayesian approach, we are not restricted to evaluating the evidence at a finite grid of values. Instead, we can use numerical optimization to find $\lambda^* = \mathrm{argmax}_\lambda \, p(\mathcal{D}|\lambda)$. This technique is called **empirical Bayes** or **type II maximum likelihood** (see Section 5.6 for details). An example is shown in Figure 7.8(b): we see that the curve has a similar shape to the CV estimate, but it can be computed more efficiently.

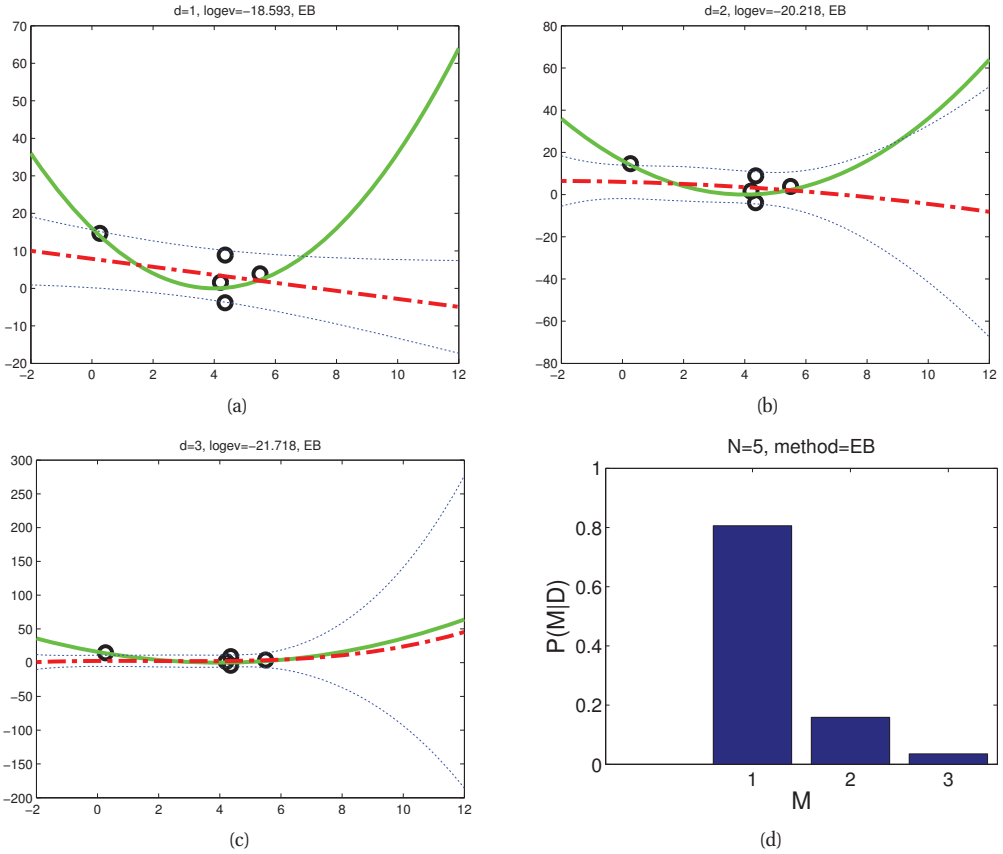


Figure 5.7 (a-c) We plot polynomials of degrees 1, 2 and 3 fit to $N = 5$ data points using empirical Bayes. The solid green curve is the true function, the dashed red curve is the prediction (dotted blue lines represent $\pm\sigma$ around the mean). (d) We plot the posterior over models, $p(d|\mathcal{D})$, assuming a uniform prior $p(d) \propto 1$. Based on a figure by Zoubin Ghahramani. Figure generated by `linregEbModelSelVsN`.

5.3.2 Computing the marginal likelihood (evidence)

When discussing parameter inference for a fixed model, we often wrote

$$p(\boldsymbol{\theta}|\mathcal{D}, m) \propto p(\boldsymbol{\theta}|m)p(\mathcal{D}|\boldsymbol{\theta}, m) \tag{5.15}$$

thus ignoring the normalization constant $p(\mathcal{D}|m)$. This is valid since $p(\mathcal{D}|m)$ is constant wrt $\boldsymbol{\theta}$. However, when comparing models, we need to know how to compute the marginal likelihood, $p(\mathcal{D}|m)$. In general, this can be quite hard, since we have to integrate over all possible parameter values, but when we have a conjugate prior, it is easy to compute, as we now show.

Let $p(\boldsymbol{\theta}) = q(\boldsymbol{\theta})/Z_0$ be our prior, where $q(\boldsymbol{\theta})$ is an unnormalized distribution, and Z_0 is the normalization constant of the prior. Let $p(\mathcal{D}|\boldsymbol{\theta}) = q(\mathcal{D}|\boldsymbol{\theta})/Z_\ell$ be the likelihood, where Z_ℓ contains any constant factors in the likelihood. Finally let $p(\boldsymbol{\theta}|\mathcal{D}) = q(\boldsymbol{\theta}|\mathcal{D})/Z_N$ be our poste-

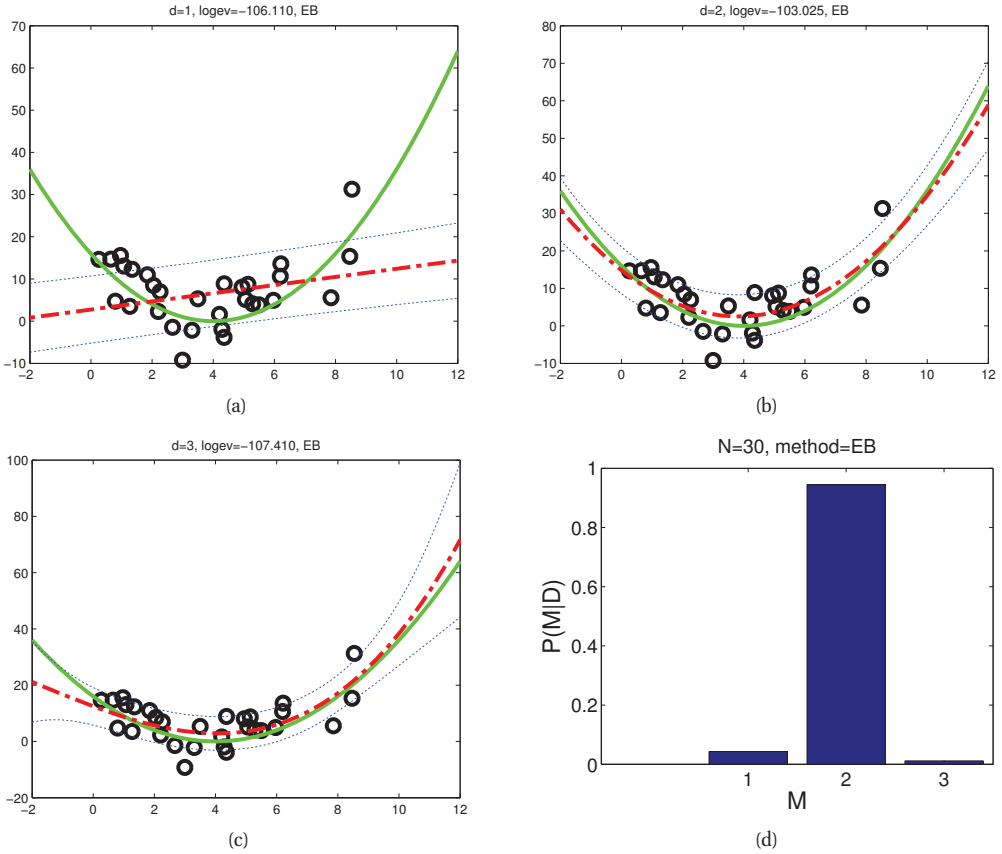


Figure 5.8 Same as Figure 5.7 except now $N = 30$. Figure generated by `linregEbModelSelVsN`.

rior, where $q(\boldsymbol{\theta}|\mathcal{D}) = q(\mathcal{D}|\boldsymbol{\theta})q(\boldsymbol{\theta})$ is the unnormalized posterior, and Z_N is the normalization constant of the posterior. We have

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta})}{p(\mathcal{D})} \tag{5.16}$$

$$\frac{q(\boldsymbol{\theta}|\mathcal{D})}{Z_N} = \frac{q(\mathcal{D}|\boldsymbol{\theta})q(\boldsymbol{\theta})}{Z_\ell Z_0 p(\mathcal{D})} \tag{5.17}$$

$$p(\mathcal{D}) = \frac{Z_N}{Z_0 Z_\ell} \tag{5.18}$$

So assuming the relevant normalization constants are tractable, we have an easy way to compute the marginal likelihood. We give some examples below.

5.3.2.1 Beta-binomial model

Let us apply the above result to the Beta-binomial model. Since we know $p(\theta|\mathcal{D}) = \text{Beta}(\theta|a', b')$, where $a' = a + N_1$ and $b' = b + N_0$, we know the normalization constant of the posterior is $B(a', b')$. Hence

$$p(\theta|\mathcal{D}) = \frac{p(\mathcal{D}|\theta)p(\theta)}{p(\mathcal{D})} \tag{5.19}$$

$$= \frac{1}{p(\mathcal{D})}\left[\frac{1}{B(a,b)}\theta^{a-1}(1-\theta)^{b-1}\right]\left[\binom{N}{N_1}\theta^{N_1}(1-\theta)^{N_0}\right] \tag{5.20}$$

$$= \binom{N}{N_1}\frac{1}{p(\mathcal{D})}\frac{1}{B(a,b)}\left[\theta^{a+N_1-1}(1-\theta)^{b+N_0-1}\right] \tag{5.21}$$

So

$$\frac{1}{B(a+N_1, b+N_0)} = \binom{N}{N_1}\frac{1}{p(\mathcal{D})}\frac{1}{B(a,b)} \tag{5.22}$$

$$p(\mathcal{D}) = \binom{N}{N_1}\frac{B(a+N_1, b+N_0)}{B(a,b)} \tag{5.23}$$

The marginal likelihood for the Beta-Bernoulli model is the same as above, except it is missing the $\binom{N}{N_1}$ term.

5.3.2.2 Dirichlet-multinoulli model

By the same reasoning as the Beta-Bernoulli case, one can show that the marginal likelihood for the Dirichlet-multinoulli model is given by

$$p(\mathcal{D}) = \frac{B(\mathbf{N}+\boldsymbol{\alpha})}{B(\boldsymbol{\alpha})} \tag{5.24}$$

where

$$B(\boldsymbol{\alpha}) = \frac{\prod_{k=1}^{K}\Gamma(\alpha_k)}{\Gamma(\sum_k \alpha_k)} \tag{5.25}$$

Hence we can rewrite the above result in the following form, which is what is usually presented in the literature:

$$p(\mathcal{D}) = \frac{\Gamma(\sum_k \alpha_k)}{\Gamma(N + \sum_k \alpha_k)}\prod_k \frac{\Gamma(N_k + \alpha_k)}{\Gamma(\alpha_k)} \tag{5.26}$$

We will see many applications of this equation later.

5.3.2.3 Gaussian-Gaussian-Wishart model

Consider the case of an MVN with a conjugate NIW prior. Let Z_0 be the normalizer for the prior, Z_N be normalizer for the posterior, and let $Z_l = (2\pi)^{ND/2}$ be the normalizer for the

likelihood. Then it is easy to see that

$$p(\mathcal{D}) = \frac{Z_N}{Z_0 Z_l} \tag{5.27}$$

$$= \frac{1}{\pi^{ND/2}} \frac{1}{2^{ND/2}} \frac{\left(\frac{2\pi}{\kappa_N}\right)^{D/2} |\mathbf{S}_N|^{-\nu_N/2} 2^{(\nu_0+N)D/2} \Gamma_D(\nu_N/2)}{\left(\frac{2\pi}{\kappa_0}\right)^{D/2} |\mathbf{S}_0|^{-\nu_0/2} 2^{\nu_0 D/2} \Gamma_D(\nu_0/2)} \tag{5.28}$$

$$= \frac{1}{\pi^{ND/2}} \left(\frac{\kappa_0}{\kappa_N}\right)^{D/2} \frac{|\mathbf{S}_0|^{\nu_0/2}}{|\mathbf{S}_N|^{\nu_N/2}} \frac{\Gamma_D(\nu_N/2)}{\Gamma_D(\nu_0/2)} \tag{5.29}$$

This equation will prove useful later.

5.3.2.4 BIC approximation to log marginal likelihood

In general, computing the integral in Equation 5.13 can be quite difficult. One simple but popular approximation is known as the **Bayesian information criterion** or **BIC**, which has the following form (Schwarz 1978):

$$\text{BIC} \triangleq \log p(\mathcal{D}|\hat{\boldsymbol{\theta}}) - \frac{\text{dof}(\hat{\boldsymbol{\theta}})}{2} \log N \approx \log p(\mathcal{D}) \tag{5.30}$$

where $\text{dof}(\hat{\boldsymbol{\theta}})$ is the number of **degrees of freedom** in the model, and $\hat{\boldsymbol{\theta}}$ is the MLE for the model.[2] We see that this has the form of a **penalized log likelihood**, where the penalty term depends on the model's complexity. See Section 8.4.2 for the derivation of the BIC score.

As an example, consider linear regression. As we show in Section 7.3, the MLE is given by $\hat{\mathbf{w}} = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\mathbf{y}$ and $\hat{\sigma}^2 = \text{RSS}/N$, where $\text{RSS} = \sum_{i=1}^N (y_i - \hat{\mathbf{w}}_{mle}^T \mathbf{x}_i)^2$. The corresponding log likelihood is given by

$$\log p(\mathcal{D}|\hat{\boldsymbol{\theta}}) = -\frac{N}{2} \log(2\pi\hat{\sigma}^2) - \frac{N}{2} \tag{5.31}$$

Hence the BIC score is as follows (dropping constant terms)

$$\text{BIC} = -\frac{N}{2} \log(\hat{\sigma}^2) - \frac{D}{2} \log(N) \tag{5.32}$$

where D is the number of variables in the model. In the statistics literature, it is common to use an alternative definition of BIC, which we call the BIC *cost* (since we want to minimize it):

$$\text{BIC-cost} \triangleq -2 \log p(\mathcal{D}|\hat{\boldsymbol{\theta}}) + \text{dof}(\hat{\boldsymbol{\theta}}) \log N \approx -2 \log p(\mathcal{D}) \tag{5.33}$$

In the context of linear regression, this becomes

$$\text{BIC-cost} = N \log(\hat{\sigma}^2) + D \log(N) \tag{5.34}$$

2. Traditionally the BIC score is defined using the ML estimate $\hat{\boldsymbol{\theta}}$, so it is independent of the prior. However, for models such as mixtures of Gaussians, the ML estimate can be poorly behaved, so it is better to evaluate the BIC score using the MAP estimate, as in (Fraley and Raftery 2007).

The BIC method is very closely related to the **minimum description length** or **MDL** principle, which characterizes the score for a model in terms of how well it fits the data, minus how complex the model is to define. See (Hansen and Yu 2001) for details.

There is a very similar expression to BIC/ MDL called the **Akaike information criterion** or **AIC**, defined as

$$\text{AIC}(m, \mathcal{D}) \triangleq \log p(\mathcal{D}|\hat{\boldsymbol{\theta}}_{MLE}) - \text{dof}(m) \tag{5.35}$$

This is derived from a frequentist framework, and cannot be interpreted as an approximation to the marginal likelihood. Nevertheless, the form of this expression is very similar to BIC. We see that the penalty for AIC is less than for BIC. This causes AIC to pick more complex models. However, this can result in better predictive accuracy. See e.g., (Clarke et al. 2009, sec 10.2) for further discussion on such information criteria.

5.3.2.5 Effect of the prior

Sometimes it is not clear how to set the prior. When we are performing posterior inference, the details of the prior may not matter too much, since the likelihood often overwhelms the prior anyway. But when computing the marginal likelihood, the prior plays a much more important role, since we are averaging the likelihood over all possible parameter settings, as weighted by the prior.

In Figures 5.7 and 5.8, where we demonstrated model selection for linear regression, we used a prior of the form $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \alpha^{-1}\mathbf{I})$. Here α is a tuning parameter that controls how strong the prior is. This parameter can have a large effect, as we discuss in Section 7.5. Intuitively, if α is large, the weights are "forced" to be small, so we need to use a complex model with many small parameters (e.g., a high degree polynomial) to fit the data. Conversely, if α is small, we will favor simpler models, since each parameter is "allowed" to vary in magnitude by a lot.

If the prior is unknown, the correct Bayesian procedure is to put a prior on the prior. That is, we should put a prior on the hyper-parameter α as well as the parametrs $\mathbf{w}$. To compute the marginal likelihood, we should integrate out all unknowns, i.e., we should compute

$$p(\mathcal{D}|m) = \int\int p(\mathcal{D}|\mathbf{w})p(\mathbf{w}|\alpha, m)p(\alpha|m)d\mathbf{w}d\alpha \tag{5.36}$$

Of course, this requires specifying the hyper-prior. Fortunately, the higher up we go in the Bayesian hierarchy, the less sensitive are the results to the prior settings. So we can usually make the hyper-prior uninformative.

A computational shortcut is to optimize α rather than integrating it out. That is, we use

$$p(\mathcal{D}|m) \approx \int p(\mathcal{D}|\mathbf{w})p(\mathbf{w}|\hat{\alpha}, m)d\mathbf{w} \tag{5.37}$$

where

$$\hat{\alpha} = \operatorname*{argmax}_{\alpha} p(\mathcal{D}|\alpha, m) = \operatorname*{argmax}_{\alpha} \int p(\mathcal{D}|\mathbf{w})p(\mathbf{w}|\alpha, m)d\mathbf{w} \tag{5.38}$$

This approach is called empirical Bayes (EB), and is discussed in more detail in Section 5.6. This is the method used in Figures 5.7 and 5.8.

Bayes factor $BF(1,0)$	Interpretation
$BF < \frac{1}{100}$	Decisive evidence for M_0
$BF < \frac{1}{10}$	Strong evidence for M_0
$\frac{1}{10} < BF < \frac{1}{3}$	Moderate evidence for M_0
$\frac{1}{3} < BF < 1$	Weak evidence for M_0
$1 < BF < 3$	Weak evidence for M_1
$3 < BF < 10$	Moderate evidence for M_1
$BF > 10$	Strong evidence for M_1
$BF > 100$	Decisive evidence for M_1

Table 5.1 Jeffreys' scale of evidence for interpreting Bayes factors.

5.3.3 Bayes factors

Suppose our prior on models is uniform, $p(m) \propto 1$. Then model selection is equivalent to picking the model with the highest marginal likelihood. Now suppose we just have two models we are considering, call them the **null hypothesis**, M_0, and the **alternative hypothesis**, M_1. Define the **Bayes factor** as the ratio of marginal likelihoods:

$$BF_{1,0} \triangleq \frac{p(\mathcal{D}|M_1)}{p(\mathcal{D}|M_0)} = \frac{p(M_1|\mathcal{D})}{p(M_0|\mathcal{D})} \Big/ \frac{p(M_1)}{p(M_0)} \tag{5.39}$$

(This is like a **likelihood ratio**, except we integrate out the parameters, which allows us to compare models of different complexity.) If $BF_{1,0} > 1$ then we prefer model 1, otherwise we prefer model 0.

Of course, it might be that $BF_{1,0}$ is only slightly greater than 1. In that case, we are not very confident that model 1 is better. Jeffreys (1961) proposed a scale of evidence for interpreting the magnitude of a Bayes factor, which is shown in Table 5.1. This is a Bayesian alternative to the frequentist concept of a p-value.[3] Alternatively, we can just convert the Bayes factor to a posterior over models. If $p(M_1) = p(M_0) = 0.5$, we have

$$p(M_0|\mathcal{D}) = \frac{BF_{0,1}}{1 + BF_{0,1}} = \frac{1}{BF_{1,0} + 1} \tag{5.40}$$

5.3.3.1 Example: Testing if a coin is fair

Suppose we observe some coin tosses, and want to decide if the data was generated by a fair coin, $\theta = 0.5$, or a potentially biased coin, where θ could be any value in $[0, 1]$. Let us denote the first model by M_0 and the second model by M_1. The marginal likelihood under M_0 is simply

$$p(\mathcal{D}|M_0) = \left(\frac{1}{2}\right)^N \tag{5.41}$$

3. A **p-value**, is defined as the probability (under the null hypothesis) of observing some **test statistic** $f(\mathcal{D})$ (such as the **chi-squared statistic**) that is as large *or larger* than that actually observed, i.e., $\text{pvalue}(\mathcal{D}) \triangleq P(f(\tilde{\mathcal{D}}) \geq f(\mathcal{D})|\tilde{\mathcal{D}} \sim H_0)$. Note that has almost nothing to do with what we really want to know, which is $p(H_0|\mathcal{D})$.

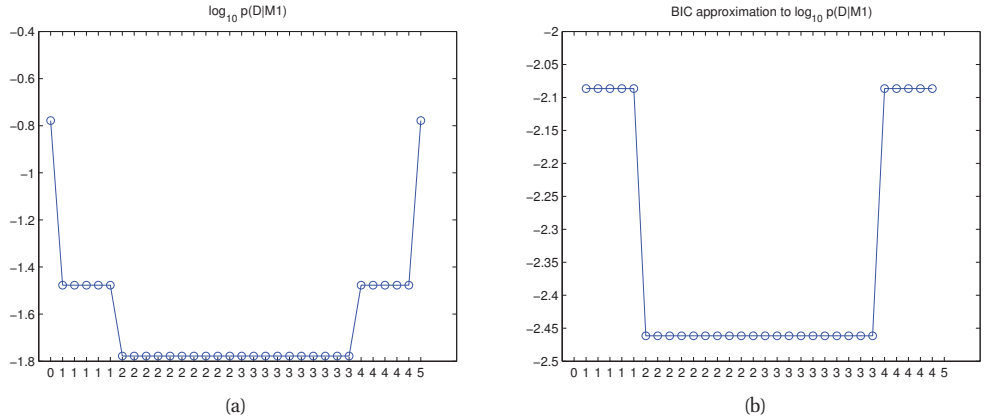


Figure 5.9 (a) Log marginal likelihood for the coins example. (b) BIC approximation. Figure generated by `coinsModelSelDemo`.

where N is the number of coin tosses. The marginal likelihood under M_1, using a Beta prior, is

$$p(\mathcal{D}|M_1) = \int p(\mathcal{D}|\theta)p(\theta)d\theta = \frac{B(\alpha_1 + N_1, \alpha_0 + N_0)}{B(\alpha_1, \alpha_0)} \tag{5.42}$$

We plot $\log p(\mathcal{D}|M_1)$ vs the number of heads N_1 in Figure 5.9(a), assuming $N = 5$ and $\alpha_1 = \alpha_0 = 1$. (The shape of the curve is not very sensitive to α_1 and α_0, as long as $\alpha_0 = \alpha_1$.) If we observe 2 or 3 heads, the unbiased coin hypothesis M_0 is more likely than M_1, since M_0 is a simpler model (it has no free parameters) — it would be a suspicious coincidence if the coin were biased but happened to produce almost exactly 50/50 heads/tails. However, as the counts become more extreme, we favor the biased coin hypothesis. Note that, if we plot the log Bayes factor, $\log BF_{1,0}$, it will have exactly the same shape, since $\log p(\mathcal{D}|M_0)$ is a constant. See also Exercise 3.18.

In Figure 5.9(b) shows the BIC approximation to $\log p(\mathcal{D}|M_1)$ for our biased coin example from Section 5.3.3.1. We see that the curve has approximately the same shape as the exact log marginal likelihood, which is all that matters for model selection purposes, since the absolute scale is irrelevant. In particular, it favors the simpler model unless the data is overwhelmingly in support of the more complex model.

5.3.4 Jeffreys-Lindley paradox *

Problems can arise when we use improper priors (i.e., priors that do not integrate to 1) for model selection/ hypothesis testing, even though such priors may be acceptable for other purposes. For example, consider testing the hypotheses $M_0 : \theta \in \Theta_0$ vs $M_1 : \theta \in \Theta_1$. To define the marginal density on θ, we use the following mixture model

$$p(\theta) = p(\theta|M_0)p(M_0) + p(\theta|M_1)p(M_1) \tag{5.43}$$

This is only meaningful if $p(\theta|M_0)$ and $p(\theta|M_1)$ are proper (normalized) density functions. In this case, the posterior is given by

$$p(M_0|\mathcal{D}) = \frac{p(M_0)p(\mathcal{D}|M_0)}{p(M_0)p(\mathcal{D}|M_0) + p(M_1)p(\mathcal{D}|M_1)} \tag{5.44}$$

$$= \frac{p(M_0)\int_{\Theta_0} p(\mathcal{D}|\theta)p(\theta|M_0)d\theta}{p(M_0)\int_{\Theta_0} p(\mathcal{D}|\theta)p(\theta|M_0)d\theta + p(M_1)\int_{\Theta_1} p(\mathcal{D}|\theta)p(\theta|M_1)d\theta} \tag{5.45}$$

Now suppose we use improper priors, $p(\theta|M_0) \propto c_0$ and $p(\theta|M_1) \propto c_1$. Then

$$p(M_0|\mathcal{D}) = \frac{p(M_0)c_0\int_{\Theta_0} p(\mathcal{D}|\theta)d\theta}{p(M_0)c_0\int_{\Theta_0} p(\mathcal{D}|\theta)d\theta + p(M_1)c_1\int_{\Theta_1} p(\mathcal{D}|\theta)d\theta} \tag{5.46}$$

$$= \frac{p(M_0)c_0\ell_0}{p(M_0)c_0\ell_0 + p(M_1)c_1\ell_1} \tag{5.47}$$

where $\ell_i = \int_{\Theta_i} p(\mathcal{D}|\theta)d\theta$ is the integrated or marginal likelihood for model i. Now let $p(M_0) = p(M_1) = \frac{1}{2}$. Hence

$$p(M_0|\mathcal{D}) = \frac{c_0\ell_0}{c_0\ell_0 + c_1\ell_1} = \frac{\ell_0}{\ell_0 + (c_1/c_0)\ell_1} \tag{5.48}$$

Thus we can change the posterior arbitrarily by choosing c_1 and c_0 as we please. Note that using proper, but very vague, priors can cause similar problems. In particular, the Bayes factor will always favor the simpler model, since the probability of the observed data under a complex model with a very diffuse prior will be very small. This is called the **Jeffreys-Lindley paradox**.

Thus it is important to use proper priors when performing model selection. Note, however, that, if M_0 and M_1 share the same prior over a subset of the parameters, this part of the prior can be improper, since the corresponding normalization constant will cancel out.

5.4 Priors

The most controversial aspect of Bayesian statistics is its reliance on priors. Bayesians argue this is unavoidable, since nobody is a **tabula rasa** or **blank slate**: all inference must be done conditional on certain assumptions about the world. Nevertheless, one might be interested in minimizing the impact of one's prior assumptions. We briefly discuss some ways to do this below.

5.4.1 Uninformative priors

If we don't have strong beliefs about what θ should be, it is common to use an **uninformative** or **non-informative** prior, and to "let the data speak for itself".

The issue of designing uninformative priors is actually somewhat tricky. As an example of the difficulty, consider a Bernoulli parameter, $\theta \in [0, 1]$. One might think that the most uninformative prior would be the uniform distribution, $\mathrm{Beta}(1, 1)$. But the posterior mean in this case is $\mathbb{E}[\theta|\mathcal{D}] = \frac{N_1+1}{N_1+N_0+2}$, whereas the MLE is $\frac{N_1}{N_1+N_0}$. Hence one could argue that the prior wasn't completely uninformative after all.

Clearly by decreasing the magnitude of the pseudo counts, we can lessen the impact of the prior. By the above argument, the most non-informative prior is

$$\lim_{c \to 0} \text{Beta}(c, c) = \text{Beta}(0, 0) \tag{5.49}$$

which is a mixture of two equal point masses at 0 and 1 (see (Zhu and Lu 2004)). This is also called the **Haldane prior**. Note that the Haldane prior is an improper prior, meaning it does not integrate to 1. However, as long as we see at least one head and at least one tail, the posterior will be proper.

In Section 5.4.2.1 we will argue that the "right" uninformative prior is in fact $\text{Beta}(\frac{1}{2}, \frac{1}{2})$. Clearly the difference in practice between these three priors is very likely negligible. In general, it is advisable to perform some kind of **sensitivity analysis**, in which one checks how much one's conclusions or predictions change in response to change in the modeling assumptions, which includes the choice of prior, but also the choice of likelihood and any kind of data pre-processing. If the conclusions are relatively insensitive to the modeling assumptions, one can have more confidence in the results.

5.4.2 Jeffreys priors *

Harold Jeffreys[4] designed a general purpose technique for creating non-informative priors. The result is known as the **Jeffreys prior**. The key observation is that if $p(\phi)$ is non-informative, then any re-parameterization of the prior, such as $\theta = h(\phi)$ for some function h, should also be non-informative. Now, by the change of variables formula,

$$p_\theta(\theta) = p_\phi(\phi) \left| \frac{d\phi}{d\theta} \right| \tag{5.50}$$

so the prior will in general change. However, let us pick

$$p_\phi(\phi) \propto (I(\phi))^{\frac{1}{2}} \tag{5.51}$$

where $I(\phi)$ is the **Fisher information**:

$$I(\phi) \triangleq -\mathbb{E}\left[\left(\frac{d \log p(X|\phi)}{d\phi} \right) 2 \right] \tag{5.52}$$

This is a measure of curvature of the expected negative log likelihood and hence a measure of stability of the MLE (see Section 6.2.2). Now

$$\frac{d \log p(x|\theta)}{d\theta} = \frac{d \log p(x|\phi)}{d\phi} \frac{d\phi}{d\theta} \tag{5.53}$$

Squaring and taking expectations over x, we have

$$I(\theta) = -\mathbb{E}\left[\left(\frac{d \log p(X|\theta)}{d\theta} \right)^2 \right] = I(\phi) \left(\frac{d\phi}{d\theta} \right)^2 \tag{5.54}$$

$$I(\theta)^{\frac{1}{2}} = I(\phi)^{\frac{1}{2}} \left| \frac{d\phi}{d\theta} \right| \tag{5.55}$$

4. Harold Jeffreys, 1891 – 1989, was an English mathematician, statistician, geophysicist, and astronomer.

so we find the transformed prior is

$$p_\theta(\theta) \;=\; p_\phi(\phi)\left|\frac{d\phi}{d\theta}\right| \propto (I(\phi))^{\frac{1}{2}}\left|\frac{d\phi}{d\theta}\right| = I(\theta)^{\frac{1}{2}} \tag{5.56}$$

So $p_\theta(\theta)$ and $p_\phi(\phi)$ are the same.

Some examples will make this clearer.

5.4.2.1 Example: Jeffreys prior for the Bernoulli and multinoulli

Suppose $X \sim \mathrm{Ber}(\theta)$. The log likelihood for a single sample is

$$\log p(X|\theta) = X \log \theta + (1 - X) \log(1 - \theta) \tag{5.57}$$

The **score function** is just the gradient of the log-likelihood:

$$s(\theta) \;\triangleq\; \frac{d}{d\theta} \log p(X|\theta) = \frac{X}{\theta} - \frac{1 - X}{1 - \theta} \tag{5.58}$$

The **observed information** is the second derivative of the log-likelihood:

$$J(\theta) \;=\; -\frac{d^2}{d\theta^2} \log p(X|\theta) = -s'(\theta|X) = \frac{X}{\theta^2} + \frac{1 - X}{(1 - \theta)^2} \tag{5.59}$$

The Fisher information is the expected information:

$$I(\theta) \;=\; E[J(\theta|X)|X \sim \theta] = \frac{\theta}{\theta^2} + \frac{1 - \theta}{(1 - \theta)^2} = \frac{1}{\theta(1 - \theta)} \tag{5.60}$$

Hence Jeffreys' prior is

$$p(\theta) \propto \theta^{-\frac{1}{2}}(1 - \theta)^{-\frac{1}{2}} = \frac{1}{\sqrt{\theta(1 - \theta)}} \propto \mathrm{Beta}(\frac{1}{2}, \frac{1}{2}) \tag{5.61}$$

Now consider a multinoulli random variable with K states. One can show that the Jeffreys' prior is given by

$$p(\boldsymbol{\theta}) \propto \mathrm{Dir}(\frac{1}{2}, \ldots, \frac{1}{2}) \tag{5.62}$$

Note that this is different from the more obvious choices of $\mathrm{Dir}(\frac{1}{K}, \ldots, \frac{1}{K})$ or $\mathrm{Dir}(1, \ldots, 1)$.

5.4.2.2 Example: Jeffreys prior for location and scale parameters

One can show that the Jeffreys prior for a location parameter, such as the Gaussian mean, is $p(\mu) \propto 1$. Thus is an example of a **translation invariant prior**, which satisfies the property that the probability mass assigned to any interval, $[A, B]$ is the same as that assigned to any other shifted interval of the same width, such as $[A - c, B - c]$. That is,

$$\int_{A-c}^{B-c} p(\mu)d\mu = (A - c) - (B - c) = (A - B) = \int_A^B p(\mu)d\mu \tag{5.63}$$

This can be achieved using $p(\mu) \propto 1$, which we can approximate by using a Gaussian with infinite variance, $p(\mu) = \mathcal{N}(\mu|0, \infty)$. Note that this is an **improper prior**, since it does not integrate to 1. Using improper priors is fine as long as the posterior is proper, which will be the case provided we have seen $N \geq 1$ data points, since we can "nail down" the location as soon as we have seen a single data point.

Similarly, one can show that the Jeffreys prior for a scale parameter, such as the Gaussian variance, is $p(\sigma^2) \propto 1/\sigma^2$. This is an example of a **scale invariant prior**, which satisfies the property that the probability mass assigned to any interval $[A, B]$ is the same as that assigned to any other interval $[A/c, B/c]$ which is scaled in size by some constant factor $c > 0$. (For example, if we change units from meters to feet we do not want that to affect our inferences.) This can be achieved by using

$$p(s) \propto 1/s \tag{5.64}$$

To see this, note that

$$\int_{A/c}^{B/c} p(s)ds = [\log s]_{A/c}^{B/c} = \log(B/c) - \log(A/c) \tag{5.65}$$

$$= \log(B) - \log(A) = \int_A^B p(s)ds \tag{5.66}$$

We can approximate this using a degenerate Gamma distribution (Section 2.4.4), $p(s) = \mathrm{Ga}(s|0, 0)$. The prior $p(s) \propto 1/s$ is also improper, but the posterior is proper as soon as we have seen $N \geq 2$ data points (since we need at least two data points to estimate a variance).

5.4.3 Robust priors

In many cases, we are not very confident in our prior, so we want to make sure it does not have an undue influence on the result. This can be done by using **robust priors** (Insua and Ruggeri 2000), which typically have heavy tails, which avoids forcing things to be too close to the prior mean.

Let us consider an example from (Berger 1985, p7). Suppose $x \sim \mathcal{N}(\theta, 1)$. We observe that $x = 5$ and we want to estimate θ. The MLE is of course $\hat{\theta} = 5$, which seems reasonable. The posterior mean under a uniform prior is also $\overline{\theta} = 5$. But now suppose we know that the prior median is 0, and the prior quantiles are at -1 and 1, so $p(\theta \leq -1) = p(-1 < \theta \leq 0) = p(0 < \theta \leq 1) = p(1 < \theta) = 0.25$. Let us also assume the prior is smooth and unimodal.

It is easy to show that a Gaussian prior of the form $\mathcal{N}(\theta|0, 2.19^2)$ satisfies these prior constraints. But in this case the posterior mean is given by 3.43, which doesn't seem very satisfactory.

Now suppose we use as a Cauchy prior $\mathcal{T}(\theta|0, 1, 1)$. This also satisfies the prior constraints of our example. But this time we find (using numerical method integration: see `robustPriorDemo` for the code) that the posterior mean is about 4.6, which seems much more reasonable.

5.4.4 Mixtures of conjugate priors

Robust priors are useful, but can be computationally expensive to use. Conjugate priors simplify the computation, but are often not robust, and not flexible enough to encode our prior knowl-

edge. However, it turns out that a **mixture of conjugate priors** is also conjugate (Exercise 5.1), and can approximate any kind of prior (Dallal and Hall 1983; Diaconis and Ylvisaker 1985). Thus such priors provide a good compromise between computational convenience and flexibility.

For example, suppose we are modeling coin tosses, and we think the coin is either fair, or is biased towards heads. This cannot be represented by a beta distribution. However, we can model it using a mixture of two beta distributions. For example, we might use

$$p(\theta) = 0.5\ \mathrm{Beta}(\theta|20, 20) + 0.5\ \mathrm{Beta}(\theta|30, 10) \tag{5.67}$$

If θ comes from the first distribution, the coin is fair, but if it comes from the second, it is biased towards heads.

We can represent a mixture by introducing a latent indicator variable z, where $z = k$ means that θ comes from mixture component k. The prior has the form

$$p(\theta) = \sum_k p(z = k)p(\theta|z = k) \tag{5.68}$$

where each $p(\theta|z = k)$ is conjugate, and $p(z = k)$ are called the (prior) mixing weights. One can show (Exercise 5.1) that the posterior can also be written as a mixture of conjugate distributions as follows:

$$p(\theta|\mathcal{D}) \quad = \quad \sum_k p(z = k|\mathcal{D})p(\theta|\mathcal{D}, z = k) \tag{5.69}$$

where $p(Z = k|\mathcal{D})$ are the posterior mixing weights given by

$$p(Z = k|\mathcal{D}) \quad = \quad \frac{p(Z = k)p(\mathcal{D}|Z = k)}{\sum_{k'} p(Z = k')p(\mathcal{D}|Z = k')} \tag{5.70}$$

Here the quantity $p(\mathcal{D}|Z = k)$ is the marginal likelihood for mixture component k (see Section 5.3.2.1).

5.4.4.1 Example

Suppose we use the mixture prior

$$p(\theta) = 0.5\mathrm{Beta}(\theta|a_1, b_1) + 0.5\mathrm{Beta}(\theta|a_2, b_2) \tag{5.71}$$

where $a_1 = b_1 = 20$ and $a_2 = b_2 = 10$. and we observe N_1 heads and N_0 tails. The posterior becomes

$$p(\theta|\mathcal{D}) = p(Z = 1|\mathcal{D})\mathrm{Beta}(\theta|a_1 + N_1, b_1 + N_0) + p(Z = 2|\mathcal{D})\mathrm{Beta}(\theta|a_2 + N_1, b_2 + N_0) \tag{5.72}$$

If $N_1 = 20$ heads and $N_0 = 10$ tails, then, using Equation 5.23, the posterior becomes

$$p(\theta|\mathcal{D}) = 0.346\ \mathrm{Beta}(\theta|40, 30) + 0.654\ \mathrm{Beta}(\theta|50, 20) \tag{5.73}$$

See Figure 5.10 for an illustration.

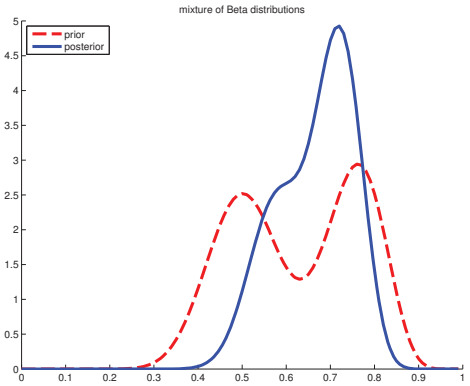


Figure 5.10 A mixture of two Beta distributions. Figure generated by `mixBetaDemo`.

5.4.4.2 Application: Finding conserved regions in DNA and protein sequences

We mentioned that Dirichlet-multinomial models are widely used in biosequence analysis. Let us give a simple example to illustrate some of the machinery that has developed. Specifically, consider the sequence logo discussed in Section 2.3.2.1. Now suppose we want to find locations which represent coding regions of the genome. Such locations often have the same letter across all sequences, because of evolutionary pressure. So we need to find columns which are "pure", or nearly so, in the sense that they are mostly all As, mostly all Ts, mostly all Cs, or mostly all Gs. One approach is to look for low-entropy columns; these will be ones whose distribution is nearly deterministic (pure).

But suppose we want to associate a confidence measure with our estimates of purity. This can be useful if we believe adjacent locations are conserved together. In this case, we can let $Z_1 = 1$ if location t is conserved, and let $Z_t = 0$ otherwise. We can then add a dependence between adjacent Z_t variables using a Markov chain; see Chapter 17 for details.

In any case, we need to define a likelihood model, $p(\mathbf{N}_t|Z_t)$, where $\mathbf{N}_t$ is the vector of (A,C,G,T) counts for column t. It is natural to make this be a multinomial distribution with parameter $\boldsymbol{\theta}_t$. Since each column has a different distribution, we will want to integrate out $\boldsymbol{\theta}_t$ and thus compute the marginal likelihood

$$p(\mathbf{N}_t|Z_t) = \int p(\mathbf{N}_t|\boldsymbol{\theta}_t)p(\boldsymbol{\theta}_t|Z_t)d\boldsymbol{\theta}_t \tag{5.74}$$

But what prior should we use for $\boldsymbol{\theta}_t$? When $Z_t = 0$ we can use a uniform prior, $p(\boldsymbol{\theta}|Z_t = 0) = \mathrm{Dir}(1, 1, 1, 1)$, but what should we use if $Z_t = 1$? After all, if the column is conserved, it could be a (nearly) pure column of As, Cs, Gs, or Ts. A natural approach is to use a mixture of Dirichlet priors, each one of which is "tilted" towards the appropriate corner of the 4-dimensional simplex, e.g.,

$$p(\boldsymbol{\theta}|Z_t = 1) = \frac{1}{4}\mathrm{Dir}(\boldsymbol{\theta}|(10, 1, 1, 1)) + \cdots + \frac{1}{4}\mathrm{Dir}(\boldsymbol{\theta}|(1, 1, 1, 10)) \tag{5.75}$$

Since this is conjugate, we can easily compute $p(\mathbf{N}_t|Z_t)$. See (Brown et al. 1993) for an

application of these ideas to a real bio-sequence problem.

5.5 Hierarchical Bayes

A key requirement for computing the posterior $p(\boldsymbol{\theta}|\mathcal{D})$ is the specification of a prior $p(\boldsymbol{\theta}|\boldsymbol{\eta})$, where $\boldsymbol{\eta}$ are the hyper-parameters. What if we don't know how to set $\boldsymbol{\eta}$? In some cases, we can use uninformative priors, we we discussed above. A more Bayesian approach is to put a prior on our priors! In terms of graphical models (Chapter 10), we can represent the situation as follows:

$$\boldsymbol{\eta} \rightarrow \boldsymbol{\theta} \rightarrow \mathcal{D} \tag{5.76}$$

This is an example of a **hierarchical Bayesian model**, also called a **multi-level model**, since there are multiple levels of unknown quantities. We give a simple example below, and we will see many others later in the book.

5.5.1 Example: modeling related cancer rates

Consider the problem of predicting cancer rates in various cities (this example is from (Johnson and Albert 1999, p24)). In particular, suppose we measure the number of people in various cities, N_i, and the number of people who died of cancer in these cities, x_i. We assume $x_i \sim \text{Bin}(N_i, \theta_i)$, and we want to estimate the cancer rates θ_i. One approach is to estimate them all separately, but this will suffer from the sparse data problem (underestimation of the rate of cancer due to small N_i). Another approach is to assume all the θ_i are the same; this is called **parameter tying**. The resulting pooled MLE is just $\hat{\theta} = \frac{\sum_i x_i}{\sum_i N_i}$. But the assumption that all the cities have the same rate is a rather strong one. A compromise approach is to assume that the θ_i are similar, but that there may be city-specific variations. This can be modeled by assuming the θ_i are drawn from some common distribution, say $\theta_i \sim \text{Beta}(a, b)$. The full joint distribution can be written as

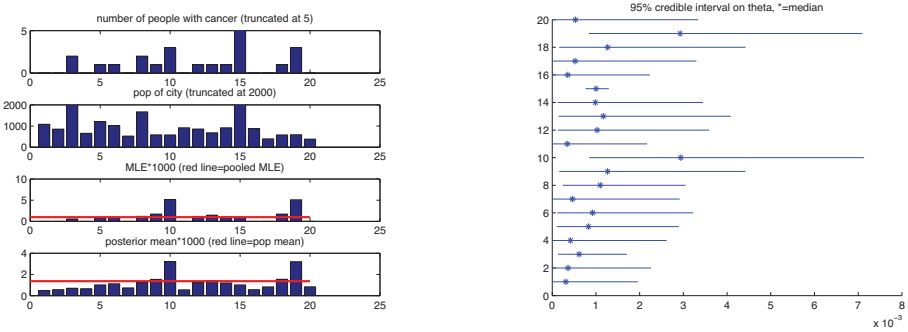
$$p(\mathcal{D}, \boldsymbol{\theta}, \boldsymbol{\eta}|\mathbf{N}) \quad = \quad p(\boldsymbol{\eta}) \prod_{i=1}^{N} \text{Bin}(x_i|N_i, \theta_i)\text{Beta}(\theta_i|\boldsymbol{\eta}) \tag{5.77}$$

where $\boldsymbol{\eta} = (a, b)$.

Note that it is crucial that we infer $\boldsymbol{\eta} = (a, b)$ from the data; if we just clamp it to a constant, the θ_i will be conditionally independent, and there will be no information flow between them. By contrast, by treating $\boldsymbol{\eta}$ as an unknown (hidden variable), we allow the data-poor cities to **borrow statistical strength** from data-rich ones.

Suppose we compute the joint posterior $p(\boldsymbol{\eta}, \boldsymbol{\theta}|\mathcal{D})$. From this we can get the posterior marginals $p(\theta_i|\mathcal{D})$. In Figure 5.11(a), we plot the posterior means, $\mathbb{E}[\theta_i|\mathcal{D}]$, as blue bars, as well as the population level mean, $\mathbb{E}[a/(a+b)|\mathcal{D}]$, shown as a red line (this represents the average of the θ_i's). We see that the posterior mean is shrunk towards the pooled estimate more strongly for cities with small sample sizes N_i. For example, city 1 and city 20 both have a 0 observed cancer incidence rate, but city 20 has a smaller population, so its rate is shrunk more towards the population-level estimate (i.e., it is closer to the horizontal red line) than city 1.

Figure 5.11(b) shows the 95% posterior credible intervals for θ_i. We see that city 15, which has a very large population (53,637 people), has small posterior uncertainty. Consequently this city



(a) (b)

Figure 5.11 (a) Results of fitting the model using the data from (Johnson and Albert 1999, p24). First row: Number of cancer incidents x_i in 20 cities in Missouri. Second row: population size N_i. The largest city (number 15) has a population of $N_{15} = 53637$ and $x_{15} = 54$ incidents, but we truncate the vertical axes of the first two rows so that the differences between the other cities are visible. Third row: MLE $\hat{\theta}_i$. The red line is the pooled MLE. Fourth row: posterior mean $\mathbb{E}[\theta_i|\mathcal{D}]$. The red line is $\mathbb{E}[a/(a+b)|\mathcal{D}]$, the population-level mean. (b) Posterior 95% credible intervals on the cancer rates. Figure generated by `cancerRatesEb`

has the largest impact on the posterior estimate of $\boldsymbol{\eta}$, which in turn will impact the estimate of the cancer rates for other cities. Cities 10 and 19, which have the highest MLE, also have the highest posterior uncertainty, reflecting the fact that such a high estimate is in conflict with the prior (which is estimated from all the other cities).

In the above example, we have one parameter per city, modeling the probability the response is on. By making the Bernoulli rate parameter be a function of covariates, $\theta_i = \text{sigm}(\mathbf{w}_i^T \mathbf{x})$, we can model multiple correlated logistic regression tasks. This is called **multi-task learning**, and will be discussed in more detail in Section 9.5.

5.6 Empirical Bayes

In hierarchical Bayesian models, we need to compute the posterior on multiple levels of latent variables. For example, in a two-level model, we need to compute

$$p(\boldsymbol{\eta}, \boldsymbol{\theta}|\mathcal{D}) \propto p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{\eta})p(\boldsymbol{\eta}) \tag{5.78}$$

In some cases, we can analytically marginalize out $\boldsymbol{\theta}$; this leaves is with the simpler problem of just computing $p(\boldsymbol{\eta}|\mathcal{D})$.

As a computational shortcut, we can approximate the posterior on the hyper-parameters with a point-estimate, $p(\boldsymbol{\eta}|\mathcal{D}) \approx \delta_{\hat{\boldsymbol{\eta}}}(\boldsymbol{\eta})$, where $\hat{\boldsymbol{\eta}} = \text{argmax}\, p(\boldsymbol{\eta}|\mathcal{D})$. Since $\boldsymbol{\eta}$ is typically much smaller than $\boldsymbol{\theta}$ in dimensionality, it is less prone to overfitting, so we can safely use a uniform prior on $\boldsymbol{\eta}$. Then the estimate becomes

$$\hat{\boldsymbol{\eta}} = \text{argmax}\, p(\mathcal{D}|\boldsymbol{\eta}) = \text{argmax} \left[\int p(\mathcal{D}|\boldsymbol{\theta})p(\boldsymbol{\theta}|\boldsymbol{\eta})d\boldsymbol{\theta} \right] \tag{5.79}$$

where the quantity inside the brackets is the marginal or integrated likelihood, sometimes called the evidence. This overall approach is called **empirical Bayes** (**EB**) or **type-II maximum likelihood**. In machine learning, it is sometimes called the **evidence procedure**.

Empirical Bayes violates the principle that the prior should be chosen independently of the data. However, we can just view it as a computationally cheap approximation to inference in a hierarchical Bayesian model, just as we viewed MAP estimation as an approximation to inference in the one level model $\boldsymbol{\theta} \rightarrow \mathcal{D}$. In fact, we can construct a hierarchy in which the more integrals one performs, the "more Bayesian" one becomes:

Method	Definition
Maximum likelihood	$\hat{\boldsymbol{\theta}} = \text{argmax}_{\boldsymbol{\theta}}\, p(\mathcal{D}\|\boldsymbol{\theta})$
MAP estimation	$\hat{\boldsymbol{\theta}} = \text{argmax}_{\boldsymbol{\theta}}\, p(\mathcal{D}\|\boldsymbol{\theta})p(\boldsymbol{\theta}\|\boldsymbol{\eta})$
ML-II (Empirical Bayes)	$\hat{\boldsymbol{\eta}} = \text{argmax}_{\boldsymbol{\eta}} \int p(\mathcal{D}\|\boldsymbol{\theta})p(\boldsymbol{\theta}\|\boldsymbol{\eta})d\boldsymbol{\theta} = \text{argmax}_{\boldsymbol{\eta}}\, p(\mathcal{D}\|\boldsymbol{\eta})$
MAP-II	$\hat{\boldsymbol{\eta}} = \text{argmax}_{\boldsymbol{\eta}} \int p(\mathcal{D}\|\boldsymbol{\theta})p(\boldsymbol{\theta}\|\boldsymbol{\eta})p(\boldsymbol{\eta})d\boldsymbol{\theta} = \text{argmax}_{\boldsymbol{\eta}}\, p(\mathcal{D}\|\boldsymbol{\eta})p(\boldsymbol{\eta})$
Full Bayes	$p(\boldsymbol{\theta}, \boldsymbol{\eta}\|\mathcal{D}) \propto p(\mathcal{D}\|\boldsymbol{\theta})p(\boldsymbol{\theta}\|\boldsymbol{\eta})p(\boldsymbol{\eta})$

Note that EB can be shown to have good frequentist properties (see e.g., (Carlin and Louis 1996; Efron 2010)), so it is widely used by non-Bayesians. For example, the popular James-Stein estimator, discussed in Section 6.3.3.2, can be derived using EB.

5.6.1 Example: beta-binomial model

Let us return to the cancer rates model. We can analytically integrate out the θ_i's, and write down the marginal likelihood directly, as follows:

$$p(\mathcal{D}|a,b) = \prod_i \int \text{Bin}(x_i|N_i,\theta_i)\text{Beta}(\theta_i|a,b)d\theta_i \tag{5.80}$$

$$= \prod_i \frac{B(a+x_i, b+N_i-x_i)}{B(a,b)} \tag{5.81}$$

Various ways of maximizing this wrt a and b are discussed in (Minka 2000e).

Having estimated a and b, we can plug in the hyper-parameters to compute the posterior $p(\theta_i|\hat{a},\hat{b},\mathcal{D})$ in the usual way, using conjugate analysis. The net result is that the posterior mean of each θ_i is a weighted average of its local MLE and the prior means, which depends on $\boldsymbol{\eta} = (a,b)$; but since $\boldsymbol{\eta}$ is estimated based on all the data, each θ_i is influenced by all the data.

5.6.2 Example: Gaussian-Gaussian model

We now study another example that is analogous to the cancer rates example, except the data is real-valued. We will use a Gaussian likelihood and a Gaussian prior. This will allow us to write down the solution analytically.

In particular, suppose we have data from multiple related groups. For example, x_{ij} could be the test score for student i in school j, for $j = 1:D$ and $i = 1:N_j$. We want to estimate the mean score for each school, θ_j. However, since the sample size, N_j, may be small for

some schools, we can regularize the problem by using a hierarchical Bayesian model, where we assume θ_j come from a common prior, $\mathcal{N}(\mu, \tau^2)$.

The joint distribution has the following form:

$$p(\boldsymbol{\theta}, \mathcal{D}|\boldsymbol{\eta}, \sigma^2) = \prod_{j=1}^{D} \mathcal{N}(\theta_j|\mu, \tau^2) \prod_{i=1}^{N_j} \mathcal{N}(x_{ij}|\theta_j, \sigma^2) \tag{5.82}$$

where we assume σ^2 is known for simplicity. (We relax this assumption in Exercise 24.4.) We explain how to estimate $\boldsymbol{\eta}$ below. Once we have estimated $\boldsymbol{\eta} = (\mu, \tau)$, we can compute the posteriors over the θ_j's. To do that, it simplifies matters to rewrite the joint distribution in the following form, exploiting the fact that N_j Gaussian measurements with values x_{ij} and variance σ^2 are equivalent to one measurement of value $\overline{x}_j \triangleq \frac{1}{N_j} \sum_{i=1}^{N_j} x_{ij}$ with variance $\sigma_j^2 \triangleq \sigma^2/N_j$. This yields

$$p(\boldsymbol{\theta}, \mathcal{D}|\hat{\boldsymbol{\eta}}, \sigma^2) = \prod_{j=1}^{D} \mathcal{N}(\theta_j|\hat{\mu}, \hat{\tau}^2) \mathcal{N}(\overline{x}_j|\theta_j, \sigma_j^2) \tag{5.83}$$

From this, it follows from the results of Section 4.4.1 that the posteriors are given by

$$p(\theta_j|\mathcal{D}, \hat{\mu}, \hat{\tau}^2) = \mathcal{N}(\theta_j|\hat{B}_j\hat{\mu} + (1 - \hat{B}_j)\overline{x}_j, (1 - \hat{B}_j)\sigma_j^2) \tag{5.84}$$

$$\hat{B}_j \triangleq \frac{\sigma_j^2}{\sigma_j^2 + \hat{\tau}^2} \tag{5.85}$$

where $\hat{\mu} = \overline{x}$ and $\hat{\tau}^2$ will be defined below.

The quantity $0 \leq \hat{B}_j \leq 1$ controls the degree of **shrinkage** towards the overall mean, μ. If the data is reliable for group j (e.g., because the sample size N_j is large), then σ_j^2 will be small relative to τ^2; hence $\hat{B}_j$ will be small, and we will put more weight on $\overline{x}_j$ when we estimate θ_j. However, groups with small sample sizes will get regularized (shrunk towards the overall mean μ) more heavily. We will see an example of this below.

If $\sigma_j = \sigma$ for all groups j, the posterior mean becomes

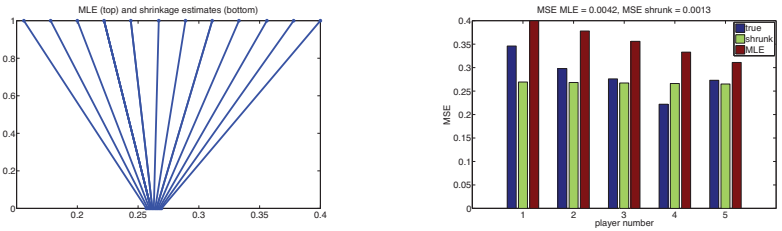
$$\hat{\theta}_j = \hat{B}\overline{x} + (1 - \hat{B})\overline{x}_j = \overline{x} + (1 - \hat{B})(\overline{x}_j - \overline{x}) \tag{5.86}$$

This has exactly the same form as the James Stein estimator discussed in Section 6.3.3.2.

5.6.2.1 Example: predicting baseball scores

We now give an example of shrinkage applied to baseball batting averages, from (Efron and Morris 1975). We observe the number of hits for $D = 18$ players during the first $T = 45$ games. Call the number of hits b_i. We assume $b_j \sim \text{Bin}(T, \theta_j)$, where θ_j is the "true" batting average for player j. The goal is to estimate the θ_j. The MLE is of course $\hat{\theta}_j = x_j$, where $x_j = b_j/T$ is the empirical batting average. However, we can use an EB approach to do better.

To apply the Gaussian shrinkage approach described above, we require that the likelihood be Gaussian, $x_j \sim \mathcal{N}(\theta_j, \sigma^2)$ for known σ^2. (We drop the i subscript since we assume $N_j = 1$,



(a) (b)

Figure 5.12 (a) MLE parameters (top) and corresponding shrunken estimates (bottom). (b) We plot the true parameters (blue), the posterior mean estimate (green), and the MLEs (red) for 5 of the players. Figure generated by `shrinkageDemoBaseball`.

since x_j already represents the average for player j.) However, in this example we have a binomial likelihood. While this has the right mean, $\mathbb{E}[x_j] = \theta_j$, the variance is not constant:

$$\text{var}[x_j] = \frac{1}{T^2}\,\text{var}[b_j] \;= \frac{T\theta_j(1-\theta_j)}{T^2} \tag{5.87}$$

So let us apply a **variance stabilizing transform**[5] to x_j to better match the Gaussian assumption:

$$y_j = f(y_j) = \sqrt{T}\arcsin(2y_j - 1) \tag{5.88}$$

Now we have approximately $y_j \sim \mathcal{N}(f(\theta_j), 1) = \mathcal{N}(\mu_j, 1)$. We use Gaussian shrinkage to estimate the μ_j using Equation 5.86 with $\sigma^2 = 1$, and we then transform back to get

$$\hat{\theta}_j = 0.5(\sin(\hat{\mu}_j/\sqrt{T}) + 1) \tag{5.89}$$

The results are shown in Figure 5.12(a-b). In (a), we plot the MLE $\hat{\theta}_j$ and the posterior mean $\overline{\theta}_j$. We see that all the estimates have shrunk towards the global mean, 0.265. In (b), we plot the true value θ_j, the MLE $\hat{\theta}_j$ and the posterior mean $\overline{\theta}_j$. (The "true" values of θ_j are estimated from a large number of independent games.) We see that, on average, the shrunken estimate is much closer to the true parameters than the MLE is. Specifically, the mean squared error, defined by $\text{MSE} = \frac{1}{N}\sum_{j=1}^{D}(\theta_j - \overline{\theta}_j)^2$, is over three times smaller using the shrinkage estimates $\overline{\theta}_j$ than using the MLEs $\hat{\theta}_j$.

5.6.2.2 Estimating the hyper-parameters

In this section, we give an algorithm for estimating $\boldsymbol{\eta}$. Suppose initially that $\sigma_j^2 = \sigma^2$ is the same for all groups. In this case, we can derive the EB estimate in closed form, as we now show. From Equation 4.126, we have

$$p(\overline{x}_j|\mu, \tau^2, \sigma^2) \;\; = \;\; \int \mathcal{N}(\overline{x}_j|\theta_j, \sigma^2)\mathcal{N}(\theta_j|\mu, \tau^2)d\theta_j = \mathcal{N}(\overline{x}_j|\mu, \tau^2 + \sigma^2) \tag{5.90}$$

5. Suppose $\mathbb{E}[X] = \mu$ and $\text{var}[X] = \sigma^2(\mu)$. Let $Y = f(X)$. Then a Taylor series expansions gives $Y \approx f(\mu) + (X - \mu)f'(\mu)$. Hence $\text{var}[Y] \approx f'(\mu)^2\text{var}[X - \mu] = f'(\mu)^2\sigma^2(\mu)$. A variance stabilizing transformation is a function f such that $f'(\mu)^2\sigma^2(\mu)$ is independent of μ.

Hence the marginal likelihood is

$$p(\mathcal{D}|\mu, \tau^2, \sigma^2) = \prod_{j=1}^{D} \mathcal{N}(\overline{x}_j|\mu, \tau^2 + \sigma^2) \tag{5.91}$$

Thus we can estimate the hyper-parameters using the usual MLEs for a Gaussian. For μ, we have

$$\hat{\mu} = \frac{1}{D}\sum_{j=1}^{D} \overline{x}_j = \overline{x} \tag{5.92}$$

which is the overall mean.

For the variance, we can use moment matching (which is equivalent to the MLE for a Gaussian): we simply equate the model variance to the empirical variance:

$$\hat{\tau}^2 + \sigma^2 = \frac{1}{D}\sum_{j=1}^{D} (\overline{x}_j - \overline{x})^2 \triangleq s^2 \tag{5.93}$$

so $\hat{\tau}^2 = s^2 - \sigma^2$. Since we know τ^2 must be positive, it is common to use the following revised estimate:

$$\hat{\tau}^2 = \max\{0, s^2 - \sigma^2\} = (s^2 - \sigma^2)_+ \tag{5.94}$$

Hence the shrinkage factor is

$$\hat{B} = \frac{\sigma^2}{\sigma^2 + \hat{\tau}^2} = \frac{\sigma^2}{\sigma^2 + (s^2 - \sigma^2)_+} \tag{5.95}$$

In the case where the σ_j^2's are different, we can no longer derive a solution in closed form. Exercise 11.13 discusses how to use the EM algorithm to derive an EB estimate, and Exercise 24.4 discusses how to perform full Bayesian inference in this hierarchical model.

5.7 Bayesian decision theory

We have seen how probability theory can be used to represent and updates our beliefs about the state of the world. However, ultimately our goal is to convert our beliefs into actions. In this section, we discuss the optimal way to do this.

We can formalize any given statistical decision problem as a game against nature (as opposed to a game against other strategic players, which is the topic of game theory, see e.g., (Shoham and Leyton-Brown 2009) for details). In this game, nature picks a state or parameter or label, $y \in \mathcal{Y}$, unknown to us, and then generates an observation, $\mathbf{x} \in \mathcal{X}$, which we get to see. We then have to make a decision, that is, we have to choose an action a from some **action space** $\mathcal{A}$. Finally we incur some **loss**, $L(y, a)$, which measures how compatible our action a is with nature's hidden state y. For example, we might use misclassification loss, $L(y, a) = \mathbb{I}(y \neq a)$, or squared loss, $L(y, a) = (y - a)^2$. We will see some other examples below.

Our goal is to devise a **decision procedure** or **policy**, $\delta : \mathcal{X} \to \mathcal{A}$, which specifies the optimal action for each possible input. By optimal, we mean the action that minimizes the expected loss:

$$\delta(\mathbf{x}) = \underset{a \in \mathcal{A}}{\operatorname{argmin}} \, \mathbb{E}\left[L(y, a)\right] \tag{5.96}$$

In economics, it is more common to talk of a **utility function**; this is just negative loss, $U(y, a) = -L(y, a)$. Thus the above rule becomes

$$\delta(\mathbf{x}) = \underset{a \in \mathcal{A}}{\operatorname{argmax}} \, \mathbb{E}\left[U(y, a)\right] \tag{5.97}$$

This is called the **maximum expected utility principle**, and is the essence of what we mean by **rational behavior**.

Note that there are two different interpretations of what we mean by "expected". In the Bayesian version, which we discuss below, we mean the expected value of y given the data we have seen so far. In the frequentist version, which we discuss in Section 6.3, we mean the expected value of y and $\mathbf{x}$ that we expect to see in the future.

In the Bayesian approach to decision theory, the optimal action, having observed $\mathbf{x}$, is defined as the action a that minimizes the **posterior expected loss**:

$$\rho(a|\mathbf{x}) \triangleq \mathbb{E}_{p(y|\mathbf{x})}\left[L(y, a)\right] = \sum_y L(y, a) p(y|\mathbf{x}) \tag{5.98}$$

(If y is continuous (e.g., when we want to estimate a parameter vector), we should replace the sum with an integral.) Hence the **Bayes estimator**, also called the **Bayes decision rule**, is given by

$$\delta(\mathbf{x}) = \arg\min_{a \in \mathcal{A}} \rho(\mathbf{a}|\mathbf{x}) \tag{5.99}$$

5.7.1 Bayes estimators for common loss functions

In this section we show how to construct Bayes estimators for the loss functions most commonly arising in machine learning.

5.7.1.1 MAP estimate minimizes 0-1 loss

The **0-1 loss** is defined by

$$L(y, a) = \mathbb{I}(y \neq a) = \begin{cases} 0 & \text{if } a = y \\ 1 & \text{if } a \neq y \end{cases} \tag{5.100}$$

This is commonly used in classification problems where y is the true class label and $a = \hat{y}$ is the estimate.

For example, in the two class case, we can write the loss matrix as follows:

	$\hat{y} = 1$	$\hat{y} = 0$
$y = 1$	0	1
$y = 0$	1	0

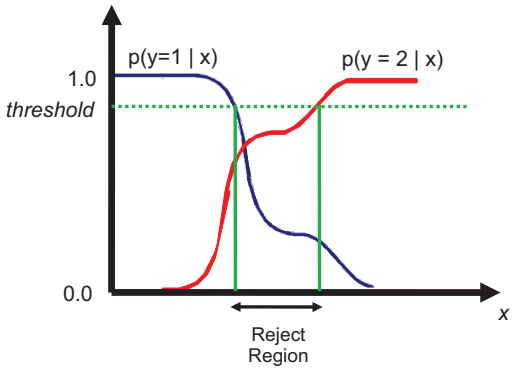


Figure 5.13 For some regions of input space, where the class posteriors are uncertain, we may prefer not to choose class 1 or 2; instead we may prefer the reject option. Based on Figure 1.26 of (Bishop 2006a).

(In Section 5.7.2, we generalize this loss function so it penalizes the two kinds of errors on the off-diagonal differently.)

The posterior expected loss is

$$\rho(a|\mathbf{x}) = p(a \neq y|\mathbf{x}) = 1 - p(y|\mathbf{x}) \tag{5.101}$$

Hence the action that minimizes the expected loss is the posterior mode or MAP estimate

$$y^*(\mathbf{x}) = \arg\max_{y \in \mathcal{Y}} p(y|\mathbf{x}) \tag{5.102}$$

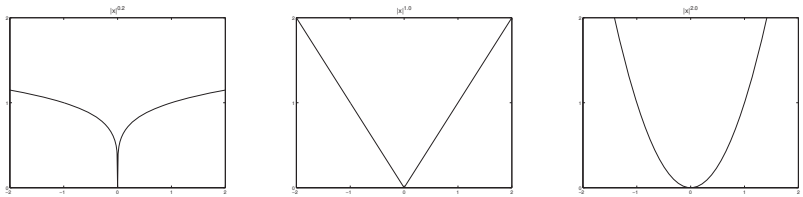
5.7.1.2 Reject option

In classification problems where $p(y|\mathbf{x})$ is very uncertain, we may prefer to choose a **reject action**, in which we refuse to classify the example as any of the specified classes, and instead say "don't know". Such ambiguous cases can be handled by e.g., a human expert. See Figure 5.13 for an illustration. This is useful in **risk averse** domains such as medicine and finance.

We can formalize the reject option as follows. Let choosing $a = C + 1$ correspond to picking the reject action, and choosing $a \in \{1, \ldots, C\}$ correspond to picking one of the classes. Suppose we define the loss function as

$$L(y = j, a = i) = \begin{cases} 0 & \text{if } i = j \text{ and } i, j \in \{1, \ldots, C\} \\ \lambda_r & \text{if } i = C + 1 \\ \lambda_s & \text{otherwise} \end{cases} \tag{5.103}$$

where λ_r is the cost of the reject action, and λ_s is the cost of a substitution error. In Exercise 5.3, you will show that the optimal action is to pick the reject action if the most probable class has a probability below $1 - \frac{\lambda_r}{\lambda_s}$; otherwise you should just pick the most probable class.



(a) (b) (c)

Figure 5.14 (a-c). Plots of the $L(y,a) = |y-a|^q$ vs $|y-a|$ for $q = 0.2$, $q = 1$ and $q = 2$. Figure generated by `lossFunctionFig`.

5.7.1.3 Posterior mean minimizes ℓ_2 (quadratic) loss

For continuous parameters, a more appropriate loss function is **squared error**, ℓ_2 **loss**, or **quadratic loss**, defined as

$$L(y,a) = (y-a)^2 \tag{5.104}$$

The posterior expected loss is given by

$$\rho(a|\mathbf{x}) \quad = \quad \mathbb{E}\left[(y-a)^2|\mathbf{x}\right] = \mathbb{E}\left[y^2|\mathbf{x}\right] - 2a\mathbb{E}\left[y|\mathbf{x}\right] + a^2 \tag{5.105}$$

Hence the optimal estimate is the posterior mean:

$$\frac{\partial}{\partial a}\rho(a|\mathbf{x}) = -2\mathbb{E}\left[y|\mathbf{x}\right] + 2a = 0 \;\Rightarrow\; \hat{y} = \mathbb{E}\left[y|\mathbf{x}\right] = \int yp(y|\mathbf{x})dy \tag{5.106}$$

This is often called the **minimum mean squared error** estimate or **MMSE** estimate.

In a linear regression problem, we have

$$p(y|\mathbf{x},\boldsymbol{\theta}) = \mathcal{N}(y|\mathbf{x}^T\mathbf{w},\sigma^2) \tag{5.107}$$

In this case, the optimal estimate given some training data $\mathcal{D}$ is given by

$$\mathbb{E}\left[y|\mathbf{x},\mathcal{D}\right] = \mathbf{x}^T\mathbb{E}\left[\mathbf{w}|\mathcal{D}\right] \tag{5.108}$$

That is, we just plug-in the posterior mean parameter estimate. Note that this is the optimal thing to do no matter what prior we use for $\mathbf{w}$.

5.7.1.4 Posterior median minimizes ℓ_1 (absolute) loss

The ℓ_2 loss penalizes deviations from the truth quadratically, and thus is sensitive to outliers. A more robust alternative is the absolute or ℓ_1 **loss**, $L(y,a) = |y-a|$ (see Figure 5.14). The optimal estimate is the posterior median, i.e., a value a such that $P(y < a|\mathbf{x}) = P(y \geq a|\mathbf{x}) = 0.5$. See Exercise 5.9 for a proof.

5.7.1.5 Supervised learning

Consider a prediction function $\delta : \mathcal{X} \to \mathcal{Y}$, and suppose we have some cost function $\ell(y,y')$ which gives the cost of predicting y' when the truth is y. We can define the loss incurred by

taking action δ (i.e., using this predictor) when the unknown state of nature is $\boldsymbol{\theta}$ (the parameters of the data generating mechanism) as follows:

$$L(\boldsymbol{\theta}, \delta) \triangleq \mathbb{E}_{(\mathbf{x},y)\sim p(\mathbf{x},y|\boldsymbol{\theta})}\left[\ell(y, \delta(\mathbf{x})\right] = \sum_{\mathbf{x}} \sum_{y} L(y, \delta(\mathbf{x}))p(\mathbf{x}, y|\boldsymbol{\theta}) \tag{5.109}$$

This is known as the **generalization error**. Our goal is to minimize the posterior expected loss, given by

$$\rho(\delta|\mathcal{D}) = \int p(\boldsymbol{\theta}|\mathcal{D})L(\boldsymbol{\theta}, \delta)d\boldsymbol{\theta} \tag{5.110}$$

This should be contrasted with the frequentist risk which is defined in Equation 6.47.

5.7.2 The false positive vs false negative tradeoff

In this section, we focus on binary decision problems, such as hypothesis testing, two-class classification, object/ event detection, etc. There are two types of error we can make: a **false positive** (aka **false alarm**), which arises when we estimate $\hat{y} = 1$ but the truth is $y = 0$; or a **false negative** (aka **missed detection**), which arises when we estimate $\hat{y} = 0$ but the truth is $y = 1$. The 0-1 loss treats these two kinds of errors equivalently. However, we can consider the following more general loss matrix:

	$\hat{y} = 1$	$\hat{y} = 0$
$y = 1$	0	L_{FN}
$y = 0$	L_{FP}	0

where L_{FN} is the cost of a false negative, and L_{FP} is the cost of a false positive. The posterior expected loss for the two possible actions is given by

$$\rho(\hat{y} = 0|\mathbf{x}) = L_{FN}\ p(y = 1|\mathbf{x}) \tag{5.111}$$

$$\rho(\hat{y} = 1|\mathbf{x}) = L_{FP}\ p(y = 0|\mathbf{x}) \tag{5.112}$$

Hence we should pick class $\hat{y} = 1$ iff

$$\rho(\hat{y} = 0|\mathbf{x}) > \rho(\hat{y} = 1|\mathbf{x}) \tag{5.113}$$

$$\frac{p(y = 1|\mathbf{x})}{p(y = 0|\mathbf{x})} > \frac{L_{FP}}{L_{FN}} \tag{5.114}$$

If $L_{FN} = cL_{FP}$, it is easy to show (Exercise 5.10) that we should pick $\hat{y} = 1$ iff $p(y = 1|\mathbf{x})/p(y = 0|\mathbf{x}) > \tau$, where $\tau = c/(1 + c)$ (see also (Muller et al. 2004)). For example, if a false negative costs twice as much as false positive, so $c = 2$, then we use a decision threshold of $2/3$ before declaring a positive.

Below we discuss ROC curves, which provide a way to study the FP-FN tradeoff without having to choose a specific threshold.

5.7.2.1 ROC curves and all that

Suppose we are solving a binary decision problem, such as classification, hypothesis testing, object detection, etc. Also, assume we have a labeled data set, $\mathcal{D} = \{(\mathbf{x}_i, y_i)\}$. Let $\delta(\mathbf{x}) =$

		Truth		
		1	0	Σ
Estimate	1	TP	FP	$\hat{N}_+ = TP + FP$
	0	FN	TN	$\hat{N}_- = FN + TN$
	Σ	$N_+ = TP + FN$	$N_- = FP + TN$	$N = TP + FP + FN + TN$

Table 5.2 Quantities derivable from a confusion matrix. N_+ is the true number of positives, $\hat{N}_+$ is the "called" number of positives, N_- is the true number of negatives, $\hat{N}_-$ is the "called" number of negatives.

	$y = 1$	$y = 0$
$\hat{y} = 1$	TP/N_+=TPR=sensitivity=recall	FP/N_-=FPR=type I
$\hat{y} = 0$	FN/N_+=FNR=miss rate=type II	TN/N_-=TNR=specifity

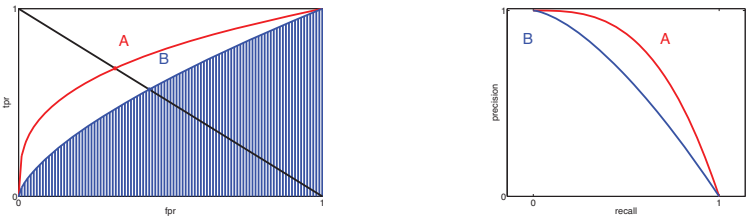
Table 5.3 Estimating $p(\hat{y}|y)$ from a confusion matrix. Abbreviations: FNR = false negative rate, FPR = false positive rate, TNR = true negative rate, TPR = true positive rate.

$\mathbb{I}(f(\mathbf{x}) > \tau)$ be our decision rule, where $f(\mathbf{x})$ is a measure of confidence that $y = 1$ (this should be monotonically related to $p(y = 1|\mathbf{x})$, but does not need to be a probability), and τ is some threshold parameter. For each given value of τ, we can apply our decision rule and count the number of true positives, false positives, true negatives, and false negatives that occur, as shown in Table 5.2. This table of errors is called a **confusion matrix**.

From this table, we can compute the **true positive rate** (TPR), also known as the **sensitivity**, **recall** or **hit rate**, by using $TPR = TP/N_+ \approx p(\hat{y} = 1|y = 1)$. We can also compute the **false positive rate** (FPR), also called the **false alarm rate**, or the **type I error rate**, by using $FPR = FP/N_- \approx p(\hat{y} = 1|y = 0)$. These and other definitions are summarized in Tables 5.3 and 5.4. We can combine these errors in any way we choose to compute a loss function.

However, rather than than computing the TPR and FPR for a fixed threshold τ, we can run our detector for a set of thresholds, and then plot the TPR vs FPR as an implicit function of τ. This is called a **receiver operating characteristic** or **ROC** curve. See Figure 5.15(a) for an example. Any system can achieve the point on the bottom left, $(FPR = 0, TPR = 0)$, by setting $\tau = 1$ and thus classifying everything as negative; similarly any system can achieve the point on the top right, $(FPR = 1, TPR = 1)$, by setting $\tau = 0$ and thus classifying everything as positive. If a system is performing at chance level, then we can achieve any point on the diagonal line $TPR = FPR$ by choosing an appropriate threshold. A system that perfectly separates the positives from negatives has a threshold that can achieve the top left corner, $(FPR = 0, TPR = 1)$; by varying the threshold such a system will "hug" the left axis and then the top axis, as shown in Figure 5.15(a).

The quality of a ROC curve is often summarized as a single number using the **area under the curve** or **AUC**. Higher AUC scores are better; the maximum is obviously 1. Another summary statistic that is used is the **equal error rate** or **EER**, also called the **cross over rate**, defined as the value which satisfies $FPR = FNR$. Since $FNR = 1 - TPR$, we can compute the EER by drawing a line from the top left to the bottom right and seeing where it intersects the ROC curve (see points A and B in Figure 5.15(a)). Lower EER scores are better; the minimum is obviously 0.



(a) (b)

Figure 5.15 (a) ROC curves for two hypothetical classification systems. A is better than B. We plot the true positive rate (TPR) vs the false positive rate (FPR) as we vary the threshold τ. We also indicate the equal error rate (EER) with the red and blue dots, and the area under the curve (AUC) for classifier B. (b) A precision-recall curve for two hypothetical classification systems. A is better than B. Figure generated by `PRhand`.

	$y=1$	$y=0$
$\hat{y}=1$	$TP/\hat{N}_+$=precision=PPV	$FP/\hat{N}_+$=FDP
$\hat{y}=0$	$FN/\hat{N}_-$	$TN/\hat{N}_-$=NPV

Table 5.4 Estimating $p(y|\hat{y})$ from a confusion matrix. Abbreviations: FDP = false discovery probability, NPV = negative predictive value, PPV = positive predictive value,

5.7.2.2 Precision recall curves

When trying to detect a rare event (such as retrieving a relevant document or finding a face in an image), the number of negatives is very large. Hence comparing $TPR = TP/N_+$ to $FPR = FP/N_-$ is not very informative, since the FPR will be very small. Hence all the "action" in the ROC curve will occur on the extreme left. In such cases, it is common to plot the TPR versus the number of false positives, rather than vs the false positive rate.

However, in some cases, the very notion of "negative" is not well-defined. For example, when detecting objects in images (see Section 1.2.1.3), if the detector works by classifying patches, then the number of patches examined — and hence the number of true negatives — is a parameter of the algorithm, not part of the problem definition. So we would like to use a measure that only talks about positives.

The **precision** is defined as $TP/\hat{N}_+ = p(y=1|\hat{y}=1)$ and the **recall** is defined as $TP/N_+ = p(\hat{y}=1|y=1)$. Precision measures what fraction of our detections are actually positive, and recall measures what fraction of the positives we actually detected. If $\hat{y}_i \in \{0,1\}$ is the predicted label, and $y_i \in \{0,1\}$ is the true label, we can estimate precision and recall using

$$P = \frac{\sum_i y_i \hat{y}_i}{\sum_i \hat{y}_i}, \; R = \frac{\sum_i y_i \hat{y}_i}{\sum_i y_i} \tag{5.115}$$

A **precision recall curve** is a plot of precision vs recall as we vary the threshold τ. See Figure 5.15(b). Hugging the top right is the best one can do.

This curve can be summarized as a single number using the mean precision (averaging over

	Class 1	
	$y=1$	$y=0$
$\hat{y}=1$	10	10
$\hat{y}=0$	10	970

	Class 2	
	$y=1$	$y=0$
$\hat{y}=1$	90	10
$\hat{y}=0$	10	890

	Pooled	
	$y=1$	$y=0$
$\hat{y}=1$	100	20
$\hat{y}=0$	20	1860

Table 5.5 Illustration of the difference between macro- and micro-averaging. y is the true label, and $\hat{y}$ is the called label. In this example, the macro-averaged precision is $[10/(10+10)+90/(10+90)]/2 = (0.5+0.9)/2 = 0.7$. The micro-averaged precision is $100/(100+20) \approx 0.83$. Based on Table 13.7 of (Manning et al. 2008).

recall values), which approximates the area under the curve. Alternatively, one can quote the precision for a fixed recall level, such as the precision of the first $K = 10$ entities recalled. This is called the **average precision at K** score. This measure is widely used when evaluating information retrieval systems.

5.7.2.3 F-scores *

For a fixed threshold, one can compute a single precision and recall value. These are often combined into a single statistic called the **F score**, or **F1 score**, which is the harmonic mean of precision and recall:

$$F_1 \triangleq \frac{2}{1/P + 1/R} = \frac{2PR}{R+P} \tag{5.116}$$

Using Equation 5.115, we can write this as

$$F_1 = \frac{2\sum_{i=1}^{N} y_i \hat{y}_i}{\sum_{i=1}^{N} y_i + \sum_{i=1}^{N} \hat{y}_i} \tag{5.117}$$

This is a widely used measure in information retrieval systems.

To understand why we use the harmonic mean instead of the arithmetic mean, $(P+R)/2$, consider the following scenario. Suppose we recall all entries, so $R = 1$. The precision will be given by the **prevalence**, $p(y=1)$. Suppose the prevalence is low, say $p(y=1) = 10^{-4}$. The arithmetic mean of P and R is given by $(P+R)/2 = (10^{-4}+1)/2 \approx 50\%$. By contrast, the harmonic mean of this strategy is only $\frac{2\times 10^{-4}\times 1}{1+10^{-4}} \approx 0.2\%$.

In the multi-class case (e.g., for document classification problems), there are two ways to generalize F_1 scores. The first is called **macro-averaged F1**, and is defined as $\sum_{c=1}^{C} F_1(c)/C$, where $F_1(c)$ is the F_1 score obtained on the task of distinguishing class c from all the others. The other is called **micro-averaged F1**, and is defined as the F_1 score where we pool all the counts from each class's contingency table.

Table 5.5 gives a worked example that illustrates the difference. We see that the precision of class 1 is 0.5, and of class 2 is 0.9. The macro-averaged precision is therefore 0.7, whereas the micro-averaged precision is 0.83. The latter is much closer to the precision of class 2 than to the precision of class 1, since class 2 is five times larger than class 1. To give equal weight to each class, use macro-averaging.

5.7.2.4 False discovery rates *

Suppose we are trying to discover a rare phenomenon using some kind of high throughput measurement device, such as a gene expression micro array, or a radio telescope. We will need to make many binary decisions of the form $p(y_i = 1|\mathcal{D}) > \tau$, where $\mathcal{D} = \{\mathbf{x}_i\}_{i=1}^N$ and N may be large. This is called **multiple hypothesis testing**. Note that the difference from standard binary classification is that we are classifying y_i based on all the data, not just based on $\mathbf{x}_i$. So this is a simultaneous classification problem, where we might hope to do better than a series of individual classification problems.

How should we set the threshold τ? A natural approach is to try to minimize the expected number of false positives. In the Bayesian approach, this can be computed as follows:

$$FD(\tau, \mathcal{D}) \triangleq \sum_i \underbrace{(1 - p_i)}_{\text{pr. error}} \underbrace{\mathbb{I}(p_i > \tau)}_{\text{discovery}} \tag{5.118}$$

where $p_i \triangleq p(y_i = 1|\mathcal{D})$ is your belief that this object exhibits the phenomenon in question. We then define the posterior expected **false discovery rate** as follows:

$$FDR(\tau, \mathcal{D}) \triangleq FD(\tau, \mathcal{D})/N(\tau, \mathcal{D}) \tag{5.119}$$

where $N(\tau, \mathcal{D}) = \sum_i \mathbb{I}(p_i > \tau)$ is the number of discovered items. Given a desired FDR tolerance, say $\alpha = 0.05$, one can then adapt τ to achieve this; this is called the **direct posterior probability approach** to controlling the FDR (Newton et al. 2004; Muller et al. 2004).

In order to control the FDR it is very helpful to estimate the p_i's jointly (e.g., using a hierarchical Bayesian model, as in Section 5.5), rather than independently. This allows the pooling of statistical strength, and thus lower FDR. See e.g., (Berry and Hochberg 1999) for more information.

5.7.3 Other topics *

In this section, we briefly mention a few other topics related to Bayesian decision theory. We do not have space to go into detail, but we include pointers to the relevant literature.

5.7.3.1 Contextual bandits

A **one-armed bandit** is a colloquial term for a slot machine, found in casinos around the world. The game is this: you insert some money, pull an arm, and wait for the machine to stop; if you're lucky, you win some money. Now imagine there is a bank of K such machines to choose from. Which one should you use? This is called a **multi-armed bandit**, and can be modeled using Bayesian decision theory: there are K possible actions, and each action has an unknown reward (payoff function) r_k. By maintaining a belief state, $p(r_{1:K}|\mathcal{D}) = \prod_k p(r_k|\mathcal{D})$, one can devise an optimal policy; this can be compiled into a series of **Gittins Indices** (Gittins 1989). This optimally solves the **exploration-exploitation** tradeoff, which specifies how many times one should try each action before deciding to go with the winner.

Now consider an extension where each arm, and the player, has an associated feature vector; call all these features $\mathbf{x}$. This is called a **contextual bandit** (see e.g., (Sarkar 1991; Scott 2010; Li et al. 2011)). For example, the "arms" could represent ads or news articles which we want to show to the user, and the features could represent properties of these ads or articles, such

as a bag of words, as well as properties of the user, such as demographics. If we assume a linear model for reward, $r_k = \boldsymbol{\theta}_k^T \mathbf{x}$, we can maintain a distribution over the parameters of each arm, $p(\boldsymbol{\theta}_k|\mathcal{D})$, where $\mathcal{D}$ is a series of tuples of the form $(a, \mathbf{x}, r)$, which specifies which arm was pulled, what its features were, and what the resulting outcome was (e.g., $r = 1$ if the user clicked on the ad, and $r = 0$ otherwise). We discuss ways to compute $p(\boldsymbol{\theta}_k|\mathcal{D})$ from linear and logistic regression models in later chapters.

Given the posterior, we must decide what action to take. One common heuristic, known as **UCB** (which stands for "upper confidence bound") is to take the action which maximizes

$$k^* = \underset{k=1}{\overset{K}{\operatorname{argmax}}} \mu_k + \lambda \sigma_k \tag{5.120}$$

where $\mu_k = \mathbb{E}[r_k|\mathcal{D}]$, $\sigma_k^2 = \operatorname{var}[r_k|\mathcal{D}]$ and λ is a tuning parameter that trades off exploration and exploitation. The intuition is that we should pick actions about which we believe are good (μ_k is large), and/ or actions about which we are uncertain (σ_k is large).

An even simpler method, known as **Thompson sampling**, is as follows. At each step, we pick action k with a probability that is equal to its probability of being the optimal action:

$$p_k = \int \mathbb{I}(\mathbb{E}[r|a, \mathbf{x}, \boldsymbol{\theta}] = \max_{a'} \mathbb{E}[r|a', \mathbf{x}, \boldsymbol{\theta}]) p(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta} \tag{5.121}$$

We can approximate this by drawing a single sample from the posterior, $\boldsymbol{\theta}^t \sim p(\boldsymbol{\theta}|\mathcal{D})$, and then choosing $k^* = \operatorname{argmax}_k \mathbb{E}\left[r|\mathbf{x}, k, \boldsymbol{\theta}^t\right]$. Despite its simplicity, this has been shown to work quite well (Chapelle and Li 2011).

5.7.3.2 Utility theory

Suppose we are a doctor trying to decide whether to operate on a patient or not. We imagine there are 3 states of nature: the patient has no cancer, the patient has lung cancer, or the patient has breast cancer. Since the action and state space is discrete, we can represent the loss function $L(\theta, a)$ as a **loss matrix**, such as the following:

	Surgery	No surgery
No cancer	20	0
Lung cancer	10	50
Breast cancer	10	60

These numbers reflects the fact that not performing surgery when the patient has cancer is very bad (loss of 50 or 60, depending on the type of cancer), since the patient might die; not performing surgery when the patient does not have cancer incurs no loss (0); performing surgery when the patient does not have cancer is wasteful (loss of 20); and performing surgery when the patient does have cancer is painful but necessary (10).

It is natural to ask where these numbers come from. Ultimately they represent the personal **preferences** or values of a fictitious doctor, and are somewhat arbitrary: just as some people prefer chocolate ice cream and others prefer vanilla, there is no such thing as the "right" loss/ utility function. However, it can be shown (see e.g., (DeGroot 1970)) that any set of consistent preferences can be converted to a scalar loss/ utility function. Note that utility can be measured on an arbitrary scale, such as dollars, since it is only relative values that matter.[6]

6. People are often squeamish about talking about human lives in monetary terms, but all decision making requires

5.7.3.3 Sequential decision theory

So far, we have concentrated on **one-shot decision problem**s, where we only have to make one decision and then the game ends. In Setion 10.6, we will generalize this to multi-stage or sequential decision problems. Such problems frequently arise in many business and engineering settings. This is closely related to the problem of reinforcement learning. However, further discussion of this point is beyond the scope of this book.

Exercises

Exercise 5.1 Proof that a mixture of conjugate priors is indeed conjugate

Derive Equation 5.69.

Exercise 5.2 Optimal threshold on classification probability

Consider a case where we have learned a conditional probability distribution $P(y|\mathbf{x})$. Suppose there are only two classes, and let $p_0 = P(Y = 0|\mathbf{x})$ and $p_1 = P(Y = 1|\mathbf{x})$. Consider the loss matrix below:

predicted label $\hat{y}$	true label y	
	0	1
0	0	λ_{01}
1	λ_{10}	0

a. Show that the decision $\hat{y}$ that minimizes the expected loss is equivalent to setting a probability threshold θ and predicting $\hat{y} = 0$ if $p_1 < \theta$ and $\hat{y} = 1$ if $p_1 \geq \theta$. What is θ as a function of λ_{01} and λ_{10}? (Show your work.)

b. Show a loss matrix where the threshold is 0.1. (Show your work.)

Exercise 5.3 Reject option in classifiers

(Source: (Duda et al. 2001, Q2.13).)

In many classification problems one has the option either of assigning $\mathbf{x}$ to class j or, if you are too uncertain, of choosing the **reject option**. If the cost for rejects is less than the cost of falsely classifying the object, it may be the optimal action. Let α_i mean you choose action i, for $i = 1 : C + 1$, where C is the number of classes and $C + 1$ is the reject action. Let $Y = j$ be the true (but unknown) **state of nature**. Define the loss function as follows

$$\lambda(\alpha_i|Y = j) = \begin{cases} 0 & \text{if } i = j \text{ and } i, j \in \{1, \ldots, C\} \\ \lambda_r & \text{if } i = C + 1 \\ \lambda_s & \text{otherwise} \end{cases} \tag{5.122}$$

In otherwords, you incur 0 loss if you correctly classify, you incur λ_r loss (cost) if you choose the reject option, and you incur λ_s loss (cost) if you make a substitution error (misclassification).

---

tradeoffs, and one needs to use some kind of "currency" to compare different courses of action. Insurance companies do this all the time. Ross Schachter, a decision theorist at Stanford University, likes to tell a story of a school board who rejected a study on absestos removal from schools because it performed a **cost-benefit analysis**, which was considered "inhumane" because they put a dollar value on children's health; the result of rejecting the report was that the absestos was not removed, which is surely more "inhumane". In medical domains, one often measures utility in terms of **QALY**, or quality-adjusted life-years, instead of dollars, but it's the same idea. Of course, even if you do not explicitly specify how much you value different people's lives, your *behavior* will reveal your implicit values/ preferences, and these preferences can then be converted to a real-valued scale, such as dollars or QALY. Inferring a utility function from behavior is called **inverse reinforcement learning**.

Decision	true label y	
$\hat{y}$	0	1
predict 0	0	10
predict 1	10	0
reject	3	3

a. Show that the minimum risk is obtained if we decide $Y = j$ if $p(Y = j|\mathbf{x}) \geq p(Y = k|\mathbf{x})$ for all k (i.e., j is the most probable class) *and* if $p(Y = j|\mathbf{x}) \geq 1 - \frac{\lambda_r}{\lambda_s}$; otherwise we decide to reject.

b. Describe qualitatively what happens as λ_r/λ_s is increased from 0 to 1 (i.e., the relative cost of rejection increases).

Exercise 5.4 More reject options

In many applications, the classifier is allowed to "reject" a test example rather than classifying it into one of the classes. Consider, for example, a case in which the cost of a misclassification is \$10 but the cost of having a human manually make the decison is only \$3. We can formulate this as the following loss matrix:

a. Suppose $P(y = 1|\mathbf{x})$ is predicted to be 0.2. Which decision minimizes the expected loss?

b. Now suppose $P(y = 1|\mathbf{x})$=0.4. Now which decision minimizes the expected loss?

c. Show that in general, for this loss matrix, but for any posterior distribution, there will be two thresholds θ_0 and θ_1 such that the optimal decisionn is to predict 0 if $p_1 < \theta_0$, reject if $\theta_0 \leq p_1 \leq \theta_1$, and predict 1 if $p_1 > \theta_1$ (where $p_1 = p(y = 1|\mathbf{x})$). What are these thresholds?

Exercise 5.5 Newsvendor problem

Consider the following classic problem in decision theory/ economics. Suppose you are trying to decide how much quantity Q of some product (e.g., newspapers) to buy to maximize your profits. The optimal amount will depend on how much demand D you think there is for your product, as well as its cost to you C and its selling price P. Suppose D is unknown but has pdf $f(D)$ and cdf $F(D)$. We can evaluate the expected profit by considering two cases: if $D > Q$, then we sell all Q items, and make profit $\pi = (P - C)Q$; but if $D < Q$, we only sell D items, at profit $(P - C)D$, but have wasted $C(Q - D)$ on the unsold items. So the expected profit if we buy quantity Q is

$$E\pi(Q) = \int_Q^\infty (P - C)Qf(D)dD + \int_0^Q (P - C)Df(D) - \int_0^Q C(Q - D)f(D)dD \tag{5.123}$$

Simplify this expression, and then take derivatives wrt Q to show that the optimal quantity Q^* (which maximizes the expected profit) satisfies

$$F(Q^*) = \frac{P - C}{P} \tag{5.124}$$

Exercise 5.6 Bayes factors and ROC curves

Let $B = p(D|H_1)/p(D|H_0)$ be the bayes factor in favor of model 1. Suppose we plot two ROC curves, one computed by thresholding B, and the other computed by thresholding $p(H_1|D)$. Will they be the same or different? Explain why.

Exercise 5.7 Bayes model averaging helps predictive accuracy

Let Δ be a quantity that we want to predict, let $\mathcal{D}$ be the observed data and $\mathcal{M}$ be a finite set of models. Suppose our action is to provide a probabilistic prediction $p()$, and the loss function is $L(\Delta, p()) =$

$-\log p(\Delta)$. We can either perform Bayes model averaging and predict using

$$p^{BMA}(\Delta) = \sum_{m \in \mathcal{M}} p(\Delta|m, \mathcal{D}) p(m|\mathcal{D}) \tag{5.125}$$

or we could predict using any single model (a plugin approximation)

$$p^m(\Delta) = p(\Delta|m, \mathcal{D}) \tag{5.126}$$

Show that, for all models $m \in \mathcal{M}$, the posterior expected loss using BMA is lower, i.e.,

$$\mathbb{E}\left[L(\Delta, p^{BMA})\right] \leq \mathbb{E}\left[L(\Delta, p^m)\right] \tag{5.127}$$

where the expectation over Δ is with respect to

$$p(\Delta|\mathcal{D}) = \sum_{m \in \mathcal{M}} p(\Delta|m, \mathcal{D}) p(m|\mathcal{D}) \tag{5.128}$$

Hint: use the non-negativity of the KL divergence.

Exercise 5.8 MLE and model selection for a 2d discrete distribution

(Source: Jaakkola.)

Let $x \in \{0, 1\}$ denote the result of a coin toss ($x = 0$ for tails, $x = 1$ for heads). The coin is potentially biased, so that heads occurs with probability θ_1. Suppose that someone else observes the coin flip and reports to you the outcome, y. But this person is unreliable and only reports the result correctly with probability θ_2; i.e., $p(y|x, \theta_2)$ is given by

	$y = 0$	$y = 1$
$x = 0$	θ_2	$1 - \theta_2$
$x = 1$	$1 - \theta_2$	θ_2

Assume that θ_2 is independent of x and θ_1.

a. Write down the joint probability distribution $p(x, y|\boldsymbol{\theta})$ as a 2×2 table, in terms of $\boldsymbol{\theta} = (\theta_1, \theta_2)$.

b. Suppose have the following dataset: $\mathbf{x} = (1, 1, 0, 1, 1, 0, 0)$, $\mathbf{y} = (1, 0, 0, 0, 1, 0, 1)$. What are the MLEs for θ_1 and θ_2? Justify your answer. Hint: note that the likelihood function factorizes,

$$p(x, y|\boldsymbol{\theta}) = p(y|x, \theta_2) p(x|\theta_1) \tag{5.129}$$

What is $p(\mathcal{D}|\hat{\boldsymbol{\theta}}, M_2)$ where M_2 denotes this 2-parameter model? (You may leave your answer in fractional form if you wish.)

c. Now consider a model with 4 parameters, $\boldsymbol{\theta} = (\theta_{0,0}, \theta_{0,1}, \theta_{1,0}, \theta_{1,1})$, representing $p(x, y|\boldsymbol{\theta}) = \theta_{x,y}$. (Only 3 of these parameters are free to vary, since they must sum to one.) What is the MLE of $\boldsymbol{\theta}$? What is $p(\mathcal{D}|\hat{\boldsymbol{\theta}}, M_4)$ where M_4 denotes this 4-parameter model?

d. Suppose we are not sure which model is correct. We compute the leave-one-out cross validated log likelihood of the 2-parameter model and the 4-parameter model as follows:

$$L(m) = \sum_{i=1}^{n} \log p(x_i, y_i|m, \hat{\theta}(\mathcal{D}_{-i})) \tag{5.130}$$

and $\hat{\theta}(\mathcal{D}_{-i}))$ denotes the MLE computed on $\mathcal{D}$ excluding row i. Which model will CV pick and why? Hint: notice how the table of counts changes when you omit each training case one at a time.

e. Recall that an alternative to CV is to use the BIC score, defined as

$$\text{BIC}(M, \mathcal{D}) \triangleq \log p(\mathcal{D}|\hat{\boldsymbol{\theta}}_{MLE}) - \frac{\text{dof}(M)}{2} \log N \tag{5.131}$$

where $\text{dof}(M)$ is the number of free parameters in the model, Compute the BIC scores for both models (use log base e). Which model does BIC prefer?

Exercise 5.9 Posterior median is optimal estimate under L1 loss

Prove that the posterior median is optimal estimate under L1 loss.

Exercise 5.10 Decision rule for trading off FPs and FNs

If $L_{FN} = cL_{FP}$, show that we should pick $\hat{y} = 1$ iff $p(y = 1|\mathbf{x})/p(y = 0|\mathbf{x}) > \tau$, where $\tau = c/(1 + c)$

6 *Frequentist statistics*

6.1 Introduction

The approach to statistical inference that we described in Chapter 5 is known as Bayesian statistics. Perhaps surprisingly, this is considered controversial by some people, whereas the application of Bayes rule to non-statistical problems — such as medical diagnosis (Section 2.2.3.1), spam filtering (Section 3.4.4.1), or airplane tracking (Section 18.2.1) — is not controversial. The reason for the objection has to do with a misguided distinction between parameters of a statistical model and other kinds of unknown quantities.[1]

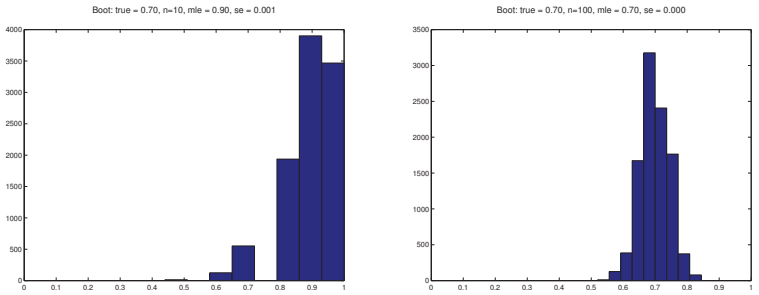
Attempts have been made to devise approaches to statistical inference that avoid treating parameters like random variables, and which thus avoid the use of priors and Bayes rule. Such approaches are known as **frequentist statistics**, **classical statistics** or **orthodox statistics**. Instead of being based on the posterior distribution, they are based on the concept of a sampling distribution. This is the distribution that an estimator has when applied to multiple data sets sampled from the true but unknown distribution; see Section 6.2 for details. It is this notion of variation across repeated trials that forms the basis for modeling uncertainty used by the frequentist approach.

By contrast, in the Bayesian approach, we only ever condition on the actually observed data; there is no notion of repeated trials. This allows the Bayesian to compute the probability of one-off events, as we discussed in Section 2.1. Perhaps more importantly, the Bayesian approach avoids certain paradoxes that plague the frequentist approach (see Section 6.6). Nevertheless, it is important to be familiar with frequentist statistics (especially Section 6.5), since it is widely used in machine learning.

6.2 Sampling distribution of an estimator

In frequentist statistics, a parameter estimate $\hat{\boldsymbol{\theta}}$ is computed by applying an **estimator** δ to some data $\mathcal{D}$, so $\hat{\boldsymbol{\theta}} = \delta(\mathcal{D})$. The parameter is viewed as fixed and the data as random, which is the exact opposite of the Bayesian approach. The uncertainty in the parameter estimate can be measured by computing the **sampling distribution** of the estimator. To understand this

1. Parameters are sometimes considered to represent true (but unknown) physical quantities, which are therefore not random. However, we have seen that it is perfectly reasonable to use a probability distribution to represent one's uncertainty about an unknown constant.



(a) (b)

Figure 6.1 A bootstrap approximation to the sampling distribution of $\hat{\theta}$ for a Bernoulli distribution. We use $B = 10,000$ bootstrap samples. The N datacases were generated from $\text{Ber}(\theta = 0.7)$. (a) MLE with $N = 10$. (b) MLE with $N = 100$. Figure generated by `bootstrapDemoBer`.

concept, imagine sampling many different data sets $\mathcal{D}^{(s)}$ from some true model, $p(\cdot|\theta^*)$, i.e., let $\mathcal{D}^{(s)} = \{x_i^{(s)}\}_{i=1}^N$, where $x_i^s \sim p(\cdot|\theta^*)$, and θ^* is the true parameter. Here $s = 1 : S$ indexes the sampled data set, and N is the size of each such dataset. Now apply the estimator $\hat{\theta}(\cdot)$ to each $\mathcal{D}^{(s)}$ to get a set of estimates, $\{\hat{\theta}(\mathcal{D}^{(s)})\}$. As we let $S \to \infty$, the distribution induced on $\hat{\theta}(\cdot)$ is the sampling distribution of the estimator. We will discuss various ways to use the sampling distribution in later sections. But first we sketch two approaches for computing the sampling distribution itself.

6.2.1 Bootstrap

The **bootstrap** is a simple Monte Carlo technique to approximate the sampling distribution. This is particularly useful in cases where the estimator is a complex function of the true parameters.

The idea is simple. If we knew the true parameters θ^*, we could generate many (say S) fake datasets, each of size N, from the true distribution, $x_i^s \sim p(\cdot|\theta^*)$, for $s = 1 : S$, $i = 1 : N$. We could then compute our estimator from each sample, $\hat{\theta}^s = f(x_{1:N}^s)$ and use the empirical distribution of the resulting samples as our estimate of the sampling distribution. Since θ is unknown, the idea of the **parametric bootstrap** is to generate the samples using $\hat{\theta}(\mathcal{D})$ instead. An alternative, called the **non-parametric bootstrap**, is to sample the x_i^s (with replacement) from the original data $\mathcal{D}$, and then compute the induced distribution as before. Some methods for speeding up the bootstrap when applied to massive data sets are discussed in (Kleiner et al. 2011).

Figure 6.1 shows an example where we compute the sampling distribution of the MLE for a Bernoulli using the parametric bootstrap. (Results using the non-parametric bootstrap are essentially the same.) We see that the sampling distribution is asymmetric, and therefore quite far from Gaussian, when $N = 10$; when $N = 100$, the distribution looks more Gaussian, as theory suggests (see below).

A natural question is: what is the connection between the parameter estimates $\hat{\theta}^s = \hat{\theta}(x_{1:N}^s)$ computed by the bootstrap and parameter values sampled from the posterior, $\theta^s \sim p(\cdot|\mathcal{D})$?

Conceptually they are quite different. But in the common case that that the prior is not very strong, they can be quite similar. For example, Figure 6.1(c-d) shows an example where we compute the posterior using a uniform Beta(1,1) prior, and then sample from it. We see that the posterior and the sampling distribution are quite similar. So one can think of the bootstrap distribution as a "poor man's" posterior; see (Hastie et al. 2001, p235) for details.

However, perhaps surprisingly, bootstrap can be slower than posterior sampling. The reason is that the bootstrap has to fit the model S times, whereas in posterior sampling, we usually only fit the model once (to find a local mode), and then perform local exploration around the mode. Such local exploration is usually much faster than fitting a model from scratch.

6.2.2 Large sample theory for the MLE *

In some cases, the sampling distribution for some estimators can be computed analytically. In particular, it can be shown that, under certain conditions, as the sample size tends to infinity, the sampling distribution of the MLE becomes Gaussian. Informally, the requirement for this result to hold is that each parameter in the model gets to "see" an infinite amount of data, and that the model be identifiable. Unfortunately this excludes many of the models of interest to machine learning. Nevertheless, let us assume we are in a simple setting where the theorem holds.

The center of the Gaussian will be the MLE $\hat{\boldsymbol{\theta}}$. But what about the variance of this Gaussian? Intuitively the variance of the estimator will be (inversely) related to the amount of curvature of the likelihood surface at its peak. If the curvature is large, the peak will be "sharp", and the variance low; in this case, the estimate is "well determined". By contrast, if the curvature is small, the peak will be nearly "flat", so the variance is high.

Let us now formalize this intuition. Define the **score function** as the gradient of the log likelihood evaluated at some point $\hat{\boldsymbol{\theta}}$:

$$\mathbf{s}(\hat{\boldsymbol{\theta}}) \triangleq \nabla \log p(\mathcal{D}|\boldsymbol{\theta})|_{\hat{\boldsymbol{\theta}}} \tag{6.1}$$

Define the **observed information matrix** as the gradient of the negative score function, or equivalently, the Hessian of the NLL:

$$\mathbf{J}(\hat{\boldsymbol{\theta}}(\mathcal{D})) \triangleq -\nabla \mathbf{s}(\hat{\boldsymbol{\theta}}) = -\nabla^2_{\boldsymbol{\theta}} \log p(\mathcal{D}|\boldsymbol{\theta})|_{\hat{\boldsymbol{\theta}}} \tag{6.2}$$

In 1D, this becomes

$$J(\hat{\theta}(\mathcal{D})) = -\frac{d}{d\theta^2} \log p(\mathcal{D}|\theta)|_{\hat{\theta}} \tag{6.3}$$

This is just a measure of curvature of the log-likelihood function at $\hat{\theta}$.

Since we are studying the sampling distribution, $\mathcal{D} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ is a set of random variables. The **Fisher information matrix** is defined to be the expected value of the observed information matrix:[2]

$$\mathbf{I}_N(\hat{\boldsymbol{\theta}}|\boldsymbol{\theta}^*) \triangleq \mathbb{E}_{\boldsymbol{\theta}^*}\left[\mathbf{J}(\hat{\boldsymbol{\theta}}|\mathcal{D})\right] \tag{6.4}$$

2. This is not the usual definition, but is equivalent to it under standard assumptions. More precisely, the standard definition is as follows (we just give the scalar case to simplify notation): $I(\hat{\theta}|\theta^*) \triangleq \mathrm{var}_{\theta^*}\left[\frac{d}{d\theta} \log p(X|\theta)|_{\hat{\theta}}\right]$, that is, the variance of the score function. If $\hat{\theta}$ is the MLE, it is easy to see that $\mathbb{E}_{\theta^*}\left[\frac{d}{d\theta} \log p(X|\theta)|_{\hat{\theta}}\right] = 0$ (since

where $\mathbb{E}_{\boldsymbol{\theta}^*}[\mathbf{f}(\mathcal{D})] \triangleq \frac{1}{N}\sum_{i=1}^{N}\mathbf{f}(\mathbf{x}_i)p(\mathbf{x}_i|\boldsymbol{\theta}^*)$ is the expected value of the function $\mathbf{f}$ when applied to data sampled from $\boldsymbol{\theta}^*$. Often $\boldsymbol{\theta}^*$, representing the "true parameter" that generated the data, is assumed known, so we just write $\mathbf{I}_N(\hat{\boldsymbol{\theta}}) \triangleq \mathbf{I}_N(\hat{\boldsymbol{\theta}}|\boldsymbol{\theta}^*)$ for short. Furthermore, it is easy to see that $\mathbf{I}_N(\hat{\boldsymbol{\theta}}) = N\mathbf{I}_1(\hat{\boldsymbol{\theta}})$, because the log-likelihood for a sample of size N is just N times "steeper" than the log-likelihood for a sample of size 1. So we can drop the 1 subscript and just write $\mathbf{I}(\hat{\boldsymbol{\theta}}) \triangleq \mathbf{I}_1(\hat{\boldsymbol{\theta}})$. This is the notation that is usually used.

Now let $\hat{\boldsymbol{\theta}} \triangleq \hat{\boldsymbol{\theta}}_{mle}(\mathcal{D})$ be the MLE, where $\mathcal{D} \sim \boldsymbol{\theta}^*$. It can be shown that

$$\hat{\boldsymbol{\theta}} \rightarrow \mathcal{N}(\boldsymbol{\theta}^*, \mathbf{I}_N(\boldsymbol{\theta}^*)^{-1}) \tag{6.5}$$

as $N \rightarrow \infty$ (see e.g., (Rice 1995, p265) for a proof). We say that the sampling distribution of the MLE is **asymptotically normal**.

What about the variance of the MLE, which can be used as some measure of confidence in the MLE? Unfortunately, $\boldsymbol{\theta}^*$ is unknown, so we can't evaluate the variance of the sampling distribution. However, we can approximate the sampling distribution by replacing $\boldsymbol{\theta}^*$ with $\hat{\boldsymbol{\theta}}$. Consequently, the approximate **standard errors** of $\hat{\theta}_k$ are given by

$$\hat{se}_k \triangleq \mathbf{I}_N(\hat{\boldsymbol{\theta}})_{kk}^{-\frac{1}{2}} \tag{6.6}$$

For example, from Equation 5.60 we know that the Fisher information for a binomial sampling model is

$$I(\theta) = \frac{1}{\theta(1-\theta)} \tag{6.7}$$

So the approximate standard error of the MLE is

$$\hat{se} = \frac{1}{\sqrt{I_N(\hat{\theta})}} = \frac{1}{\sqrt{NI(\hat{\theta})}} = \left(\frac{\hat{\theta}(1-\hat{\theta})}{N}\right)^{\frac{1}{2}} \tag{6.8}$$

where $\hat{\theta} = \frac{1}{N}\sum_i X_i$. Compare this to Equation 3.27, which is the posterior standard deviation under a uniform prior.

6.3 Frequentist decision theory

In frequentist or classical decision theory, there is a loss function and a likelihood, but there is no prior and hence no posterior or posterior expected loss. Thus there is no automatic way of deriving an optimal estimator, unlike the Bayesian case. Instead, in the frequentist approach, we are free to choose any estimator or decision procedure $\delta : \mathcal{X} \rightarrow \mathcal{A}$ we want.[3]

---

the gradient must be zero at a maximum), so the variance reduces to the expected square of the score function: $I(\hat{\theta}|\theta^*) = \mathbb{E}_{\theta^*}\left[(\frac{d}{d\theta}\log p(X|\theta))^2\right]$. It can be shown (e.g., (Rice 1995, p263)) that $\mathbb{E}_{\theta^*}\left[(\frac{d}{d\theta}\log p(X|\theta))^2\right] = -\mathbb{E}_{\theta^*}\left[\frac{d^2}{d\theta^2}\log p(X|\theta)\right]$, so now the Fisher information reduces to the expected second derivative of the NLL, which is a much more intuitive quantity than the variance of the score.

3. In practice, the frequentist approach is usually only applied to one-shot statistical decision problems — such as classification, regression and parameter estimation — since its non-constructive nature makes it difficult to apply to sequential decision problems, which adapt to data online.

Having chosen an estimator, we define its expected loss or **risk** as follows:

$$R(\theta^*, \delta) \triangleq \mathbb{E}_{p(\tilde{\mathcal{D}}|\theta^*)}\left[L(\theta^*, \delta(\tilde{\mathcal{D}}))\right] = \int L(\theta^*, \delta(\tilde{\mathcal{D}}))p(\tilde{\mathcal{D}}|\theta^*)d\tilde{\mathcal{D}} \tag{6.9}$$

where $\tilde{\mathcal{D}}$ is data sampled from "nature's distribution", which is represented by parameter θ^*. In other words, the expectation is wrt the sampling distribution of the estimator. Compare this to the Bayesian posterior expected loss:

$$\rho(a|\mathcal{D}, \pi) \triangleq \mathbb{E}_{p(\theta|\mathcal{D},\pi)}\left[L(\theta, a)\right] = \int_{\Theta} L(\theta, \mathbf{a})p(\theta|\mathcal{D}, \pi)d\theta \tag{6.10}$$

We see that the Bayesian approach averages over θ (which is unknown) and conditions on $\mathcal{D}$ (which is known), whereas the frequentist approach averages over $\tilde{\mathcal{D}}$ (thus ignoring the observed data), and conditions on θ^* (which is unknown).

Not only is the frequentist definition unnatural, it cannot even be computed, because θ^* is unknown. Consequently, we cannot compare different estimators in terms of their frequentist risk. We discuss various solutions to this below.

6.3.1 Bayes risk

How do we choose amongst estimators? We need some way to convert $R(\boldsymbol{\theta}^*, \delta)$ into a single measure of quality, $R(\delta)$, which does not depend on knowing $\boldsymbol{\theta}^*$. One approach is to put a prior on θ^*, and then to define **Bayes risk** or **integrated risk** of an estimator as follows:

$$R_B(\delta) \triangleq \mathbb{E}_{p(\boldsymbol{\theta}^*)}\left[R(\boldsymbol{\theta}^*, \delta)\right] = \int R(\boldsymbol{\theta}^*, \delta)p(\boldsymbol{\theta}^*)d\boldsymbol{\theta}^* \tag{6.11}$$

A **Bayes estimator** or **Bayes decision rule** is one which minimizes the expected risk:

$$\delta_B \triangleq \operatorname*{argmin}_{\delta} R_B(\delta) \tag{6.12}$$

Note that the integrated risk is also called the **preposterior risk**, since it is before we have seen the data. Minimizing this can be useful for experiment design.

We will now prove a very important theorem, that connects the Bayesian and frequentist approaches to decision theory.

Theorem 6.3.1. *A Bayes estimator can be obtained by minimizing the posterior expected loss for each* $\mathbf{x}$.

Proof. By switching the order of integration, we have

$$\begin{aligned}
R_B(\delta) &= \int \left[\sum_{\mathbf{x}}\sum_{y} L(y, \delta(\mathbf{x}))p(\mathbf{x}, y|\boldsymbol{\theta}^*)\right] p(\boldsymbol{\theta}^*)d\boldsymbol{\theta}^* && (6.13)\\
&= \sum_{\mathbf{x}}\sum_{y}\int_{\Theta} L(y, \delta(\mathbf{x}))p(\mathbf{x}, y, \boldsymbol{\theta}^*)d\boldsymbol{\theta}^* && (6.14)\\
&= \sum_{\mathbf{x}}\left[\sum_{y} L(y, \delta(\mathbf{x}))p(y|\mathbf{x})dy\right] p(\mathbf{x}) && (6.15)\\
&= \sum_{\mathbf{x}} \rho(\delta(\mathbf{x})|\mathbf{x})\ p(\mathbf{x}) && (6.16)
\end{aligned}$$

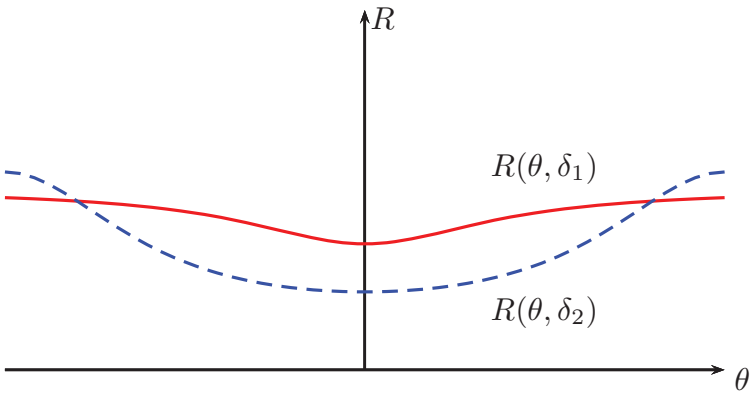


Figure 6.2 Risk functions for two decision procedures, δ_1 and δ_2. Since δ_1 has lower worst case risk, it is the minimax estimator, even though δ_2 has lower risk for most values of θ. Thus minimax estimators are overly conservative.

To minimize the overall expectation, we just minimize the term inside for each $\mathbf{x}$, so our decision rule is to pick

$$\delta_B(\mathbf{x}) = \operatorname*{argmin}_{a \in \mathcal{A}} \rho(a|\mathbf{x}) \tag{6.17}$$

□

Hence we see that the picking the optimal action on a case-by-case basis (as in the Bayesian approach) is optimal on average (as in the frequentist approach). In other words, the Bayesian approach provides a good way of achieving frequentist goals. In fact, one can go further and prove the following.

Theorem 6.3.2 (Wald, 1950)**.** *Every admissable decision rule is a Bayes decision rule with respect to some, possibly improper, prior distribution.*

This theorem shows that *the best way to minimize frequentist risk is to be Bayesian!* See (Bernardo and Smith 1994, p448) for further discussion of this point.

6.3.2 Minimax risk

Obviously some frequentists dislike using Bayes risk since it requires the choice of a prior (although this is only in the evaluation of the estimator, not necessarily as part of its construction). An alternative approach is as follows. Define the **maximum risk** of an estimator as

$$R_{max}(\delta) \triangleq \max_{\boldsymbol{\theta}^*} R(\boldsymbol{\theta}^*, \delta) \tag{6.18}$$

A **minimax rule** is one which minimizes the maximum risk:

$$\delta_{MM} \triangleq \operatorname*{argmin}_{\delta} R_{max}(\delta) \tag{6.19}$$

For example, in Figure 6.2, we see that δ_1 has lower worst-case risk than δ_2, ranging over all possible values of θ^*, so it is the minimax estimator (see Section 6.3.3.1 for an explanation of how to compute a risk function for an actual model).

Minimax estimators have a certain appeal. However, computing them can be hard. And furthermore, they are very pessimistic. In fact, one can show that all minimax estimators are equivalent to Bayes estimators under a **least favorable prior**. In most statistical situations (excluding game theoretic ones), assuming nature is an adversary is not a reasonable assumption.

6.3.3 Admissible estimators

The basic problem with frequentist decision theory is that it relies on knowing the true distribution $p(\cdot|\theta^*)$ in order to evaluate the risk. However, It might be the case that some estimators are worse than others regardless of the value of θ^*. In particular, if $R(\theta, \delta_1) \leq R(\theta, \delta_2)$ for all $\theta \in \Theta$, then we say that δ_1 **dominates** δ_2. The domination is said to be strict if the inequality is strict for some θ. An estimator is said to be **admissible** if it is not strictly dominated by any other estimator.

6.3.3.1 Example

Let us give an example, based on (Bernardo and Smith 1994). Consider the problem of estimating the mean of a Gaussian. We assume the data is sampled from $x_i \sim \mathcal{N}(\theta^*, \sigma^2 = 1)$ and use quadratic loss, $L(\theta, \hat{\theta}) = (\theta - \hat{\theta})^2$. The corresponding risk function is the MSE. Some possible decision rules or estimators $\hat{\theta}(\mathbf{x}) = \delta(\mathbf{x})$ are as follows:

- $\delta_1(\mathbf{x}) = \overline{x}$, the sample mean
- $\delta_2(\mathbf{x}) = \tilde{\mathbf{x}}$, the sample median
- $\delta_3(\mathbf{x}) = \theta_0$, a fixed value
- $\delta_\kappa(\mathbf{x})$, the posterior mean under a $\mathcal{N}(\theta|\theta_0, \sigma^2/\kappa)$ prior:

$$\delta_\kappa(\mathbf{x}) = \frac{N}{N+\kappa}\overline{x} + \frac{\kappa}{N+\kappa}\theta_0 = w\overline{x} + (1-w)\theta_0 \tag{6.20}$$

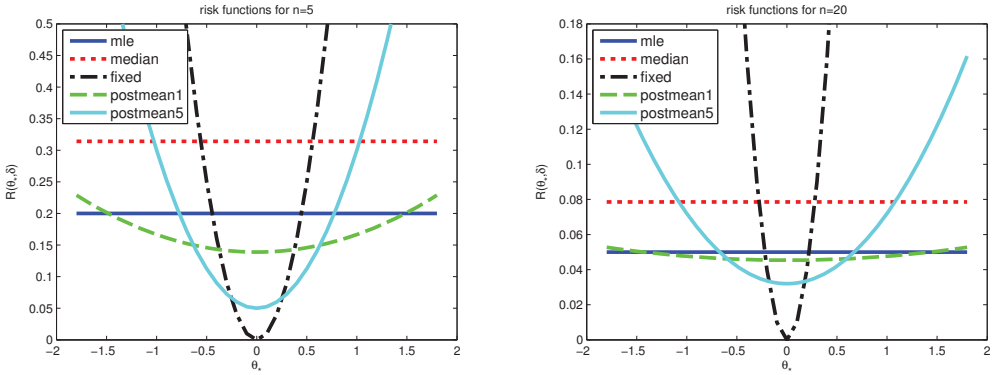
For δ_κ, we consider a weak prior, $\kappa = 1$, and a stronger prior, $\kappa = 5$. The prior mean is θ_0, some fixed value. We assume σ^2 is known. (Thus $\delta_3(\mathbf{x})$ is the same as $\delta_\kappa(\mathbf{x})$ with an infinitely strong prior, $\kappa = \infty$.)

Let us now derive the risk functions analytically. (We can do this since in this toy example, we know the true parameter θ^*.) In Section 6.4.4, we show that the MSE can be decomposed into squared bias plus variance:

$$MSE(\hat{\theta}(\cdot)|\theta^*) \quad = \quad \text{var}\left[\hat{\theta}\right] + \text{bias}^2(\hat{\theta}) \tag{6.21}$$

The sample mean is unbiased, so its risk is

$$MSE(\delta_1|\theta^*) = \text{var}\left[\overline{x}\right] = \frac{\sigma^2}{N} \tag{6.22}$$



(a) (b)

Figure 6.3 Risk functions for estimating the mean of a Gaussian using data sampled $\mathcal{N}(\theta^*, \sigma^2 = 1)$. The solid dark blue horizontal line is the MLE, the solid light blue curved line is the posterior mean when $\kappa = 5$. Left: $N = 5$ samples. Right: $N = 20$ samples. Based on Figure B.1 of (Bernardo and Smith 1994). Figure generated by `riskFnGauss`.

The sample median is also unbiased. One can show that the variance is approximately $\pi/(2N)$, so

$$MSE(\delta_2|\theta^*) = \frac{\pi}{2N} \tag{6.23}$$

For $\delta_3(\mathbf{x}) = \theta_0$, the variance is zero, so

$$MSE(\delta_3|\theta^*) = (\theta^* - \theta_0)^2 \tag{6.24}$$

Finally, for the posterior mean, we have

$$\begin{aligned} MSE(\delta_\kappa|\theta^*) &= \mathbb{E}\left[(w\overline{x} + (1-w)\theta_0 - \theta^*)^2\right] && (6.25)\\ &= \mathbb{E}\left[(w(\overline{x} - \theta^*) + (1-w)(\theta_0 - \theta^*))^2\right] && (6.26)\\ &= w^2\frac{\sigma^2}{N} + (1-w)^2(\theta_0 - \theta^*)^2 && (6.27)\\ &= \frac{1}{(N+\kappa)^2}\left(N\sigma^2 + \kappa^2(\theta_0 - \theta^*)^2\right) && (6.28) \end{aligned}$$

These functions are plotted in Figure 6.3 for $N \in \{5, 20\}$. We see that in general, the best estimator depends on the value of θ^*, which is unknown. If θ^* is very close to θ_0, then δ_3 (which just predicts θ_0) is best. If θ^* is within some reasonable range around θ_0, then the posterior mean, which combines the prior guess of θ_0 with the actual data, is best. If θ^* is far from θ_0, the MLE is best. None of this should be suprising: a small amount of shrinkage (using the posterior mean with a weak prior) is usually desirable, assuming our prior mean is sensible.

What is more surprising is that the risk of decision rule δ_2 (sample median) is always higher than that of δ_1 (sample mean) for every value of θ^*. Consequently the sample median is an

inadmissible estimator for this particular problem (where the data is assumed to come from a Gaussian).

In practice, the sample median is often better than the sample mean, because it is more robust to outliers. One can show (Minka 2000d) that the median is the Bayes estimator (under squared loss) if we assume the data comes from a Laplace distribution, which has heavier tails than a Gaussian. More generally, we can construct robust estimators by using flexible models of our data, such as mixture models or non-parametric density estimators (Section 14.7.2), and then computing the posterior mean or median.

6.3.3.2 Stein's paradox *

Suppose we have N iid random variables $X_i \sim \mathcal{N}(\theta_i, 1)$, and we want to estimate the θ_i. The obvious estimator is the MLE, which in this case sets $\hat{\theta}_i = x_i$. It turns out that this is an inadmissible estimator under quadratic loss, when $N \geq 4$.

To show this, it suffices to construct an estimator that is better. The James-Stein estimator is one such estimator, and is defined as follows:

$$\hat{\theta}_i \quad = \quad \hat{B}\overline{x} + (1 - \hat{B})x_i = \overline{x} + (1 - \hat{B})(x_i - \overline{x}) \tag{6.29}$$

where $\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$ and $0 < B < 1$ is some tuning constant. This estimate "shrinks" the θ_i towards the overall mean. (We derive this estimator using an empirical Bayes approach in Section 5.6.2.)

It can be shown that this shrinkage estimator has lower frequentist risk (MSE) than the MLE (sample mean) for $N \geq 4$. This is known as **Stein's paradox**. The reason it is called a paradox is illustrated by the following example. Suppose θ_i is the "true" IQ of student i and X_i is his test score. Why should my estimate of θ_i depend on the global mean $\overline{x}$, and hence on some other student's scores? One can create even more paradoxical examples by making the different dimensions be qualitatively different, e.g., θ_1 is my IQ, θ_2 is the average rainfall in Vancouver, etc.

The solution to the paradox is the following. If your goal is to estimate just θ_i, you cannot do better than using x_i, but if the goal is to estimate the whole vector $\boldsymbol{\theta}$, and you use squared error as your loss function, then shrinkage helps. To see this, suppose we want to estimate $||\boldsymbol{\theta}||_2^2$ from a single sample $\mathbf{x} \sim \mathcal{N}(\boldsymbol{\theta}, \mathbf{I})$. A simple estimate is $||\mathbf{x}||_2^2$, but this will overestimate the result, since

$$\mathbb{E}\left[\|\mathbf{x}\|_2^2\right] = \mathbb{E}\left[\sum_i x_i^2\right] = \sum_{i=1}^{N}\left(1 + \theta_i^2\right) = N + ||\boldsymbol{\theta}||_2^2 \tag{6.30}$$

Consequently we can reduce our risk by pooling information, even from unrelated sources, and shrinking towards the overall mean. In Section 5.6.2, we give a Bayesian explanation for this. See also (Efron and Morris 1975).

6.3.3.3 Admissibility is not enough

It seems clear that we can restrict our search for good estimators to the class of admissible estimators. But in fact it is easy to construct admissible estimators, as we show in the following example.

Theorem 6.3.3. *Let $X \sim \mathcal{N}(\theta, 1)$, and consider estimating θ under squared loss. Let $\delta_1(x) = \theta_0$, a constant* independent of the data. *This is an admissible estimator.*

Proof. Suppose not. Then there is some other estimator δ_2 with smaller risk, so $R(\theta^*, \delta_2) \leq R(\theta^*, \delta_1)$, where the inequality must be strict for some θ^*. Suppose the true parameter is $\theta^* = \theta_0$. Then $R(\theta^*, \delta_1) = 0$, and

$$R(\theta^*, \delta_2) = \int (\delta_2(x) - \theta_0)^2 p(x|\theta_0) dx \tag{6.31}$$

Since $0 \leq R(\theta^*, \delta_2) \leq R(\theta^*, \delta_1)$ for all θ^*, and $R(\theta_0, \delta_1) = 0$, we have $R(\theta_0, \delta_2) = 0$ and hence $\delta_2(x) = \theta_0 = \delta_1(x)$. Thus the only way δ_2 can avoid having higher risk than δ_1 at some specific point θ_0 is by being equal to δ_1. Hence there is no other estimator δ_2 with strictly lower risk, so δ_2 is admissible. □

6.4 Desirable properties of estimators

Since frequentist decision theory does not provide an automatic way to choose the best estimator, we need to come up with other heuristics for choosing amongst them. In this section, we discuss some properties we would like estimators to have. Unfortunately, we will see that we cannot achieve all of these properties at the same time.

6.4.1 Consistent estimators

An estimator is said to be **consistent** if it eventually recovers the true parameters that generated the data as the sample size goes to infinity, i.e., $\hat{\theta}(\mathcal{D}) \to \theta^*$ as $|\mathcal{D}| \to \infty$ (where the arrow denotes convergence in probability). Of course, this concept only makes sense if the data actually comes from the specified model with parameters θ^*, which is not usually the case with real data. Nevertheless, it can be a useful theoretical property.

It can be shown that the MLE is a consistent estimator. The intuitive reason is that maximizing likelihood is equivalent to minimizing $\mathbb{KL}\left(p(\cdot|\boldsymbol{\theta}^*)||p(\cdot|\hat{\boldsymbol{\theta}})\right)$, where $p(\cdot|\boldsymbol{\theta}^*)$ is the true distribution and $p(\cdot|\hat{\boldsymbol{\theta}})$ is our estimate. We can achieve 0 KL divergence iff $\hat{\boldsymbol{\theta}} = \boldsymbol{\theta}^*$.[4]

6.4.2 Unbiased estimators

The **bias** of an estimator is defined as

$$\text{bias}(\hat{\theta}(\cdot)) = \mathbb{E}_{p(\mathcal{D}|\theta_*)}\left[\hat{\theta}(\mathcal{D}) - \theta_*\right] \tag{6.32}$$

where θ_* is the true parameter value. If the bias is zero, the estimator is called **unbiased**. This means the sampling distribution is centered on the true parameter. For example, the MLE for a Gaussian mean is unbiased:

$$\text{bias}(\hat{\mu}) = \mathbb{E}\left[\overline{x}\right] - \mu = \mathbb{E}\left[\frac{1}{N}\sum_{i=1}^{N} x_i\right] - \mu = \frac{N\mu}{N} - \mu = 0 \tag{6.33}$$

4. If the model is unidentifiable, the MLE may select a set of parameters that is different from the true parameters but for which the induced distribution, $p(\cdot|\hat{\boldsymbol{\theta}})$, is the same as the exact distribution. Such parameters are said to be likelihood equivalent.

However, the MLE for a Gaussian variance, $\hat{\sigma}^2$, is not an unbiased estimator of σ^2. In fact, one can show (Exercise 6.3) that

$$\mathbb{E}\left[\hat{\sigma}^2\right] = \frac{N-1}{N}\sigma^2 \tag{6.34}$$

However, the following estimator

$$\hat{\sigma}^2_{N-1} = \frac{N}{N-1}\hat{\sigma}^2 = \frac{1}{N-1}\sum_{i=1}^{N}(x_i - \overline{x})^2 \tag{6.35}$$

is an unbiased estimator, which we can easily prove as follows:

$$\mathbb{E}\left[\hat{\sigma}^2_{N-1}\right] = \mathbb{E}\left[\frac{N}{N-1}\hat{\sigma}^2\right] = \frac{N}{N-1}\frac{N-1}{N}\sigma^2 = \sigma^2 \tag{6.36}$$

In Matlab, `var(X)` returns $\hat{\sigma}^2_{N-1}$, whereas `var(X,1)` returns $\hat{\sigma}^2$ (the MLE). For large enough N, the difference will be negligible.

Although the MLE may sometimes be a biased estimator, one can show that asymptotically, it is always unbiased. (This is necessary for the MLE to be a consistent estimator.)

Although being unbiased sounds like a desirable property, this is not always true. See Section 6.4.4 and (Lindley 1972) for discussion of this point.

6.4.3 Minimum variance estimators

It seems intuitively reasonable that we want our estimator to be unbiased (although we shall give some arguments against this claim below). However, being unbiased is not enough. For example, suppose we want to estimate the mean of a Gaussian from $\mathcal{D} = \{x_1, \ldots, x_N\}$. The estimator that just looks at the first data point, $\hat{\theta}(\mathcal{D}) = x_1$, is an unbiased estimator, but will generally be further from θ_* than the empirical mean $\overline{x}$ (which is also unbiased). So the variance of an estimator is also important.

A natural question is: how long can the variance go? A famous result, called the **Cramer-Rao lower bound**, provides a lower bound on the variance of any unbiased estimator. More precisely,

Theorem 6.4.1 (Cramer-Rao inequality)**.** *Let $X_1, \ldots, X_n \sim p(X|\theta_0)$ and $\hat{\theta} = \hat{\theta}(x_1, \ldots, x_n)$ be an unbiased estimator of θ_0. Then, under various smoothness assumptions on $p(X|\theta_0)$, we have*

$$\text{var}\left[\hat{\theta}\right] \geq \frac{1}{nI(\theta_0)} \tag{6.37}$$

where $I(\theta_0)$ is the Fisher information matrix (see Section 6.2.2).

A proof can be found e.g., in (Rice 1995, p275).

It can be shown that the MLE achieves the Cramer Rao lower bound, and hence has the smallest asymptotic variance of any unbiased estimator. Thus MLE is said to be **asymptotically optimal**.

6.4.4 The bias-variance tradeoff

Although using an unbiased estimator seems like a good idea, this is not always the case. To see why, suppose we use quadratic loss. As we showed above, the corresponding risk is the MSE. We now derive a very useful decomposition of the MSE. (All expectations and variances are wrt the true distribution $p(\mathcal{D}|\theta^*)$, but we drop the explicit conditioning for notational brevity.) Let $\hat{\theta} = \hat{\theta}(\mathcal{D})$ denote the estimate, and $\overline{\theta} = \mathbb{E}\left[\hat{\theta}\right]$ denote the expected value of the estimate (as we vary $\mathcal{D}$). Then we have

$$\begin{aligned}
\mathbb{E}\left[(\hat{\theta} - \theta^*)^2\right] &= \mathbb{E}\left[\left[(\hat{\theta} - \overline{\theta}) + (\overline{\theta} - \theta^*)\right]^2\right] && (6.38)\\
&= \mathbb{E}\left[\left(\hat{\theta} - \overline{\theta}\right)^2\right] + 2(\overline{\theta} - \theta^*)\mathbb{E}\left[\hat{\theta} - \overline{\theta}\right] + (\overline{\theta} - \theta^*)^2 && (6.39)\\
&= \mathbb{E}\left[\left(\hat{\theta} - \overline{\theta}\right)^2\right] + (\overline{\theta} - \theta^*)^2 && (6.40)\\
&= \mathrm{var}\left[\hat{\theta}\right] + \mathrm{bias}^2(\hat{\theta}) && (6.41)
\end{aligned}$$

In words,

$$\boxed{\text{MSE = variance + bias}^2} \tag{6.42}$$

This is called the **bias-variance tradeoff** (see e.g., (Geman et al. 1992)). What it means is that it might be wise to use a biased estimator, so long as it reduces our variance, assuming our goal is to minimize squared error.

6.4.4.1 Example: estimating a Gaussian mean

Let us give an example, based on (Hoff 2009, p79). Suppose we want to estimate the mean of a Gaussian from $\mathbf{x} = (x_1, \ldots, x_N)$. We assume the data is sampled from $x_i \sim \mathcal{N}(\theta^* = 1, \sigma^2)$. An obvious estimate is the MLE. This has a bias of 0 and a variance of

$$\mathrm{var}\left[\overline{x}|\theta^*\right] = \frac{\sigma^2}{N} \tag{6.43}$$

But we could also use a MAP estimate. In Section 4.6.1, we show that the MAP estimate under a Gaussian prior of the form $\mathcal{N}(\theta_0, \sigma^2/\kappa_0)$ is given by

$$\tilde{x} \triangleq \frac{N}{N+\kappa_0}\overline{x} + \frac{\kappa_0}{N+\kappa_0}\theta_0 = w\overline{x} + (1-w)\theta_0 \tag{6.44}$$

where $0 \leq w \leq 1$ controls how much we trust the MLE compared to our prior. (This is also the posterior mean, since the mean and mode of a Gaussian are the same.) The bias and variance are given by

$$\begin{aligned}
\mathbb{E}\left[\tilde{x}\right] - \theta^* &= w\theta_0 + (1-w)\theta_0 - \theta^* = (1-w)(\theta_0 - \theta^*) && (6.45)\\
\mathrm{var}\left[\tilde{x}\right] &= w^2\frac{\sigma^2}{N} && (6.46)
\end{aligned}$$

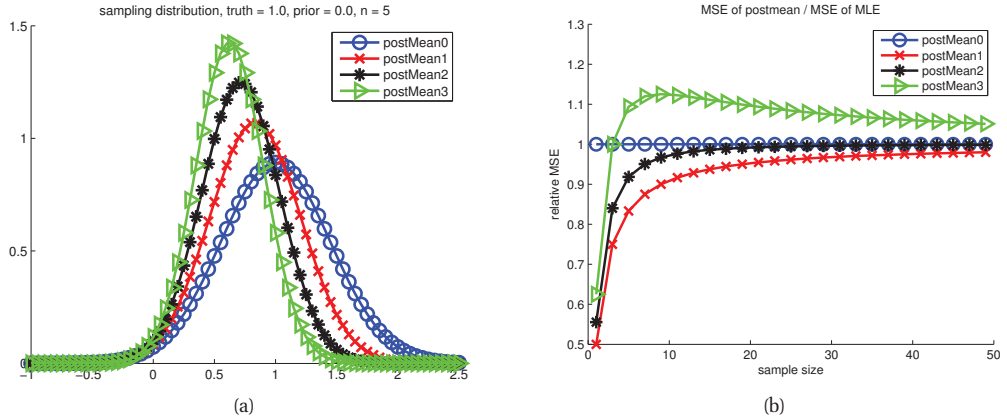


Figure 6.4 Left: Sampling distribution of the MAP estimate with different prior strengths κ_0. (The MLE corresponds to $\kappa_0 = 0$.) Right: MSE relative to that of the MLE versus sample size. Based on Figure 5.6 of (Hoff 2009). Figure generated by `samplingDistGaussShrinkage`.

So although the MAP estimate is biased (assuming $w < 1$), it has lower variance.

Let us assume that our prior is slightly misspecified, so we use $\theta_0 = 0$, whereas the truth is $\theta^* = 1$. In Figure 6.4(a), we see that the sampling distribution of the MAP estimate for $\kappa_0 > 0$ is biased away from the truth, but has lower variance (is narrower) than that of the MLE.

In Figure 6.4(b), we plot $\text{mse}(\tilde{x})/\text{mse}(\overline{x})$ vs N. We see that the MAP estimate has lower MSE than the MLE, especially for small sample size, for $\kappa_0 \in \{1, 2\}$. The case $\kappa_0 = 0$ corresponds to the MLE, and the case $\kappa_0 = 3$ corresponds to a strong prior, which hurts performance because the prior mean is wrong. It is clearly important to "tune" the strength of the prior, a topic we discuss later.

6.4.4.2 Example: ridge regression

Another important example of the bias variance tradeoff arises in ridge regression, which we discuss in Section 7.5. In brief, this corresponds to MAP estimation for linear regression under a Gaussian prior, $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \lambda^{-1}\mathbf{I})$ The zero-mean prior encourages the weights to be small, which reduces overfitting; the precision term, λ, controls the strength of this prior. Setting $\lambda = 0$ results in the MLE; using $\lambda > 0$ results in a biased estimate. To illustrate the effect on the variance, consider a simple example. Figure 6.5 on the left plots each individual fitted curve, and on the right plots the average fitted curve. We see that as we increase the strength of the regularizer, the variance decreases, but the bias increases.

6.4.4.3 Bias-variance tradeoff for classification

If we use 0-1 loss instead of squared error, the above analysis breaks down, since the frequentist risk is no longer expressible as squared bias plus variance. In fact, one can show (Exercise 7.2 of (Hastie et al. 2009)) that the bias and variance combine multiplicatively. If the estimate is on

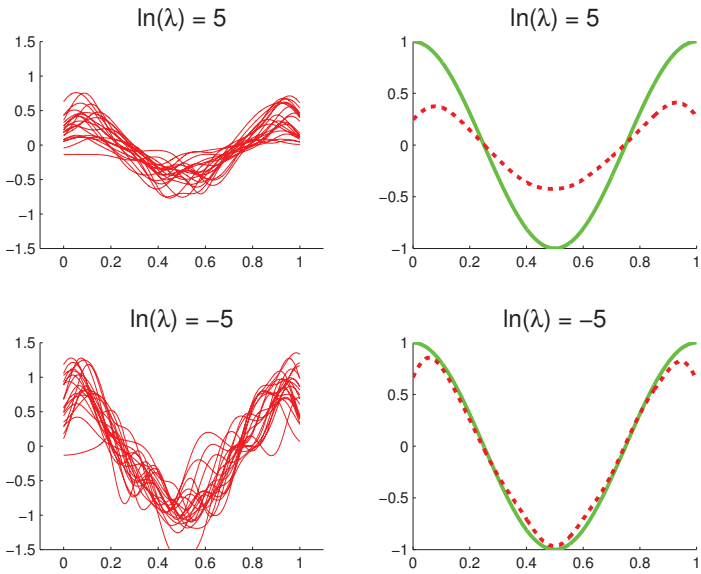


Figure 6.5 Illustration of bias-variance tradeoff for ridge regression. We generate 100 data sets from the true function, shown in solid green. Left: we plot the regularized fit for 20 different data sets. We use linear regression with a Gaussian RBF expansion, with 25 centers evenly spread over the $[0, 1]$ interval. Right: we plot the average of the fits, averaged over all 100 datasets. Top row: strongly regularized: we see that the individual fits are similar to each other (low variance), but the average is far from the truth (high bias). Bottom row: lightly regularized: we see that the individual fits are quite different from each other (high variance), but the average is close to the truth (low bias). Based on (Bishop 2006a) Figure 3.5. Figure generated by `biasVarModelComplexity3`.

the correct side of the decision boundary, then the bias is negative, and decreasing the variance will decrease the misclassification rate. But if the estimate is on the wrong side of the decision boundary, then the bias is positive, so it pays to *increase* the variance (Friedman 1997a). This little known fact illustrates that the bias-variance tradeoff is not very useful for classification. It is better to focus on expected loss (see below), not directly on bias and variance. We can approximate the expected loss using cross validatinon, as we discuss in Section 6.5.3.

6.5 Empirical risk minimization

Frequentist decision theory suffers from the fundamental problem that one cannot actually compute the risk function, since it relies on knowing the true data distribution. (By contrast, the Bayesian posterior expected loss can always be computed, since it conditions on the the data rather than conditioning on θ^*.) However, there is one setting which avoids this problem, and that is where the task is to predict observable quantities, as opposed to estimating hidden variables or parameters. That is, instead of looking at loss functions of the form $L(\boldsymbol{\theta}, \delta(\mathcal{D}))$, where $\boldsymbol{\theta}$ is the true but unknown parameter, and $\delta(\mathcal{D})$ is our estimator, let us look at loss

functions of the form $L(y, \delta(\mathbf{x}))$, where y is the true but unknown response, and $\delta(\mathbf{x})$ is our prediction given the input $\mathbf{x}$. In this case, the frequentist risk becomes

$$R(p_*, \delta) \triangleq \mathbb{E}_{(\mathbf{x},y)\sim p_*}\left[L(y, \delta(\mathbf{x})\right] = \sum_{\mathbf{x}} \sum_{y} L(y, \delta(\mathbf{x}))p_*(\mathbf{x}, y) \tag{6.47}$$

where p_* represents "nature's distribution". Of course, this distribution is unknown, but a simple approach is to use the empirical distribution, derived from some training data, to approximate p_*, i.e.,

$$p_*(\mathbf{x}, y) \approx p_{\text{emp}}(\mathbf{x}, y) \triangleq \frac{1}{N}\sum_{i=1}^{N} \delta_{\mathbf{x}_i}(\mathbf{x})\delta_{y_i}(y) \tag{6.48}$$

We then define the **empirical risk** as follows:

$$R_{emp}(\mathcal{D}, \mathcal{D}) \triangleq R(p_{\text{emp}}, \delta) = \frac{1}{N}\sum_{i=1}^{N} L(y_i, \delta(\mathbf{x}_i)) \tag{6.49}$$

In the case of 0-1 loss, $L(y, \delta(\mathbf{x})) = \mathbb{I}(y \neq \delta(\mathbf{x}))$, this becomes the **misclassification rate**. In the case of squared error loss, $L(y, \delta(\mathbf{x})) = (y - \delta(\mathbf{x}))^2$, this becomes the **mean squared error**. We define the task of **empirical risk minimization** or **ERM** as finding a decision procedure (typically a classification rule) to minimize the empirical risk:

$$\delta_{ERM}(\mathcal{D}) = \underset{\delta}{\operatorname{argmin}}\, R_{emp}(\mathcal{D}, \delta) \tag{6.50}$$

In the unsupervised case, we eliminate all references to y, and replace $L(y, \delta(\mathbf{x}))$ with $L(\mathbf{x}, \delta(\mathbf{x}))$, where, for example, $L(\mathbf{x}, \delta(\mathbf{x})) = ||\mathbf{x} - \delta(\mathbf{x})||_2^2$, which measures the reconstruction error. We can define the decision rule using $\delta(\mathbf{x}) = \text{decode}(\text{encode}(\mathbf{x}))$, as in vector quantization (Section 11.4.2.6) or PCA (section 12.2). Finally, we define the empirical risk as

$$R_{emp}(\mathcal{D}, \delta) = \frac{1}{N}\sum_{i=1}^{N} L(\mathbf{x}_i, \delta(\mathbf{x}_i)) \tag{6.51}$$

Of course, we can always trivially minimize this risk by setting $\delta(\mathbf{x}) = \mathbf{x}$, so it is critical that the encoder-decoder go via some kind of bottleneck.

6.5.1 Regularized risk minimization

Note that the empirical risk is equal to the Bayes risk if our prior about "nature's distribution" is that it is exactly equal to the empirical distribution (Minka 2001b):

$$\mathbb{E}\left[R(p_*, \delta)|p_* = p_{\text{emp}}\right] = R_{emp}(\mathcal{D}, \delta) \tag{6.52}$$

Therefore minimizing the empirical risk will typically result in overfitting. It is therefore often necessary to add a complexity penalty to the objective function:

$$R'(\mathcal{D}, \delta) = R_{emp}(\mathcal{D}, \delta) + \lambda C(\delta) \tag{6.53}$$

where $C(\delta)$ measures the complexity of the prediction function $\delta(\mathbf{x})$ and λ controls the strength of the complexity penalty. This approach is known as **regularized risk minimization** (RRM). Note that if the loss function is negative log likelihood, and the regularizer is a negative log prior, this is equivalent to MAP estimation.

The two key issues in RRM are: how do we measure complexity, and how do we pick λ. For a linear model, we can define the complexity of in terms of its degrees of freedom, discussed in Section 7.5.3. For more general models, we can use the VC dimension, discussed in Section 6.5.4. To pick λ, we can use the methods discussed in Section 6.5.2.

6.5.2 Structural risk minimization

The regularized risk minimization principle says that we should fit the model, for a given complexity penalty, by using

$$\hat{\delta}_\lambda = \underset{\delta}{\operatorname{argmin}} \left[R_{emp}(\mathcal{D}, \delta) + \lambda C(\delta) \right] \tag{6.54}$$

But how should we pick λ? We cannot using the training set, since this will underestimate the true risk, a problem known as **optimism of the training error**. As an alternative, we can use the following rule, known as the **structural risk minimization** principle: (Vapnik 1998):

$$\hat{\lambda} = \underset{\lambda}{\operatorname{argmin}} \hat{R}(\hat{\delta}_\lambda) \tag{6.55}$$

where $\hat{R}(\delta)$ is an estimate of the risk. There are two widely used estimates: cross validation and theoretical upper bounds on the risk. We discuss both of these below.

6.5.3 Estimating the risk using cross validation

We can estimate the risk of some estimator using a validation set. If we don't have a separate validation set, we can use **cross validation** (CV), as we briefly discussed in Section 1.4.8. More precisely, CV is defined as follows. Let there be $N = |\mathcal{D}|$ data cases in the training set. Denote the data in the k'th test fold by $\mathcal{D}_k$ and all the other data by $\mathcal{D}_{-k}$. (In **stratified CV**, these folds are chosen so the class proportions (if discrete labels are present) are roughly equal in each fold.) Let $\mathcal{F}$ be a learning algorithm or fitting function that takes a dataset and a model index m (this could a discrete index, such as the degree of a polynomial, or a continuous index, such as the strength of a regularizer) and returns a parameter vector:

$$\hat{\boldsymbol{\theta}}_m = \mathcal{F}(\mathcal{D}, m) \tag{6.56}$$

Finally, let $\mathcal{P}$ be a prediction function that takes an input and a parameter vector and returns a prediction:

$$\hat{y} = \mathcal{P}(\mathbf{x}, \hat{\boldsymbol{\theta}}) = f(\mathbf{x}, \hat{\boldsymbol{\theta}}) \tag{6.57}$$

Thus the combined **fit-predict cycle** is denoted as

$$f_m(\mathbf{x}, \mathcal{D}) = \mathcal{P}(\mathbf{x}, \mathcal{F}(\mathcal{D}, m)) \tag{6.58}$$

The K-fold CV estimate of the risk of f_m is defined by

$$R(m, \mathcal{D}, K) \triangleq \frac{1}{N} \sum_{k=1}^{K} \sum_{i \in \mathcal{D}_k} L\left(y_i, \mathcal{P}(\mathbf{x}_i, \mathcal{F}(\mathcal{D}_{-k}, m))\right) \tag{6.59}$$

Note that we can call the fitting algorithm once per fold. Let $f_m^k(\mathbf{x}) = \mathcal{P}(\mathbf{x}, \mathcal{F}(\mathcal{D}_{-k}, m))$ be the function that was trained on all the data except for the test data in fold k. Then we can rewrite the CV estimate as

$$R(m, \mathcal{D}, K) = \frac{1}{N} \sum_{k=1}^{K} \sum_{i \in \mathcal{D}_k} L\left(y_i, f_m^k(\mathbf{x}_i)\right) = \frac{1}{N} \sum_{i=1}^{N} L\left(y_i, f_m^{k(i)}(\mathbf{x}_i)\right) \tag{6.60}$$

where $k(i)$ is the fold in which i is used as test data. In other words, we predict y_i using a model that was trained on data that does not contain $\mathbf{x}_i$.

Of $K = N$, the method is known as **leave one out cross validation** or LOOCV. In this case, n the estimated risk becomes

$$R(m, \mathcal{D}, N) = \frac{1}{N} \sum_{i=1}^{N} L\left(y_i, f_m^{-i}(\mathbf{x}_i)\right) \tag{6.61}$$

where $f_m^i(\mathbf{x}) = \mathcal{P}(\mathbf{x}, \mathcal{F}(\mathcal{D}_{-i}, m))$. This requires fitting the model N times, where for f_m^{-i} we omit the i'th training case. Fortunately, for some model classes and loss functions (namely linear models and quadratic loss), we can fit the model once, and analytically "remove" the effect of the i'th training case. This is known as **generalized cross validation** or GCV.

6.5.3.1 Example: using CV to pick λ for ridge regression

As a concrete example, consider picking the strength of the ℓ_2 regularizer in penalized linear regression. We use the following rule:

$$\hat{\lambda} = \arg \min_{\lambda \in [\lambda_{min}, \lambda_{max}]} R(\lambda, \mathcal{D}_{\text{train}}, K) \tag{6.62}$$

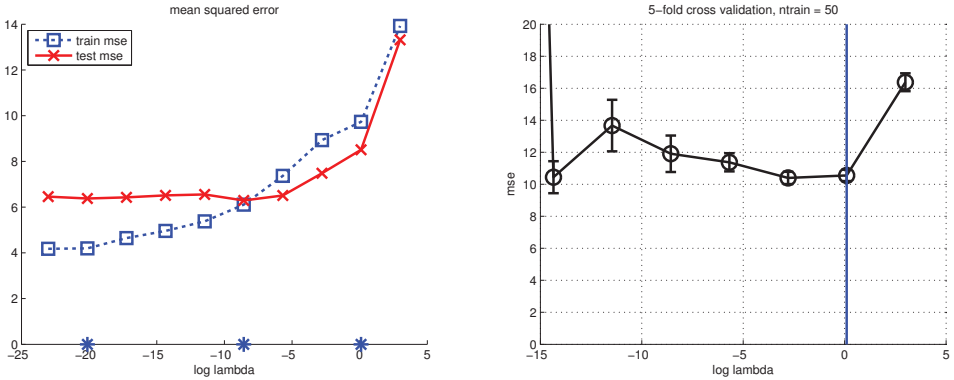
where $[\lambda_{min}, \lambda_{max}]$ is a finite range of λ values that we search over, and $R(\lambda, \mathcal{D}_{\text{train}}, K)$ is the K-fold CV estimate of the risk of using λ, given by

$$R(\lambda, \mathcal{D}_{\text{train}}, K) = \frac{1}{|\mathcal{D}_{\text{train}}|} \sum_{k=1}^{K} \sum_{i \in \mathcal{D}_k} L(y_i, f_\lambda^k(\mathbf{x}_i)) \tag{6.63}$$

where $f_\lambda^k(\mathbf{x}) = \mathbf{x}^T \hat{\mathbf{w}}_\lambda(\mathcal{D}_{-k})$ is the prediction function trained on data excluding fold k, and $\hat{\mathbf{w}}_\lambda(\mathcal{D}) = \arg\min_{\mathbf{w}} NLL(\mathbf{w}, \mathcal{D}) + \lambda ||\mathbf{w}||_2^2$ is the MAP estimate. Figure 6.6(b) gives an example of a CV estimate of the risk vs $\log(\lambda)$, where the loss function is squared error.

When performing classification, we usually use 0-1 loss. In this case, we optimize a convex upper bound on the empirical risk to estimate $\mathbf{w}_\lambda$m but we optimize (the CV estimate of) the risk itself to estimate λ. We can handle the non-smooth 0-1 loss function when estimating λ because we are using brute-force search over the entire (one-dimensional) space.

When we have more than one or two tuning parameters, this approach becomes infeasible. In such cases, one can use empirical Bayes, which allows one to optimize large numbers of hyper-parameters using gradient-based optimizers instead of brute-force search. See Section 5.6 for details.



(a) (b)

Figure 6.6 (a) Mean squared error for ℓ_2 penalized degree 14 polynomial regression vs log regularizer. Same as in Figures 7.8, except now we have $N = 50$ training points instead of 21. The stars correspond to the values used to plot the functions in Figure 7.7. (b) CV estimate. The vertical scale is truncated for clarity. The blue line corresponds to the value chosen by the one standard error rule. Figure generated by `linregPolyVsRegDemo`.

6.5.3.2 The one standard error rule

The above procedure estimates the risk, but does not give any measure of uncertainty. A standard frequentist measure of uncertainty of an estimate is the standard error of the mean, defined by

$$se = \frac{\hat{\sigma}}{\sqrt{N}} = \sqrt{\frac{\hat{\sigma}^2}{N}} \tag{6.64}$$

where $\hat{\sigma}^2$ is an estimate of the variance of the loss:

$$\hat{\sigma}^2 = \frac{1}{N}\sum_{i=1}^{N}(L_i - \overline{L})^2, \quad L_i = L(y_i, f_m^{k(i)}(\mathbf{x}_i)) \quad \overline{L} = \frac{1}{N}\sum_{i=1}^{N} L_i \tag{6.65}$$

Note that σ measures the intrinsic variability of L_i across samples, whereas se measures our uncertainty about the mean $\overline{L}$.

Suppose we apply CV to a set of models and compute the mean and se of their estimated risks. A common heuristic for picking a model from these noisy estimates is to pick the value which corresponds to the simplest model whose risk is no more than one standard error above the risk of the best model; this is called the **one-standard error rule** (Hastie et al. 2001, p216). For example, in Figure 6.6, we see that this heuristic does not choose the lowest point on the curve, but one that is slightly to its right, since that corresponds to a more heavily regularized model with essentially the same empirical performance.

6.5.3.3 CV for model selection in non-probabilistic unsupervised learning

If we are performing unsupervised learning, we must use a loss function such as $L(\mathbf{x}, \delta(\mathbf{x})) = ||\mathbf{x} - \delta(\mathbf{x})||_2$, which measures reconstruction error. Here $\delta(\mathbf{x})$ is some encode-decode scheme. However, as we discussed in Section 11.5.2, we cannot use CV to determine the complexity of δ, since we will always get lower loss with a more complex model, even if evaluated on the test set. This is because more complex models will compress the data less, and induce less distortion. Consequently, we must either use probabilistic models, or invent other heuristics.

6.5.4 Upper bounding the risk using statistical learning theory *

The principle problem with cross validation is that it is slow, since we have to fit the model multiple times. This motivates the desire to compute analytic approximations or bounds to the generalization error. This is the studied in the field of **statistical learning theory** (SLT). More precisely, SLT tries to bound the risk $R(p_*, h)$ for any data distribution p_* and hypothesis $h \in \mathcal{H}$ in terms of the empirical risk $R_{emp}(\mathcal{D}, h)$, the sample size $N = |\mathcal{D}|$, and the size of the hypothesis space $\mathcal{H}$.

Let us initially consider the case where the hypothesis space is finite, with size $\dim(\mathcal{H}) = |\mathcal{H}|$. In other words, we are selecting a model/ hypothesis from a finite list, rather than optimizing real-valued parameters, Then we can prove the following.

Theorem 6.5.1. *For any data distribution p_*, and any dataset $\mathcal{D}$ of size N drawn from p_*, the probability that our estimate of the error rate will be more than ϵ wrong, in the worst case, is upper bounded as follows:*

$$P\left(\max_{h \in \mathcal{H}} |R_{emp}(\mathcal{D}, h) - R(p_*, h)| > \epsilon\right) \leq 2\dim(\mathcal{H}) e^{-2N\epsilon^2} \tag{6.66}$$

Proof. To prove this, we need two useful results. First, **Hoeffding's inequality**, which states that if $X_1, \ldots, X_N \sim \text{Ber}(\theta)$, then, for any $\epsilon > 0$,

$$P(|\overline{x} - \theta| > \epsilon) \leq 2e^{-2N\epsilon^2} \tag{6.67}$$

where $\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$. Second, the **union bound**, which says that if $A_1, \ldots, A_d$ are a set of events, then $P(\cup_{i=1}^{d} A_i) \leq \sum_{i=1}^{d} P(A_i)$.

Finally, for notational brevity, let $R(h) = R(h, p_*)$ be the true risk, and $\hat{R}_N(h) = R_{emp}(\mathcal{D}, h)$ be the empirical risk.

Using these results we have

$$\begin{aligned} P\left(\max_{h \in \mathcal{H}} |\hat{R}_N(h) - R(h)| > \epsilon\right) &= P\left(\bigcup_{h \in \mathcal{H}} |\hat{R}_N(h) - R(h)| > \epsilon\right) && (6.68) \\ &\leq \sum_{h \in \mathcal{H}} P\left(|\hat{R}_N(h) - R(h)| > \epsilon\right) && (6.69) \\ &\leq \sum_{h \in \mathcal{H}} 2e^{-2N\epsilon^2} = 2\dim(\mathcal{H}) e^{-2N\epsilon^2} && (6.70) \end{aligned}$$

□

Ths bound tells us that the optimism of the training error increases with $\dim(\mathcal{H})$ but decreases with $N = |\mathcal{D}|$, as is to be expected.

If the hypothesis space $\mathcal{H}$ is infinite (e.g., we have real-valued parameters), we cannot use $\dim(\mathcal{H}) = |\mathcal{H}|$. Instead, we can use a quantity called the **Vapnik-Chervonenkis** or **VC** dimension of the hypothesis class. See (Vapnik 1998) for details.

Stepping back from all the theory, the key intuition behind statistical learning theory is quite simple. Suppose we find a model with low empirical risk. If the hypothesis space $\mathcal{H}$ is very big, relative to the data size, then it is quite likely that we just got "lucky" and were given a data set that is well-modeled by our chosen function by chance. However, this does not mean that such a function will have low generalization error. But if the hypothesis class is sufficiently constrained in size, and/or the training set is sufficiently large, then we are unlikely to get lucky in this way, so a low empirical risk is evidence of a low true risk.

Note that optimism of the training error does not necessarily increase with model complexity, but it does increase with the number of different models that are being searched over.

The advantage of statistical learning theory compared to CV is that the bounds on the risk are quicker to compute than using CV. The disadvantage is that it is hard to compute the VC dimension for many interesting models, and the upper bounds are usually very loose (although see (Kaariainen and Langford 2005)).

One can extend statistical learning theory by taking computational complexity of the learner into account. This field is called **computational learning theory** or **COLT**. Most of this work focuses on the case where h is a binary classifier, and the loss function is 0-1 loss. If we observe a low empirical risk, and the hypothesis space is suitably "small", then we can say that our estimated function is **probably approximately correct** or **PAC**. A hypothesis space is said to be **efficiently PAC-learnable** if there is a polynomial time algorithm that can identify a function that is PAC. See (Kearns and Vazirani 1994) for details.

6.5.5 Surrogate loss functions

Minimizing the loss in the ERM/ RRM framework is not always easy. For example, we might want to optimize the AUC or F1 scores. Or more simply, we might just want to minimize the 0-1 loss, as is common in classification. Unfortunately, the 0-1 risk is a very non-smooth objective and hence is hard to optimize. One alternative is to use maximum likelihood estimation instead, since log-likelihood is a smooth convex upper bound on the 0-1 risk, as we show below.

To see this, consider binary logistic regression, and let $y_i \in \{-1, +1\}$. Suppose our decision function computes the log-odds ratio,

$$f(\mathbf{x}_i) = \log \frac{p(y=1|\mathbf{x}_i, \mathbf{w})}{p(y=-1|\mathbf{x}_i, \mathbf{w})} = \mathbf{w}^T \mathbf{x}_i = \eta_i \tag{6.71}$$

Then the corresponding probability distribution on the output label is

$$p(y_i|\mathbf{x}_i, \mathbf{w}) = \text{sigm}(y_i \eta_i) \tag{6.72}$$

Let us define the **log-loss** as as

$$L_{\text{nll}}(y, \eta) = -\log p(y|\mathbf{x}, \mathbf{w}) = \log(1 + e^{-y\eta}) \tag{6.73}$$

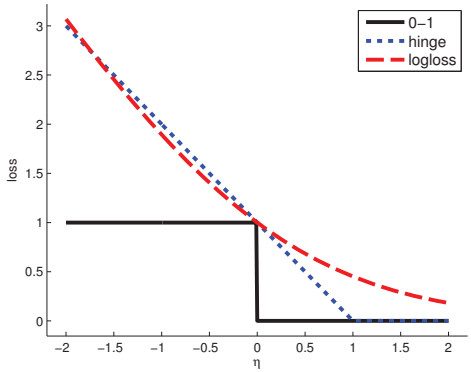


Figure 6.7 Illustration of various loss functions for binary classification. The horizontal axis is the margin $y\eta$, the vertical axis is the loss. The log loss uses log base 2. Figure generated by `hingeLossPlot`.

It is clear that minimizing the average log-loss is equivalent to maximizing the likelihood.

Now consider computing the most probable label, which is equivalent to using $\hat{y} = -1$ if $\eta_i < 0$ and $\hat{y} = +1$ if $\eta_i \geq 0$. The 0-1 loss of our function becomes

$$L_{01}(y, \eta) = \mathbb{I}(y \neq \hat{y}) = \mathbb{I}(y\eta < 0) \tag{6.74}$$

Figure 6.7 plots these two loss functions. We see that the NLL is indeed an upper bound on the 0-1 loss.

Log-loss is an example of a **surrogate loss function**. Another example is the **hinge loss**:

$$L_{\text{hinge}}(y, \eta) = \max(0, 1 - y\eta) \tag{6.75}$$

See Figure 6.7 for a plot. We see that the function looks like a door hinge, hence its name. This loss function forms the basis of a popular classification method known as support vector machines (SVM), which we will discuss in Section 14.5.

The surrogate is usually chosen to be a convex upper bound, since convex functions are easy to minimize. See e.g., (Bartlett et al. 2006) for more information.

6.6 Pathologies of frequentist statistics *

> I believe that it would be very difficult to persuade an intelligent person that current [frequentist] statistical practice was sensible, but that there would be much less difficulty with an approach via likelihood and Bayes' theorem. — George Box, 1962.

Frequentist statistics exhibits various forms of weird and undesirable behaviors, known as **pathologies**. We give a few examples below, in order to caution the reader; these and other examples are explained in more detail in (Lindley 1972; Lindley and Phillips 1976; Lindley 1982; Berger 1985; Jaynes 2003; Minka 1999).

6.6.1 Counter-intuitive behavior of confidence intervals

A **confidence interval** is an interval derived from the sampling distribution of an estimator (whereas a Bayesian credible interval is derived from the posterior of a parameter, as we discussed in Section 5.2.2). More precisely, a frequentist confidence interval for some parameter θ is defined by the following (rather un-natural) expression:

$$C'_\alpha(\theta) = (\ell, u) : P(\ell(\tilde{\mathcal{D}}) \leq \theta \leq u(\tilde{\mathcal{D}})|\tilde{\mathcal{D}} \sim \theta) = 1 - \alpha \tag{6.76}$$

That is, if we sample hypothetical future data $\tilde{\mathcal{D}}$ from θ, then $(\ell(\tilde{\mathcal{D}}), u(\tilde{\mathcal{D}}))$, is a confidence interval if the parameter θ lies inside this interval $1 - \alpha$ percent of the time.

Let us step back for a moment and think about what is going on. In Bayesian statistics, we condition on what is known — namely the observed data, $\mathcal{D}$ — and average over what is not known, namely the parameter θ. In frequentist statistics, we do exactly the opposite: we condition on what is unknown — namely the true parameter value θ — and average over hypothetical future data sets $\tilde{\mathcal{D}}$.

This counter-intuitive definition of confidence intervals can lead to bizarre results. Consider the following example from (Berger 1985, p11). Suppose we draw two integers $\mathcal{D} = (x_1, x_2)$ from

$$p(x|\theta) = \begin{cases} 0.5 & \text{if } x = \theta \\ 0.5 & \text{if } x = \theta + 1 \\ 0 & \text{otherwise} \end{cases} \tag{6.77}$$

If $\theta = 39$, we would expect the following outcomes each with probability 0.25:

$$(39, 39), (39, 40), (40, 39), (40, 40) \tag{6.78}$$

Let $m = \min(x_1, x_2)$ and define the following confidence interval:

$$[\ell(\mathcal{D}), u(\mathcal{D})] = [m, m] \tag{6.79}$$

For the above samples this yields

$$[39, 39], \quad [39, 39], \quad [39, 39], \quad [40, 40] \tag{6.80}$$

Hence Equation 6.79 is clearly a 75% CI, since 39 is contained in 3/4 of these intervals. However, if $\mathcal{D} = (39, 40)$ then $p(\theta = 39|\mathcal{D}) = 1.0$, so we know that θ must be 39, yet we only have 75% "confidence" in this fact.

Another, less contrived example, is as follows. Suppose we want to estimate the parameter θ of a Bernoulli distribution. Let $\overline{x} = \frac{1}{N}\sum_{i=1}^{N} x_i$ be the sample mean. The MLE is $\hat{\theta} = \overline{x}$. An approximate 95% confidence interval for a Bernoulli parameter is $\overline{x} \pm 1.96\sqrt{\overline{x}(1-\overline{x})/N}$ (this is called a **Wald interval** and is based on a Gaussian approximation to the Binomial distribution; compare to Equation 3.27). Now consider a single trial, where $N = 1$ and $x_1 = 0$. The MLE is 0, which overfits, as we saw in Section 3.3.4.1. But our 95% confidence interval is also $(0, 0)$, which seems even worse. It can be argued that the above flaw is because we approximated the true sampling distribution with a Gaussian, or because the sample size was to small, or the parameter "too extreme". However, the Wald interval can behave badly even for large N, and non-extreme parameters (Brown et al. 2001).

6.6.2 p-values considered harmful

Suppose we want to decide whether to accept or reject some baseline model, which we will call the **null hypothesis**. We need to define some decision rule. In frequentist statistics, it is standard to first compute a quantity called the **p-value**, which is defined as the probability (under the null) of observing some **test statistic** $f(\mathcal{D})$ (such as the chi-squared statistic) that is as large *or larger* than that actually observed:[5]

$$\text{pvalue}(\mathcal{D}) \triangleq P(f(\tilde{\mathcal{D}}) \geq f(\mathcal{D})|\tilde{\mathcal{D}} \sim H_0) \tag{6.81}$$

This quantity relies on computing a **tail area probability** of the sampling distribution; we give an example of how to do this below.

Given the p-value, we define our decision rule as follows: we reject the null hypothesis iff the p-value is less than some threshold, such as $\alpha = 0.05$. If we do reject it, we say the difference between the observed test statistic and the expected test statistic is **statistically significant** at level α. This approach is known as **null hypothesis significance testing**, or **NHST**.

This procedure guarantees that our expected type I (false positive) error rate is at most α. This is sometimes interpreted as saying that frequentist hypothesis testing is very conservative, since it is unlikely to accidently reject the null hypothesis. But in fact the opposite is the case: because this method only worries about trying to reject the null, it can never gather evidence in favor of the null, no matter how large the sample size. Because of this, p-values tend to overstate the evidence against the null, and are thus very "trigger happy".

In general there can be huge differences between p-values and the quantity that we really care about, which is the posterior probability of the null hypothesis given the data, $p(H_0|\mathcal{D})$. In particular, Sellke et al. (2001) show that even if the p-value is as slow as 0.05, the posterior probability of H_0 is at least 30%, and often much higher. So frequentists often claim to have "significant" evidence of an effect that cannot be explained by the null hypothesis, whereas Bayesians are usually more conservative in their claims. For example, p-values have been used to "prove" that ESP (extra-sensory perception) is real (Wagenmakers et al. 2011), even though ESP is clearly very improbable. For this reason, p-values have been banned from certain medical journals (Matthews 1998).

Another problem with p-values is that their computation depends on decisions you make about when to stop collecting data, even if these decisions don't change the data you actually observed. For example, suppose I toss a coin $n = 12$ times and observe $s = 9$ successes (heads) and $f = 3$ failures (tails), so $n = s + f$. In this case, n is fixed and s (and hence f) is random. The relevant sampling model is the binomial

$$\text{Bin}(s|n, \theta) = \binom{n}{s} \theta^s (1-\theta)^{n-s} \tag{6.82}$$

Let the null hypothesis be that the coin is fair, $\theta = 0.5$, where θ is the probability of success (heads). The one-sided p-value, using test statistic $t(s) = s$, is

$$p_1 = P(S \geq 9|H_0) = \sum_{s=9}^{12} \text{Bin}(s|12, 0.5) = \sum_{s=9}^{12} \binom{12}{s} 0.5^{12} = 0.073 \tag{6.83}$$

5. The reason we cannot just compute the probability of the observed value of the test statistic is that this will have probability zero under a pdf. The p-value is defined in terms of the cdf, so is always a number between 0 and 1.

The two-sided p-value is

$$p_2 = \sum_{s=9}^{12} \text{Bin}(s|12, 0.5) + \sum_{s=0}^{3} \text{Bin}(s|12, 0.5) = 0.073 + 0.073 = 0.146 \tag{6.84}$$

In either case, the p-value is larger than the magical 5% threshold, so a frequentist would not reject the null hypothesis.

Now suppose I told you that I actually kept tossing the coin until I observed $f = 3$ tails. In this case, f is fixed and n (and hence $s = n - f$) is random. The probability model becomes the **negative binomial distribution**, given by

$$\text{NegBinom}(s|f, \theta) = \binom{s+f-1}{f-1} \theta^s (1-\theta)^f \tag{6.85}$$

where $f = n - s$.

Note that the term which depends on θ is the same in Equations 6.82 and 6.85, so the posterior over θ would be the same in both cases. However, these two interpretations of the same data give different p-values. In particular, under the negative binomial model we get

$$p_3 = P(S \geq 9|H_0) = \sum_{s=9}^{\infty} \binom{3+s-1}{2} (1/2)^s (1/2)^3 = 0.0327 \tag{6.86}$$

So the p-value is 3%, and suddenly there seems to be significant evidence of bias in the coin! Obviously this is ridiculous: the data is the same, so our inferences about the coin should be the same. After all, I could have chosen the experimental protocol at random. It is the outcome of the experiment that matters, not the details of how I decided which one to run.

Although this might seem like just a mathematical curiosity, this also has significant practical implications. In particular, the fact that the **stopping rule** affects the computation of the p-value means that frequentists often do not terminate experiments early, even when it is obvious what the conclusions are, lest it adversely affect their statistical analysis. If the experiments are costly or harmful to people, this is obviously a bad idea. Perhaps it is not surprising, then, that the US Food and Drug Administration (FDA), which regulates clinical trials of new drugs, has recently become supportive of Bayesian methods[6], since Bayesian methods are not affected by the stopping rule.

6.6.3 The likelihood principle

The fundamental reason for many of these pathologies is that frequentist inference violates the **likelihood principle**, which says that inference should be based on the likelihood of the observed data, not based on hypothetical future data that you have not observed. Bayes obviously satisfies the likelihood principle, and consequently does not suffer from these pathologies.

A compelling argument in favor of the likelihood principle was presented in (Birnbaum 1962), who showed that it followed automatically from two simpler principles. The first of these is the **sufficiency principle**, which says that a sufficient statistic contains all the relevant information

6. See http://yamlb.wordpress.com/2006/06/19/the-us-fda-is-becoming-progressively-more-bayesian/.

about an unknown parameter (arguably this is true by definition). The second principle is known as **weak conditionality**, which says that inferences should be based on the events that happened, not which might have happened. To motivate this, consider an example from (Berger 1985). Suppose we need to analyse a substance, and can send it either to a laboratory in New York or in California. The two labs seem equally good, so a fair coin is used to decide between them. The coin comes up heads, so the California lab is chosen. When the results come back, should it be taken into account that the coin could have come up tails and thus the New York lab could have been used? Most people would argue that the New York lab is irrelevant, since the tails event didn't happen. This is an example of weak conditionality. Given this principle, one can show that all inferences should only be based on what was observed, which is in contrast to standard frequentist procedures. See (Berger and Wolpert 1988) for further details on the likelihood principle.

6.6.4 Why isn't everyone a Bayesian?

Given these fundamental flaws of frequentist statistics, and the fact that Bayesian methods do not have such flaws, an obvious question to ask is: "Why isn't everyone a Bayesian?" The (frequentist) statistician Bradley Efron wrote a paper with exactly this title (Efron 1986). His short paper is well worth reading for anyone interested in this topic. Below we quote his opening section:

> The title is a reasonable question to ask on at least two counts. First of all, everone used to be a Bayesian. Laplace wholeheatedly endorsed Bayes's formulation of the inference problem, and most 19th-century scientists followed suit. This included Gauss, whose statistical work is usually presented in frequentist terms.
>
> A second and more important point is the cogency of the Bayesian argument. Modern statisticians, following the lead of Savage and de Finetti, have advanced powerful theoretical arguments for preferring Bayesian inference. A byproduct of this work is a disturbing catalogue of inconsistencies in the frequentist point of view.
>
> Nevertheless, everyone is not a Bayesian. The current era (1986) is the first century in which statistics has been widely used for scientific reporting, and in fact, 20th-century statistics is mainly non-Bayesian. However, Lindley (1975) predicts a change for the 21st century.

Time will tell whether Lindley was right....

Exercises

Exercise 6.1 Pessimism of LOOCV

(Source: Witten05, p152.). Suppose we have a completely random labeled dataset (i.e., the features $\mathbf{x}$ tell us nothing about the class labels y) with N_1 examples of class 1, and N_2 examples of class 2, where $N_1 = N_2$. What is the best misclassification rate any method can achieve? What is the estimated misclassification rate of the same method using LOOCV?

Exercise 6.2 James Stein estimator for Gaussian means

Consider the 2 stage model $Y_i|\theta_i \sim \mathcal{N}(\theta_i, \sigma^2)$ and $\theta_i|\mu \sim \mathcal{N}(m_0, \tau_0^2)$. Suppose $\sigma^2 = 500$ is known and we observe the following 6 data points, $i = 1 : 6$:

```
1505, 1528, 1564, 1498, 1600, 1470
```

a. Find the ML-II estimates of m_0 and τ_0^2.

b. Find the posterior estimates $\mathbb{E}[\theta_i|y_i, m_0, \tau_0]$ and $\text{var}[\theta_i|y_i, m_0, \tau_0]$ for $i = 1$. (The other terms, $i = 2:6$, are computed similarly.)

c. Give a 95% credible interval for $p(\theta_i|y_i, m_0, \tau_0)$ for $i = 1$. Do you trust this interval (assuming the Gaussian assumption is reasonable)? i.e. is it likely to be too large or too small, or just right?

d. What do you expect would happen to your estimates if σ^2 were much smaller (say $\sigma^2 = 1$)? You do not need to compute the numerical answer; just briefly explain what would happen qualitatively, and why.

Exercise 6.3 $\hat{\sigma}^2_{MLE}$ is biased

Show that $\hat{\sigma}^2_{MLE} = \frac{1}{N}\sum_{n=1}^{N}(x_n - \hat{\mu})^2$ is a biased estimator of σ^2, i.e., show

$$\mathbf{E}_{\mathrm{X}_1,\ldots,\mathrm{X}_\mathrm{n}\sim\mathcal{N}(\mu,\sigma)}[\hat{\sigma}^2(\mathrm{X}_1,\ldots,\mathrm{X}_\mathrm{n}) \neq \sigma^2$$

Hint: note that $X_1, \ldots, X_N$ are independent, and use the fact that the expectation of a product of independent random variables is the product of the expectations.

Exercise 6.4 Estimation of σ^2 when μ is known

Suppose we sample $x_1, \ldots, x_N \sim \mathcal{N}(\mu, \sigma^2)$ where μ is a *known* constant. Derive an expression for the MLE for σ^2 in this case. Is it unbiased?

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

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

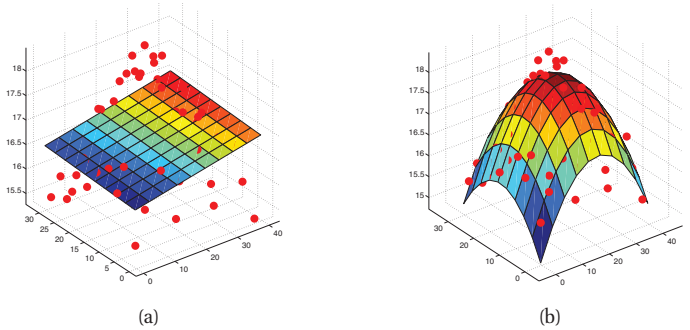
Figure 1.18 illustrates the effect of changing d: increasing the degree d allows us to create increasingly complex functions.

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

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



(a) (b)

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

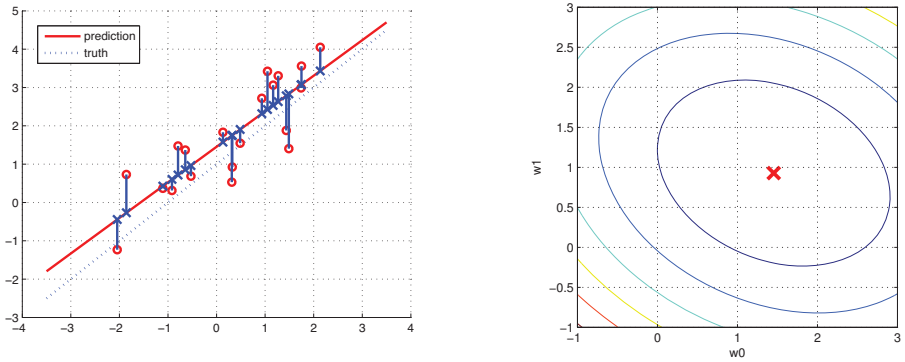
$$\ell(\boldsymbol{\theta}) = \sum_{i=1}^{N} \log \left[\left(\frac{1}{2\pi\sigma^2} \right)^{\frac{1}{2}} \exp\left(-\frac{1}{2\sigma^2} (y_i - \mathbf{w}^T \mathbf{x}_i)^2 \right) \right] \tag{7.7}$$

$$= \frac{-1}{2\sigma^2} RSS(\mathbf{w}) - \frac{N}{2} \log(2\pi\sigma^2) \tag{7.8}$$

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



(a) (b)

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:

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

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

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

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

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

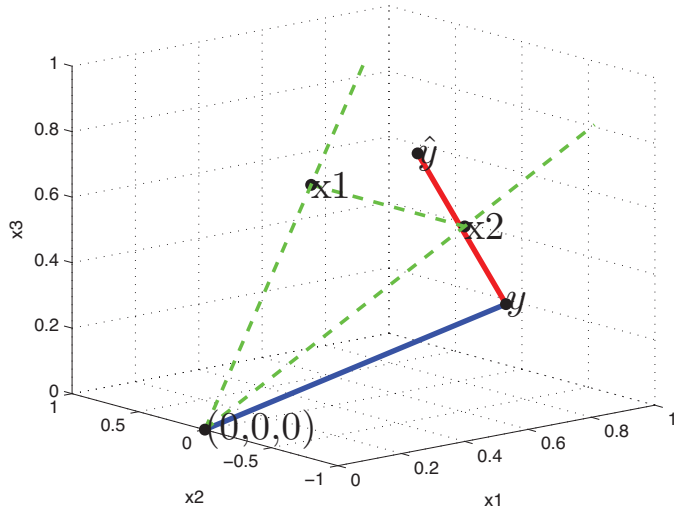


Figure 7.3 Graphical interpretation of least squares for $N = 3$ examples and $D = 2$ features. $\tilde{\mathbf{x}}_1$ and $\tilde{\mathbf{x}}_2$ are vectors in $\mathbb{R}^3$; together they define a 2D plane. $\mathbf{y}$ is also a vector in $\mathbb{R}^3$ but does not lie on this 2D plane. The orthogonal 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

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

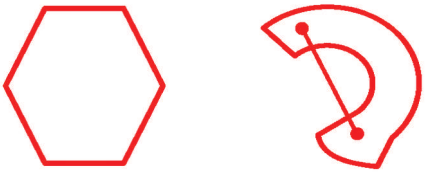
This corresponds to an **orthogonal projection** of $\mathbf{y}$ onto the column space of $\mathbf{X}$. The projection matrix $\mathbf{P} \triangleq \mathbf{X}(\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T$ is called the **hat matrix**, since it "puts the hat on $\mathbf{y}$".

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.

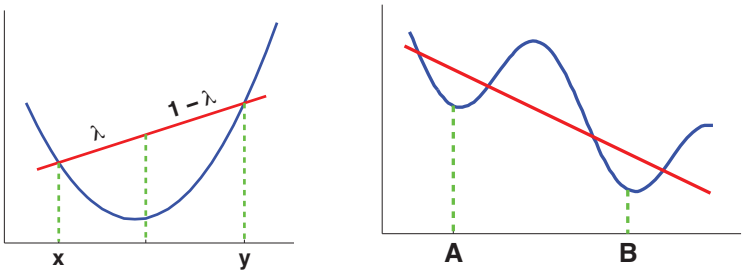
Let us define this concept more precisely. We say a *set* $\mathcal{S}$ is **convex** if for any $\boldsymbol{\theta}, \boldsymbol{\theta}' \in \mathcal{S}$, we have

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



(a) (b)

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



(a) (b)

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 $\mathbf{v}$.

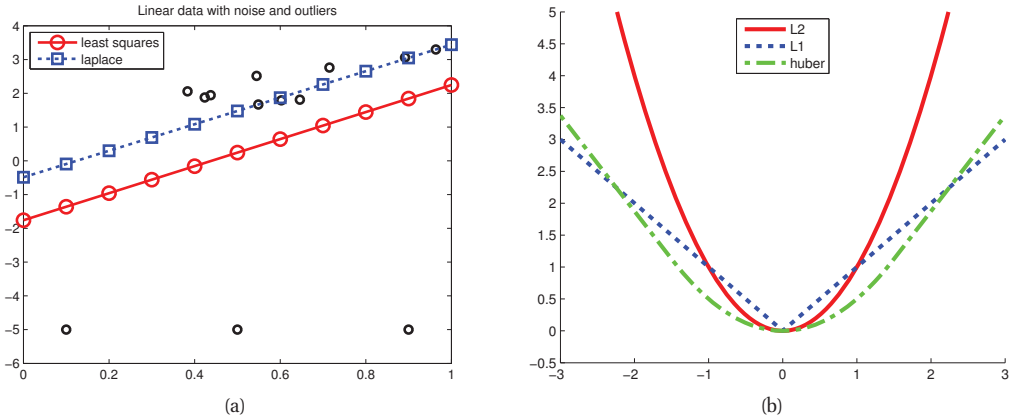


Figure 7.6 (a) Illustration of robust linear regression. Figure generated by `linregRobustDemoCombined`. (b) Illustration of ℓ_2, ℓ_1, and Huber loss functions. Figure generated by `huberLossDemo`.

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:

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

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).)

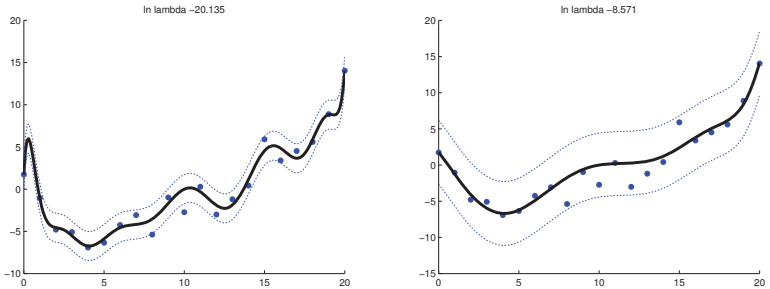


Figure 7.7 Degree 14 Polynomial fit to $N = 21$ data points with increasing amounts of ℓ_2 regularization. Data was generated from noise with variance $\sigma^2 = 4$. The error bars, representing the noise variance σ^2, get wider as the fit gets smoother, since we are ascribing more of the data variation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

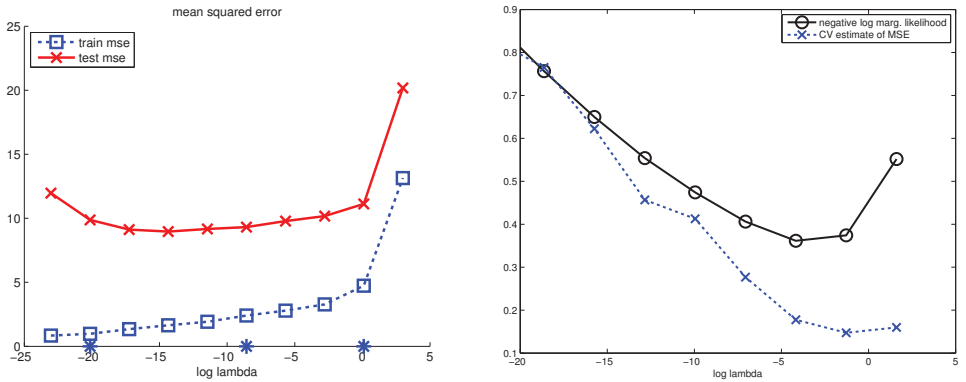
We see that there are many large positive and negative numbers. These balance out exactly to make the curve "wiggle" in just the right way so that it almost perfectly interpolates the data. But this situation is unstable: if we changed the data a little, the coefficients would change a lot.

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

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



(a) (b)

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 $p(\mathbf{w}) = \mathcal{N}(\mathbf{0}, \mathbf{\Lambda}^{-1})$, where $\mathbf{\Lambda}$ is the precision matrix. In the case of ridge regression, $\mathbf{\Lambda} = (1/\tau^2)\mathbf{I}$. 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:

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

where $\mathbf{\Lambda} = \sqrt{\mathbf{\Lambda}}\sqrt{\mathbf{\Lambda}}^T$ is a **Cholesky decomposition** of $\mathbf{\Lambda}$. 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:

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

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

$$\tilde{\mathbf{X}} = \mathbf{QR} \tag{7.41}$$

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 $\mathbf{X} = \mathbf{USV}^T$ be the SVD of $\mathbf{X}$, where $\mathbf{V}^T\mathbf{V} = \mathbf{I}_N$, $\mathbf{UU}^T = \mathbf{U}^T\mathbf{U} = \mathbf{I}_N$, and $\mathbf{S}$ is a diagonal $N \times N$ matrix. Now let $\mathbf{Z} = \mathbf{UD}$ 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 $\mathbf{X} = \mathbf{USV}^T$ be the SVD of $\mathbf{X}$. From Equation 7.44, we have

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

Hence the ridge predictions on the training set are given by

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

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

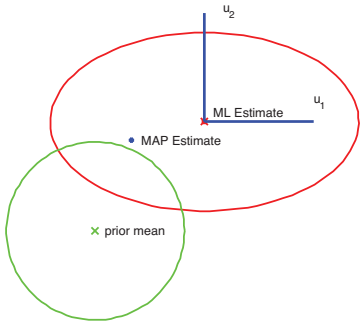


Figure 7.9 Geometry of ridge regression. The likelihood is shown as an ellipse, and the prior is shown as a circle centered on the origin. Based on Figure 3.15 of (Bishop 2006b). Figure generated by `geomRidge`

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

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

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.

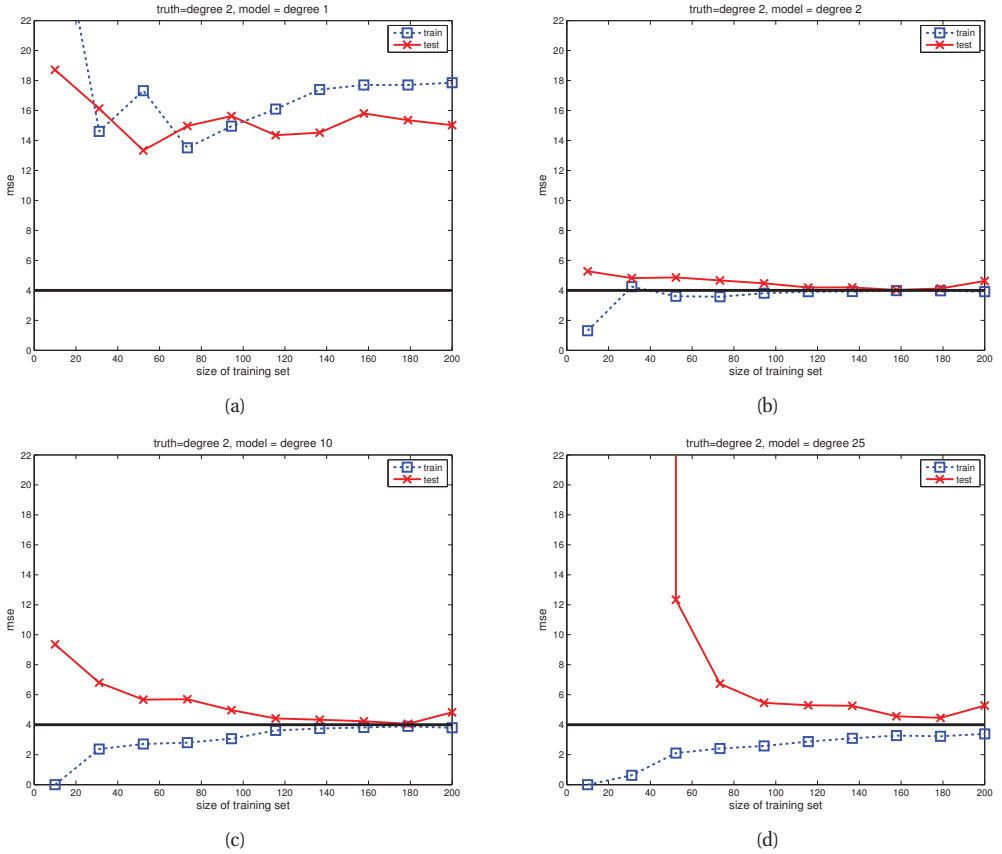


Figure 7.10 MSE on training and test sets vs size of training set, for data generated from a degree 2 polynomial with Gaussian noise of variance $\sigma^2 = 4$. We fit polynomial models of varying degree to this data. (a) Degree 1. (b) Degree 2. (c) Degree 10. (d) Degree 25. Note that for small training 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

$$p(\mathbf{y}|\mathbf{X}, \mathbf{w}, \mu, \sigma^2) = \mathcal{N}(\mathbf{y}|\mu + \mathbf{X}\mathbf{w}, \sigma^2\mathbf{I}_N) \tag{7.52}$$

$$\propto \exp\left(-\frac{1}{2\sigma^2}(\mathbf{y} - \mu\mathbf{1}_N - \mathbf{X}\mathbf{w})^T(\mathbf{y} - \mu\mathbf{1}_N - \mathbf{X}\mathbf{w})\right) \tag{7.53}$$

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

$$p(\mathbf{w}|\mathbf{X}, \mathbf{y}, \sigma^2) \propto \mathcal{N}(\mathbf{w}|\mathbf{w}_0, \mathbf{V}_0)\mathcal{N}(\mathbf{y}|\mathbf{X}\mathbf{w}, \sigma^2\mathbf{I}_N) = \mathcal{N}(\mathbf{w}|\mathbf{w}_N, \mathbf{V}_N) \tag{7.55}$$

$$\mathbf{w}_N = \mathbf{V}_N\mathbf{V}_0^{-1}\mathbf{w}_0 + \frac{1}{\sigma^2}\mathbf{V}_N\mathbf{X}^T\mathbf{y} \tag{7.56}$$

$$\mathbf{V}_N^{-1} = \mathbf{V}_0^{-1} + \frac{1}{\sigma^2}\mathbf{X}^T\mathbf{X} \tag{7.57}$$

$$\mathbf{V}_N = \sigma^2(\sigma^2\mathbf{V}_0^{-1} + \mathbf{X}^T\mathbf{X})^{-1} \tag{7.58}$$

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

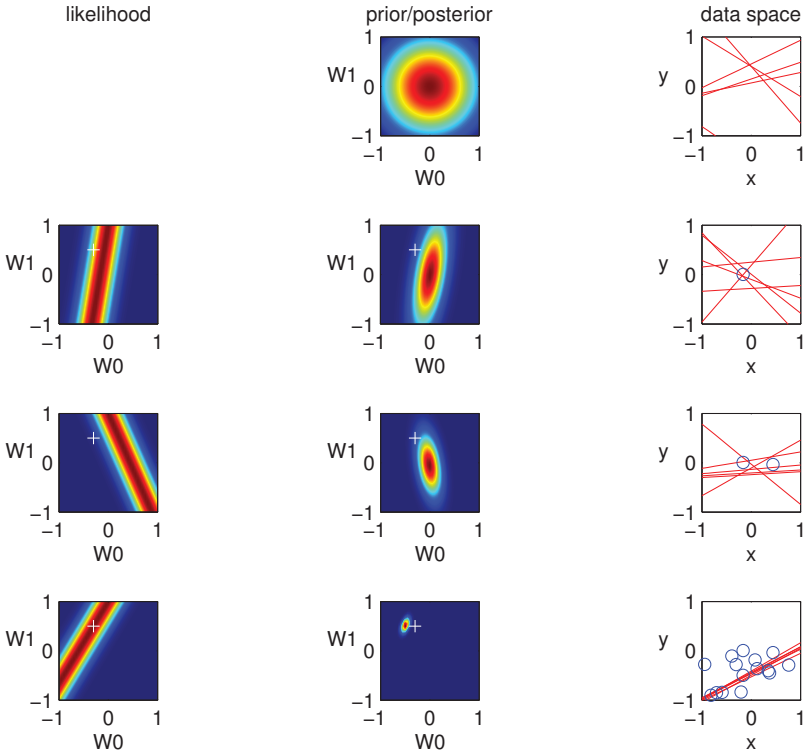


Figure 7.11 Sequential Bayesian updating of a linear regression model $p(y|\mathbf{x}) = \mathcal{N}(y|w_0 x_0 + w_1 x_1, \sigma^2)$. Row 0 represents the prior, row 1 represents the first data point (x_1, y_1), row 2 represents the second data point (x_2, y_2), row 3 represents the 20th data point (x_{20}, 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:

$$p(y|\mathbf{x}, \mathcal{D}, \sigma^2) = \int \mathcal{N}(y|\mathbf{x}^T\mathbf{w}, \sigma^2)\mathcal{N}(\mathbf{w}|\mathbf{w}_N, \mathbf{V}_N)d\mathbf{w} \tag{7.60}$$

$$= \mathcal{N}(y|\mathbf{w}_N^T\mathbf{x}, \sigma_N^2(\mathbf{x})) \tag{7.61}$$

$$\sigma_N^2(\mathbf{x}) = \sigma^2 + \mathbf{x}^T\mathbf{V}_N\mathbf{x} \tag{7.62}$$

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

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

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

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

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

$$p(\mathbf{w}, \sigma^2) = \mathrm{NIG}(\mathbf{w}, \sigma^2|\mathbf{w}_0, \mathbf{V}_0, a_0, b_0) \tag{7.65}$$

$$\triangleq \mathcal{N}(\mathbf{w}|\mathbf{w}_0, \sigma^2\mathbf{V}_0)\mathrm{IG}(\sigma^2|a_0, b_0) \tag{7.66}$$

$$= \frac{b_0^{a_0}}{(2\pi)^{D/2}|\mathbf{V}_0|^{\frac{1}{2}}\Gamma(a_0)} (\sigma^2)^{-(a_0+(D/2)+1)} \tag{7.67}$$

$$\times \exp\left[-\frac{(\mathbf{w}-\mathbf{w}_0)^T\mathbf{V}_0^{-1}(\mathbf{w}-\mathbf{w}_0) + 2b_0}{2\sigma^2}\right] \tag{7.68}$$

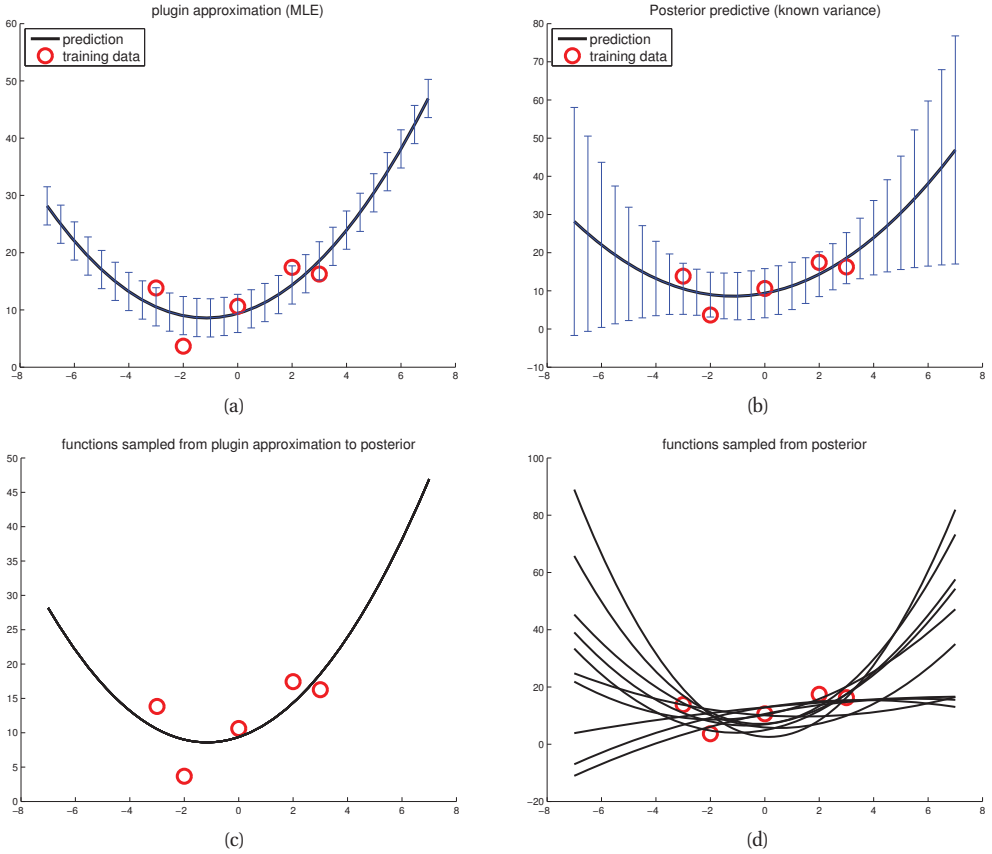


Figure 7.12 (a) Plug-in approximation to predictive density (we plug in the MLE of the parameters). (b) Posterior predictive density, obtained by integrating out the parameters. Black curve is posterior mean, error bars are 2 standard deviations of the posterior predictive density. (c) 10 samples from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

$$
\begin{aligned}
p(\mathbf{w}, \sigma^2|\mathcal{D}) &= \text{NIG}(\mathbf{w}, \sigma^2|\mathbf{w}_N, \mathbf{V}_N, a_N, b_N) && (7.69)\\
\mathbf{w}_N &= \mathbf{V}_N(\mathbf{V}_0^{-1}\mathbf{w}_0 + \mathbf{X}^T\mathbf{y}) && (7.70)\\
\mathbf{V}_N &= (\mathbf{V}_0^{-1} + \mathbf{X}^T\mathbf{X})^{-1} && (7.71)\\
a_N &= a_0 + n/2 && (7.72)\\
b_N &= b_0 + \frac{1}{2}\left(\mathbf{w}_0^T\mathbf{V}_0^{-1}\mathbf{w}_0 + \mathbf{y}^T\mathbf{y} - \mathbf{w}_N^T\mathbf{V}_N^{-1}\mathbf{w}_N\right) && (7.73)
\end{aligned}
$$

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.

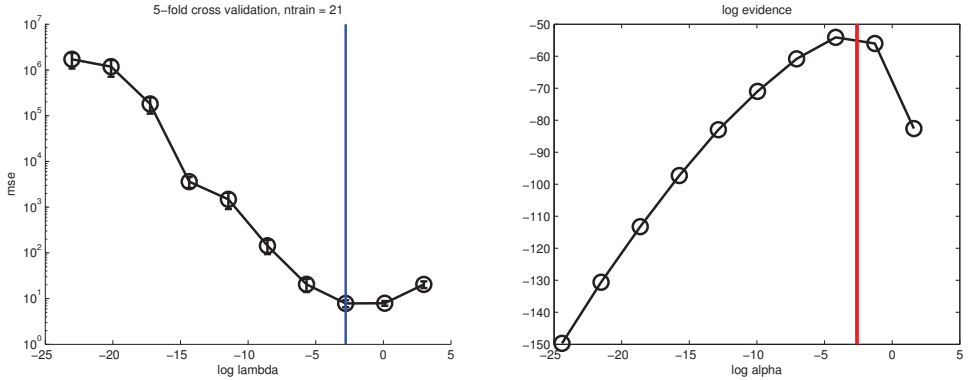
The evidence procedure is also useful when comparing different kinds of models, since it provides a good approximation to the evidence:

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

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

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).



(a) (b)

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

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

is

$$\hat{w_0} = \overline{y} \tag{7.94}$$

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

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

$$\hat{w_0} = \frac{1}{N}\sum_i y_i - \frac{1}{N}\sum_i \mathbf{x}_i^T\mathbf{w} = \overline{y} - \overline{\mathbf{x}}^T\mathbf{w} \tag{7.97}$$

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:

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

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

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:

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

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

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:

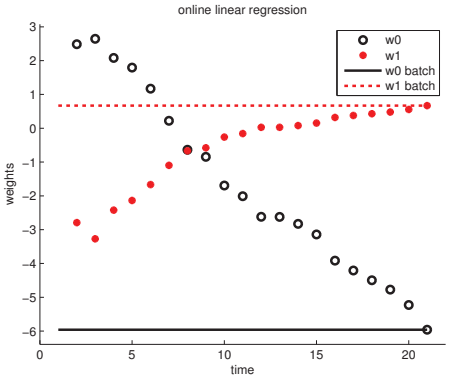


Figure 7.14 Regression coefficients over time. Produced by Exercise 7.7.

```
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 $\boldsymbol{\Sigma}_{XX}$, $\boldsymbol{\Sigma}_{XY}$, $\boldsymbol{\mu}_X$ 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, $p(\mathbf{w}, \sigma^2) = \mathrm{NIG}(\mathbf{w}, \sigma^2 | \mathbf{0}, g(\mathbf{X}^T\mathbf{X})^{-1}, 0, 0)$, the posterior has the following form:

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

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

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

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

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

(7.117)

8 *Logistic regression*

8.1 Introduction

One way to build a probabilistic classifier is to create a joint model of the form $p(y, \mathbf{x})$ and then to condition on $\mathbf{x}$, thereby deriving $p(y|\mathbf{x})$. This is called the generative approach. An alternative approach is to fit a model of the form $p(y|\mathbf{x})$ directly. This is called the **discriminative** approach, and is the approach we adopt in this chapter. In particular, we will assume discriminative models which are linear in the parameters. This will turn out to significantly simplify model fitting, as we will see. In Section 8.6, we compare the generative and discriminative approaches, and in later chapters, we will consider non-linear and non-parametric discriminative models.

8.2 Model specification

As we discussed in Section 1.4.6, logistic regression corresponds to the following binary classification model:

$$p(y|\mathbf{x}, \mathbf{w}) = \mathrm{Ber}(y|\mathrm{sigm}(\mathbf{w}^T\mathbf{x})) \tag{8.1}$$

A 1d example is shown in Figure 1.19(b). Logistic regression can easily be extended to higher-dimensional inputs. For example, Figure 8.1 shows plots of $p(y = 1|\mathbf{x}, \mathbf{w}) = \mathrm{sigm}(\mathbf{w}^T\mathbf{x})$ for 2d input and different weight vectors $\mathbf{w}$. If we threshold these probabilities at 0.5, we induce a linear decision boundary, whose normal (perpendicular) is given by $\mathbf{w}$.

8.3 Model fitting

In this section, we discuss algorithms for estimating the parameters of a logistic regression model.

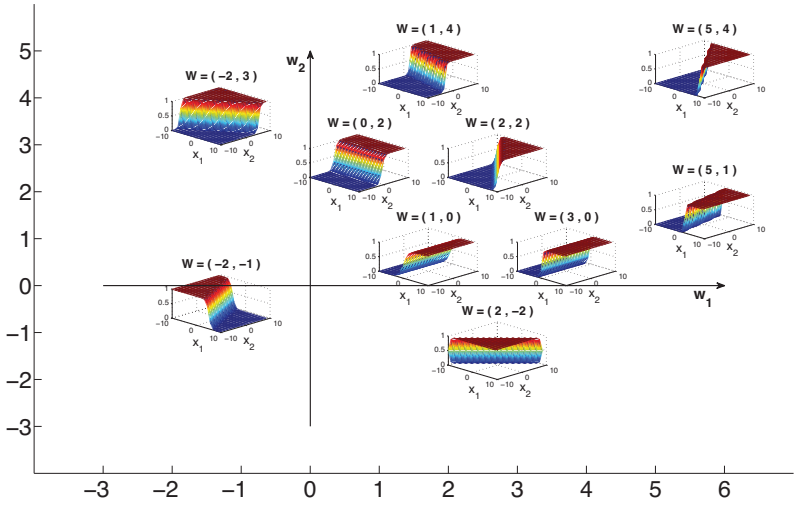


Figure 8.1 Plots of $\text{sigm}(w_1 x_1 + w_2 x_2)$. Here $\mathbf{w} = (w_1, w_2)$ defines the normal to the decision boundary. Points to the right of this have $\text{sigm}(\mathbf{w}^T \mathbf{x}) > 0.5$, and points to the left have $\text{sigm}(\mathbf{w}^T \mathbf{x}) < 0.5$. Based on Figure 39.3 of (MacKay 2003). Figure generated by `sigmoidplot2D`.

8.3.1 MLE

The negative log-likelihood for logistic regression is given by

$$\text{NLL}(\mathbf{w}) = -\sum_{i=1}^{N} \log[\mu_i^{\mathbb{I}(y_i=1)} \times (1-\mu_i)^{\mathbb{I}(y_i=0)}] \tag{8.2}$$

$$= -\sum_{i=1}^{N} [y_i \log \mu_i + (1-y_i) \log(1-\mu_i)] \tag{8.3}$$

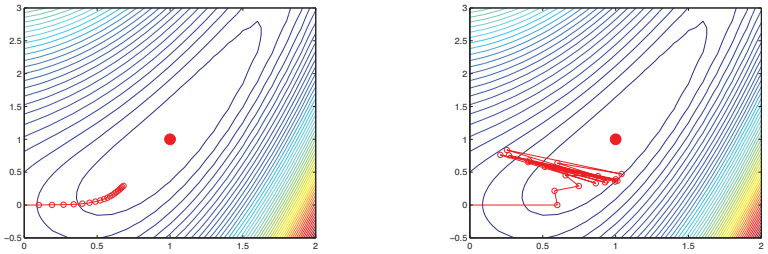
This is also called the **cross-entropy** error function (see Section 2.8.2).

Another way of writing this is as follows. Suppose $\tilde{y}_i \in \{-1, +1\}$ instead of $y_i \in \{0, 1\}$. We have $p(y=1) = \frac{1}{1+\exp(-\mathbf{w}^T \mathbf{x})}$ and $p(y=1) = \frac{1}{1+\exp(+\mathbf{w}^T \mathbf{x})}$. Hence

$$NLL(\mathbf{w}) = \sum_{i=1}^{N} \log(1 + \exp(-\tilde{y}_i \mathbf{w}^T \mathbf{x}_i)) \tag{8.4}$$

Unlike linear regression, we can no longer write down the MLE in closed form. Instead, we need to use an optimization algorithm to compute it. For this, we need to derive the gradient and Hessian.

In the case of logistic regression, one can show (Exercise 8.3) that the gradient and Hessian



(a) (b)

Figure 8.2 Gradient descent on a simple function, starting from $(0, 0)$, for 20 steps, using a fixed learning rate (step size) η. The global minimum is at $(1, 1)$. (a) $\eta = 0.1$. (b) $\eta = 0.6$. Figure generated by `steepestDescentDemo`.

of this are given by the following

$$\mathbf{g} = \frac{d}{d\mathbf{w}} f(\mathbf{w}) = \sum_i (\mu_i - y_i)\mathbf{x}_i = \mathbf{X}^T(\boldsymbol{\mu} - \mathbf{y}) \tag{8.5}$$

$$\mathbf{H} = \frac{d}{d\mathbf{w}} \mathbf{g}(\mathbf{w})^T = \sum_i (\nabla_{\mathbf{w}} \mu_i)\mathbf{x}_i^T = \sum_i \mu_i(1 - \mu_i)\mathbf{x}_i\mathbf{x}_i^T \tag{8.6}$$

$$= \mathbf{X}^T\mathbf{S}\mathbf{X} \tag{8.7}$$

where $\mathbf{S} \triangleq \text{diag}(\mu_i(1 - \mu_i))$. One can also show (Exercise 8.3) that $\mathbf{H}$ is positive definite. Hence the NLL is convex and has a unique global minimum. Below we discuss some methods for finding this minimum.

8.3.2 Steepest descent

Perhaps the simplest algorithm for unconstrained optimization is **gradient descent**, also known as **steepest descent**. This can be written as follows:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k \mathbf{g}_k \tag{8.8}$$

where η_k is the **step size** or **learning rate**. The main issue in gradient descent is: how should we set the step size? This turns out to be quite tricky. If we use a constant learning rate, but make it too small, convergence will be very slow, but if we make it too large, the method can fail to converge at all. This is illustrated in Figure 8.2. where we plot the following (convex) function

$$f(\boldsymbol{\theta}) = 0.5(\theta_1^2 - \theta_2)^2 + 0.5(\theta_1 - 1)^2, \tag{8.9}$$

We arbitrarily decide to start from $(0, 0)$. In Figure 8.2(a), we use a fixed step size of $\eta = 0.1$; we see that it moves slowly along the valley. In Figure 8.2(b), we use a fixed step size of $\eta = 0.6$; we see that the algorithm starts oscillating up and down the sides of the valley and never converges to the optimum.

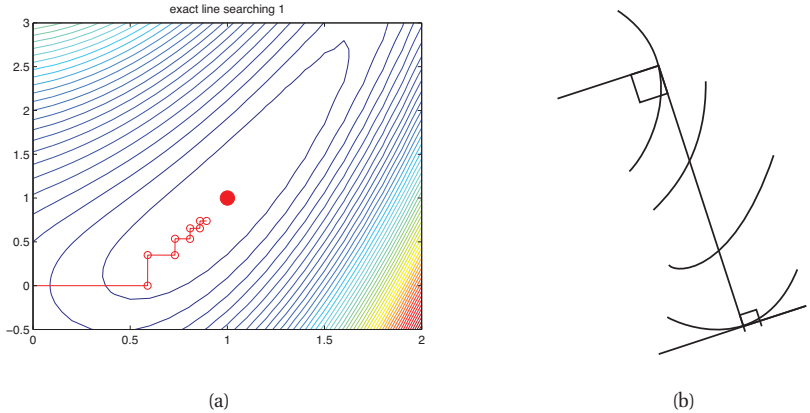


Figure 8.3 (a) Steepest descent on the same function as Figure 8.2, starting from $(0, 0)$, using line search. Figure generated by `steepestDescentDemo`. (b) Illustration of the fact that at the end of a line search (top of picture), the local gradient of the function will be perpendicular to the search direction. Based on Figure 10.6.1 of (Press et al. 1988).

Let us develop a more stable method for picking the step size, so that the method is guaranteed to converge to a local optimum no matter where we start. (This property is called **global convergence**, which should not be confused with convergence to the global optimum!) By Taylor's theorem, we have

$$f(\boldsymbol{\theta} + \eta \mathbf{d}) \approx f(\boldsymbol{\theta}) + \eta \mathbf{g}^T \mathbf{d} \tag{8.10}$$

where $\mathbf{d}$ is our descent direction. So if η is chosen small enough, then $f(\boldsymbol{\theta}+\eta\mathbf{d}) < f(\boldsymbol{\theta})$, since the gradient will be negative. But we don't want to choose the step size η too small, or we will move very slowly and may not reach the minimum. So let us pick η to minimize

$$\phi(\eta) = f(\boldsymbol{\theta}_k + \eta \mathbf{d}_k) \tag{8.11}$$

This is called **line minimization** or **line search**. There are various methods for solving this 1d optimization problem; see (Nocedal and Wright 2006) for details.

Figure 8.3(a) demonstrates that line search does indeed work for our simple problem. However, we see that the steepest descent path with exact line searches exhibits a characteristic **zig-zag** behavior. To see why, note that an exact line search satisfies $\eta_k = \arg\min_{\eta>0} \phi(\eta)$. A necessary condition for the optimum is $\phi'(\eta) = 0$. By the chain rule, $\phi'(\eta) = \mathbf{d}^T\mathbf{g}$, where $\mathbf{g} = f'(\boldsymbol{\theta} + \eta\mathbf{d})$ is the gradient at the end of the step. So we either have $\mathbf{g} = \mathbf{0}$, which means we have found a stationary point, or $\mathbf{g} \perp \mathbf{d}$, which means that exact search stops at a point where the local gradient is perpendicular to the search direction. Hence consecutive directions will be orthogonal (see Figure 8.3(b)). This explains the zig-zag behavior.

One simple heuristic to reduce the effect of zig-zagging is to add a **momentum** term, $(\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1})$, as follows:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k \mathbf{g}_k + \mu_k(\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}) \tag{8.12}$$

where $0 \leq \mu_k \leq 1$ controls the importance of the momentum term. In the optimization community, this is known as the **heavy ball method** (see e.g., (Bertsekas 1999)).

An alternative way to minimize "zig-zagging" is to use the method of **conjugate gradients** (see e.g., (Nocedal and Wright 2006, ch 5) or (Golub and van Loan 1996, Sec 10.2)). This is the method of choice for quadratic objectives of the form $f(\boldsymbol{\theta}) = \boldsymbol{\theta}^T \mathbf{A} \boldsymbol{\theta}$, which arise when solving linear systems. However, non-linear CG is less popular.

8.3.3 Newton's method

Algorithm 8.1: Newton's method for minimizing a strictly convex function

1 Initialize $\boldsymbol{\theta}_0$;
2 **for** $k = 1, 2, \ldots$ *until convergence* **do**
3 Evaluate $\mathbf{g}_k = \nabla f(\boldsymbol{\theta}_k)$;
4 Evaluate $\mathbf{H}_k = \nabla^2 f(\boldsymbol{\theta}_k)$;
5 Solve $\mathbf{H}_k \mathbf{d}_k = -\mathbf{g}_k$ for $\mathbf{d}_k$;
6 Use line search to find stepsize η_k along $\mathbf{d}_k$;
7 $\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k + \eta_k \mathbf{d}_k$;

One can derive faster optimization methods by taking the curvature of the space (i.e., the Hessian) into account. These are called **second order** optimization metods. The primary example is **Newton's algorithm**. This is an iterative algorithm which consists of updates of the form

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k \mathbf{H}_k^{-1} \mathbf{g}_k \tag{8.13}$$

The full pseudo-code is given in Algorithm 2.

This algorithm can be derived as follows. Consider making a second-order Taylor series approximation of $f(\boldsymbol{\theta})$ around $\boldsymbol{\theta}_k$:

$$f_{quad}(\boldsymbol{\theta}) = f_k + \mathbf{g}_k^T (\boldsymbol{\theta} - \boldsymbol{\theta}_k) + \frac{1}{2} (\boldsymbol{\theta} - \boldsymbol{\theta}_k)^T \mathbf{H}_k (\boldsymbol{\theta} - \boldsymbol{\theta}_k) \tag{8.14}$$

Let us rewrite this as

$$f_{quad}(\boldsymbol{\theta}) = \boldsymbol{\theta}^T \mathbf{A} \boldsymbol{\theta} + \mathbf{b}^T \boldsymbol{\theta} + c \tag{8.15}$$

where

$$\mathbf{A} = \frac{1}{2} \mathbf{H}_k, \;\; \mathbf{b} = \mathbf{g}_k - \mathbf{H}_k \boldsymbol{\theta}_k, \;\; c = f_k - \mathbf{g}_k^T \boldsymbol{\theta}_k + \frac{1}{2} \boldsymbol{\theta}_k^T \mathbf{H}_k \boldsymbol{\theta}_k \tag{8.16}$$

The minimum of f_{quad} is at

$$\boldsymbol{\theta} = -\frac{1}{2} \mathbf{A}^{-1} \mathbf{b} = \boldsymbol{\theta}_k - \mathbf{H}_k^{-1} \mathbf{g}_k \tag{8.17}$$

Thus the Newton step $\mathbf{d}_k = -\mathbf{H}_k^{-1} \mathbf{g}_k$ is what should be added to $\boldsymbol{\theta}_k$ to minimize the second order approximation of f around $\boldsymbol{\theta}_k$. See Figure 8.4(a) for an illustration.

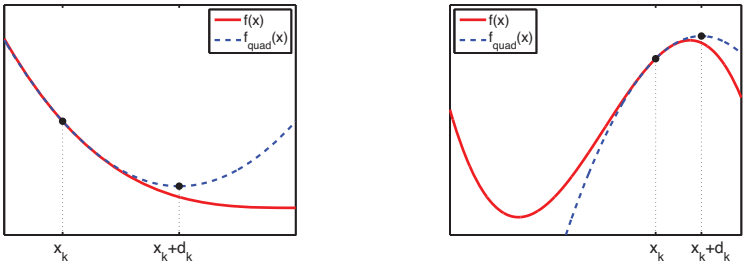


Figure 8.4 Illustration of Newton's method for minimizing a 1d function. (a) The solid curve is the function $f(x)$. The dotted line $f_{quad}(x)$ is its second order approximation at x_k. The Newton step d_k is what must be added to x_k to get to the minimum of $f_{quad}(x)$. Based on Figure 13.4 of (Vandenberghe 2006). Figure generated by `newtonsMethodMinQuad`. (b) Illustration of Newton's method applied to a nonconvex function. We fit a quadratic around the current point x_k and move to its stationary point, $x_{k+1} = x_k + d_k$. Unfortunately, this is a local maximum, not minimum. This means we need to be careful about the extent of our quadratic approximation. Based on Figure 13.11 of (Vandenberghe 2006). Figure generated by `newtonsMethodNonConvex`.

In its simplest form (as listed), Newton's method requires that $\mathbf{H}_k$ be positive definite, which will hold if the function is strictly convex. If not, the objective function is not convex, then $\mathbf{H}_k$ may not be positive definite, so $\mathbf{d}_k = -\mathbf{H}_k^{-1}\mathbf{g}_k$ may not be a descent direction (see Figure 8.4(b) for an example). In this case, one simple strategy is to revert to steepest descent, $\mathbf{d}_k = -\mathbf{g}_k$. The **Levenberg Marquardt** algorithm is an adaptive way to blend between Newton steps and steepest descent steps. This method is widely used when solving nonlinear least squares problems. An alternative approach is this: Rather than computing $\mathbf{d}_k = -\mathbf{H}_k^{-1}\mathbf{g}_k$ directly, we can solve the linear system of equations $\mathbf{H}_k\mathbf{d}_k = -\mathbf{g}_k$ for $\mathbf{d}_k$ using conjugate gradient (CG). If $\mathbf{H}_k$ is not positive definite, we can simply truncate the CG iterations as soon as negative curvature is detected; this is called **truncated Newton**.

8.3.4 Iteratively reweighted least squares (IRLS)

Let us now apply Newton's algorithm to find the MLE for binary logistic regression. The Newton update at iteration $k+1$ for this model is as follows (using $\eta_k = 1$, since the Hessian is exact):

$$\begin{aligned}
\mathbf{w}_{k+1} &= \mathbf{w}_k - \mathbf{H}^{-1}\mathbf{g}_k && (8.18)\\
&= \mathbf{w}_k + (\mathbf{X}^T\mathbf{S}_k\mathbf{X})^{-1}\mathbf{X}^T(\mathbf{y} - \boldsymbol{\mu}_k) && (8.19)\\
&= (\mathbf{X}^T\mathbf{S}_k\mathbf{X})^{-1}\left[(\mathbf{X}^T\mathbf{S}_k\mathbf{X})\mathbf{w}_k + \mathbf{X}^T(\mathbf{y} - \boldsymbol{\mu}_k)\right] && (8.20)\\
&= (\mathbf{X}^T\mathbf{S}_k\mathbf{X})^{-1}\mathbf{X}^T\left[\mathbf{S}_k\mathbf{X}\mathbf{w}_k + \mathbf{y} - \boldsymbol{\mu}_k\right] && (8.21)\\
&= (\mathbf{X}^T\mathbf{S}_k\mathbf{X})^{-1}\mathbf{X}^T\mathbf{S}_k\mathbf{z}_k && (8.22)
\end{aligned}$$

where we have defined the **working response** as

$$\mathbf{z}_k \triangleq \mathbf{X}\mathbf{w}_k + \mathbf{S}_k^{-1}(\mathbf{y} - \boldsymbol{\mu}_k) \quad (8.23)$$

Equation 8.22 is an example of a **weighted least squares problem**, which is a minimizer of

$$\sum_{i=1}^{N} S_{ki}(z_{ki} - \mathbf{w}^T\mathbf{x}_i)^2 \tag{8.24}$$

Since $\mathbf{S}_k$ is a diagonal matrix, we can rewrite the targets in component form (for each case $i = 1 : N$) as

$$z_{ki} = \mathbf{w}_k^T\mathbf{x}_i + \frac{y_i - \mu_{ki}}{\mu_{ki}(1 - \mu_{ki})} \tag{8.25}$$

This algorithm is known as **iteratively reweighted least squares** or **IRLS** for short, since at each iteration, we solve a weighted least squares problem, where the weight matrix $\mathbf{S}_k$ changes at each iteration. See Algorithm 10 for some pseudocode.

Algorithm 8.2: Iteratively reweighted least squares (IRLS)

$\mathbf{w} = \mathbf{0}_D$;
$w_0 = \log(\overline{y}/(1 - \overline{y}))$;
repeat
  $\eta_i = w_0 + \mathbf{w}^T\mathbf{x}_i$;
  $\mu_i = \text{sigm}(\eta_i)$;
  $s_i = \mu_i(1 - \mu_i)$;
  $z_i = \eta_i + \frac{y_i - \mu_i}{s_i}$;
  $\mathbf{S} = \text{diag}(s_{1:N})$;
  $\mathbf{w} = (\mathbf{X}^T\mathbf{S}\mathbf{X})^{-1}\mathbf{X}^T\mathbf{S}\mathbf{z}$;
until *converged*;

8.3.5 Quasi-Newton (variable metric) methods

The mother of all second-order optimization algorithm is Newton's algorithm, which we discussed in Section 8.3.3. Unfortunately, it may be too expensive to compute $\mathbf{H}$ explicitly. **Quasi-Newton** methods iteratively build up an approximation to the Hessian using information gleaned from the gradient vector at each step. The most common method is called **BFGS** (named after its inventors, Broyden, Fletcher, Goldfarb and Shanno), which updates the approximation to the Hessian $\mathbf{B}_k \approx \mathbf{H}_k$ as follows:

$$\mathbf{B}_{k+1} = \mathbf{B}_k + \frac{\mathbf{y}_k\mathbf{y}_k^T}{\mathbf{y}_k^T\mathbf{s}_k} - \frac{(\mathbf{B}_k\mathbf{s}_k)(\mathbf{B}_k\mathbf{s}_k)^T}{\mathbf{s}_k^T\mathbf{B}_k\mathbf{s}_k} \tag{8.26}$$

$$\mathbf{s}_k = \boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1} \tag{8.27}$$

$$\mathbf{y}_k = \mathbf{g}_k - \mathbf{g}_{k-1} \tag{8.28}$$

This is a rank-two update to the matrix, and ensures that the matrix remains positive definite (under certain restrictions on the step size). We typically start with a diagonal approximation, $\mathbf{B}_0 = \mathbf{I}$. Thus BFGS can be thought of as a "diagonal plus low-rank" approximation to the Hessian.

Alternatively, BFGS can iteratively update an approximation to the inverse Hessian, $\mathbf{C}_k \approx \mathbf{H}_k^{-1}$, as follows:

$$\mathbf{C}_{k+1} = \left(\mathbf{I} - \frac{\mathbf{s}_k \mathbf{y}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}\right) \mathbf{C}_k \left(\mathbf{I} - \frac{\mathbf{y}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k}\right) + \frac{\mathbf{s}_k \mathbf{s}_k^T}{\mathbf{y}_k^T \mathbf{s}_k} \tag{8.29}$$

Since storing the Hessian takes $O(D^2)$ space, for very large problems, one can use **limited memory BFGS**, or **L-BFGS**, where $\mathbf{H}_k$ or $\mathbf{H}_k^{-1}$ is approximated by a diagonal plus low rank matrix. In particular, the product $\mathbf{H}_k^{-1}\mathbf{g}_k$ can be obtained by performing a sequence of inner products with $\mathbf{s}_k$ and $\mathbf{y}_k$, using only the m most recent $(\mathbf{s}_k, \mathbf{y}_k)$ pairs, and ignoring older information. The storage requirements are therefore $O(mD)$. Typically $m \sim 20$ suffices for good performance. See (Nocedal and Wright 2006, p177) for more information. L-BFGS is often the method of choice for most unconstrained smooth optimization problems that arise in machine learning (although see Section 8.5).

8.3.6 ℓ_2 regularization

Just as we prefer ridge regression to linear regression, so we should prefer MAP estimation for logistic regression to computing the MLE. In fact, regularization is important in the classification setting even if we have lots of data. To see why, suppose the data is linearly separable. In this case, the MLE is obtained when $||\mathbf{w}|| \to \infty$, corresponding to an infinitely steep sigmoid function, $\mathbb{I}(\mathbf{w}^T\mathbf{x} > w_0)$, also known as a **linear threshold unit**. This assigns the maximal amount of probability mass to the training data. However, such a solution is very brittle and will not generalize well.

To prevent this, we can use ℓ_2 regularization, just as we did with ridge regression. We note that the new objective, gradient and Hessian have the following forms:

$$f'(\mathbf{w}) = \mathrm{NLL}(\mathbf{w}) + \lambda \mathbf{w}^T\mathbf{w} \tag{8.30}$$

$$\mathbf{g}'(\mathbf{w}) = \mathbf{g}(\mathbf{w}) + \lambda \mathbf{w} \tag{8.31}$$

$$\mathbf{H}'(\mathbf{w}) = \mathbf{H}(\mathbf{w}) + \lambda \mathbf{I} \tag{8.32}$$

It is a simple matter to pass these modified equations into any gradient-based optimizer.

8.3.7 Multi-class logistic regression

Now we consider **multinomial logistic regression**, sometimes called a **maximum entropy classifier**. This is a model of the form

$$p(y = c|\mathbf{x}, \mathbf{W}) = \frac{\exp(\mathbf{w}_c^T\mathbf{x})}{\sum_{c'=1}^{C} \exp(\mathbf{w}_{c'}^T\mathbf{x})} \tag{8.33}$$

A slight variant, known as a **conditional logit model**, normalizes over a different set of classes for each data case; this can be useful for modeling choices that users make between different sets of items that are offered to them.

Let us now introduce some notation. Let $\mu_{ic} = p(y_i = c|\mathbf{x}_i, \mathbf{W}) = \mathcal{S}(\boldsymbol{\eta}_i)_c$, where $\boldsymbol{\eta}_i = \mathbf{W}^T\mathbf{x}_i$ is a $C \times 1$ vector. Also, let $y_{ic} = \mathbb{I}(y_i = c)$ be the one-of-C encoding of y_i; thus $\mathbf{y}_i$ is a bit vector, in which the c'th bit turns on iff $y_i = c$. Following (Krishnapuram et al. 2005), let us

set $\mathbf{w}_C = \mathbf{0}$, to ensure identifiability, and define $\mathbf{w} = \text{vec}(\mathbf{W}(:, 1:C-1))$ to be a $D \times (C-1)$ column vector.

With this, the log-likelihood can be written as

$$\ell(\mathbf{W}) = \log \prod_{i=1}^{N} \prod_{c=1}^{C} \mu_{ic}^{y_{ic}} = \sum_{i=1}^{N} \sum_{c=1}^{C} y_{ic} \log \mu_{ic} \tag{8.34}$$

$$= \sum_{i=1}^{N} \left[\left(\sum_{c=1}^{C} y_{ic} \mathbf{w}_c^T \mathbf{x}_i \right) - \log \left(\sum_{c'=1}^{C} \exp(\mathbf{w}_{c'}^T \mathbf{x}_i) \right) \right] \tag{8.35}$$

Define the NLL as

$$f(\mathbf{w}) = -\ell(\mathbf{w}) \tag{8.36}$$

We now proceed to compute the gradient and Hessian of this expression. Since $\mathbf{w}$ is block-structured, the notation gets a bit heavy, but the ideas are simple. It helps to define $\mathbf{A} \otimes \mathbf{B}$ be the **kronecker product** of matrices $\mathbf{A}$ and $\mathbf{B}$. If $\mathbf{A}$ is an $m \times n$ matrix and $\mathbf{B}$ is a $p \times q$ matrix, then $\mathbf{A} \times \mathbf{B}$ is the $mp \times nq$ block matrix

$$\mathbf{A} \otimes \mathbf{B} = \begin{bmatrix} a_{11}\mathbf{B} & \cdots & a_{1n}\mathbf{B} \\ \vdots & \ddots & \vdots \\ a_{m1}\mathbf{B} & \cdots & a_{mn}\mathbf{B} \end{bmatrix} \tag{8.37}$$

Returning to the task at hand, one can show (Exercise 8.4) that the gradient is given by

$$\mathbf{g}(\mathbf{W}) = \nabla f(\mathbf{w}) = \sum_{i=1}^{N} (\boldsymbol{\mu}_i - \mathbf{y}_i) \otimes \mathbf{x}_i \tag{8.38}$$

where $\mathbf{y}_i = (\mathbb{I}(y_i = 1), \ldots, \mathbb{I}(y_i = C-1))$ and $\boldsymbol{\mu}_i(\mathbf{W}) = [p(y_i = 1|\mathbf{x}_i, \mathbf{W}), \ldots, p(y_i = C-1|\mathbf{x}_i, \mathbf{W})]$ are column vectors of length $C-1$, For example, if we have $D = 3$ feature dimensions and $C = 3$ classes, this becomes

$$\mathbf{g}(\mathbf{W}) = \sum_i \begin{pmatrix} (\mu_{i1} - y_{i1})x_{i1} \\ (\mu_{i1} - y_{i1})x_{i2} \\ (\mu_{i1} - y_{i1})x_{i3} \\ (\mu_{i2} - y_{i2})x_{i1} \\ (\mu_{i2} - y_{i2})x_{i2} \\ (\mu_{i2} - y_{i2})x_{i3} \end{pmatrix} \tag{8.39}$$

In other words, for each class c, the derivative for the weights in the c'th column is

$$\nabla_{\mathbf{w}_c} f(\mathbf{W}) = \sum_i (\mu_{ic} - y_{ic}) \mathbf{x}_i \tag{8.40}$$

This has the same form as in the binary logistic regression case, namely an error term times $\mathbf{x}_i$. (This turns out to be a general property of distributions in the exponential family, as we will see in Section 9.3.2.)

One can also show (Exercise 8.4) that the Hessian is the following block structured $D(C-1) \times D(C-1)$ matrix:

$$\mathbf{H}(\mathbf{W}) = \nabla^2 f(\mathbf{w}) = \sum_{i=1}^{N} (\text{diag}(\boldsymbol{\mu}_i) - \boldsymbol{\mu}_i \boldsymbol{\mu}_i^T) \otimes (\mathbf{x}_i \mathbf{x}_i^T) \tag{8.41}$$

For example, if we have 3 features and 3 classes, this becomes

$$\mathbf{H}(\mathbf{W}) = \sum_i \begin{pmatrix} \mu_{i1} - \mu_{i1}^2 & -\mu_{i1}\mu_{i2} \\ -\mu_{i1}\mu_{i2} & \mu_{i2} - \mu_{i2}^2 \end{pmatrix} \otimes \begin{pmatrix} x_{i1}x_{i1} & x_{i1}x_{i2} & x_{i1}x_{i3} \\ x_{i2}x_{i1} & x_{i2}x_{i2} & x_{i2}x_{i3} \\ x_{i3}x_{i1} & x_{i3}x_{i2} & x_{i3}x_{i3} \end{pmatrix} \tag{8.42}$$

$$= \sum_i \begin{pmatrix} (\mu_{i1} - \mu_{i1}^2)\mathbf{X}_i & -\mu_{i1}\mu_{i2}\mathbf{X}_i \\ -\mu_{i1}\mu_{i2}\mathbf{X}_i & (\mu_{i2} - \mu_{i2}^2)\mathbf{X}_i \end{pmatrix} \tag{8.43}$$

where $\mathbf{X}_i = \mathbf{x}_i \mathbf{x}_i^T$. In other words, the block c, c' submatrix is given by

$$\mathbf{H}_{c,c'}(\mathbf{W}) = \sum_i \mu_{ic}(\delta_{c,c'} - \mu_{i,c'})\mathbf{x}_i \mathbf{x}_i^T \tag{8.44}$$

This is also a positive definite matrix, so there is a unique MLE.

Now consider minimizing

$$f'(\mathbf{W}) \triangleq -\log p(\mathcal{D}|\mathbf{w}) - \log p(\mathbf{W}) \tag{8.45}$$

where $p(\mathbf{W}) = \prod_c \mathcal{N}(\mathbf{w}_c|\mathbf{0}, \mathbf{V}_0)$. The new objective, its gradient and Hessian are given by

$$f'(\mathbf{W}) = f(\mathbf{W}) + \frac{1}{2}\sum_c \mathbf{w}_c \mathbf{V}_0^{-1} \mathbf{w}_c \tag{8.46}$$

$$\mathbf{g}'(\mathbf{W}) = \mathbf{g}(\mathbf{W}) + \mathbf{V}_0^{-1}(\sum_c \mathbf{w}_c) \tag{8.47}$$

$$\mathbf{H}'(\mathbf{W}) = \mathbf{H}(\mathbf{W}) + \mathbf{I}_C \otimes \mathbf{V}_0^{-1} \tag{8.48}$$

This can be passed to any gradient-based optimizer to find the MAP estimate. Note, however, that the Hessian has size $O((CD) \times (CD))$, which is C times more row and columns than in the binary case, so limited memory BFGS is more appropriate than Newton's method. See `logregFit` for some Matlab code.

8.4 Bayesian logistic regression

It is natural to want to compute the full posterior over the parameters, $p(\mathbf{w}|\mathcal{D})$, for logistic regression models. This can be useful for any situation where we want to associate confidence intervals with our predictions (e.g., this is necessary when solving contextual bandit problems, discussed in Section 5.7.3.1).

Unfortunately, unlike the linear regression case, this cannot be done exactly, since there is no convenient conjugate prior for logistic regression. We discuss one simple approximation below; some other approaches include MCMC (Section 24.3.3.1), variational inference (Section 21.8.1.1), expectation propagation (Kuss and Rasmussen 2005), etc. For notational simplicity, we stick to binary logistic regression.

8.4.1 Laplace approximation

In this section, we discuss how to make a Gaussian approximation to a posterior distribution. The approximation works as follows. Suppose $\boldsymbol{\theta} \in \mathbb{R}^D$. Let

$$p(\boldsymbol{\theta}|\mathcal{D}) = \frac{1}{Z} e^{-E(\boldsymbol{\theta})} \tag{8.49}$$

where $E(\boldsymbol{\theta})$ is called an **energy function**, and is equal to the negative log of the unnormalized log posterior, $E(\boldsymbol{\theta}) = -\log p(\boldsymbol{\theta}, \mathcal{D})$, with $Z = p(\mathcal{D})$ being the normalization constant. Performing a Taylor series expansion around the mode $\boldsymbol{\theta}^*$ (i.e., the lowest energy state) we get

$$E(\boldsymbol{\theta}) \approx E(\boldsymbol{\theta}^*) + (\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T \mathbf{g} + \frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T \mathbf{H} (\boldsymbol{\theta} - \boldsymbol{\theta}^*) \tag{8.50}$$

where $\mathbf{g}$ is the gradient and $\mathbf{H}$ is the Hessian of the energy function evaluated at the mode:

$$\mathbf{g} \triangleq \nabla E(\boldsymbol{\theta})\big|_{\boldsymbol{\theta}^*}, \; \mathbf{H} \triangleq \frac{\partial^2 E(\boldsymbol{\theta})}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^T} \big|_{\boldsymbol{\theta}^*} \tag{8.51}$$

Since $\boldsymbol{\theta}^*$ is the mode, the gradient term is zero. Hence

$$\hat{p}(\boldsymbol{\theta}|\mathcal{D}) \approx \frac{1}{Z} e^{-E(\boldsymbol{\theta}^*)} \exp\left[-\frac{1}{2}(\boldsymbol{\theta} - \boldsymbol{\theta}^*)^T \mathbf{H} (\boldsymbol{\theta} - \boldsymbol{\theta}^*)\right] \tag{8.52}$$

$$= \mathcal{N}(\boldsymbol{\theta}|\boldsymbol{\theta}^*, \mathbf{H}^{-1}) \tag{8.53}$$

$$Z = p(\mathcal{D}) \approx \int \hat{p}(\boldsymbol{\theta}|\mathcal{D}) d\boldsymbol{\theta} = e^{-E(\boldsymbol{\theta}^*)} (2\pi)^{D/2} |\mathbf{H}|^{-\frac{1}{2}} \tag{8.54}$$

The last line follows from normalization constant of the multivariate Gaussian.

Equation 8.54 is known as the **Laplace approximation** to the marginal likelihood. Therefore Equation 8.52 is sometimes called the the **Laplace approximation** to the posterior. However, in the statistics community, the term "Laplace approximation" refers to a more sophisticated method (see e.g. (Rue et al. 2009) for details). It may therefore be better to use the term "**Gaussian approximation**" to refer to Equation 8.52. A Gaussian approximation is often a reasonable approximation, since posteriors often become more "Gaussian-like" as the sample size increases, for reasons analogous to the central limit theorem. (In physics, there is an analogous technique known as a **saddle point approximation**.)

8.4.2 Derivation of the BIC

We can use the Gaussian approximation to write the log marginal likelihood as follows, dropping irrelevant constants:

$$\log p(\mathcal{D}) \approx \log p(\mathcal{D}|\boldsymbol{\theta}^*) + \log p(\boldsymbol{\theta}^*) - \frac{1}{2} \log |\mathbf{H}| \tag{8.55}$$

The penalization terms which are added to the $\log p(\mathcal{D}|\boldsymbol{\theta}^*)$ are sometimes called the **Occam factor**, and are a measure of model complexity. If we have a uniform prior, $p(\boldsymbol{\theta}) \propto 1$, we can drop the second term, and replace $\boldsymbol{\theta}^*$ with the MLE, $\hat{\boldsymbol{\theta}}$.

We now focus on approximating the third term. We have $\mathbf{H} = \sum_{i=1}^{N} \mathbf{H}_i$, where $\mathbf{H}_i = \nabla\nabla \log p(\mathcal{D}_i|\boldsymbol{\theta})$. Let us approximate each $\mathbf{H}_i$ by a fixed matrix $\hat{\mathbf{H}}$. Then we have

$$\log|\mathbf{H}| = \log|N\hat{\mathbf{H}}| = \log(N^d|\hat{\mathbf{H}}|) = D\log N + \log|\hat{\mathbf{H}}| \tag{8.56}$$

where $D = \dim(\boldsymbol{\theta})$ and we have assumed $\mathbf{H}$ is full rank. We can drop the $\log|\hat{\mathbf{H}}|$ term, since it is independent of N, and thus will get overwhelmed by the likelihood. Putting all the pieces together, we recover the BIC score (Section 5.3.2.4):

$$\log p(\mathcal{D}) \approx \log p(\mathcal{D}|\hat{\boldsymbol{\theta}}) - \frac{D}{2}\log N \tag{8.57}$$

8.4.3 Gaussian approximation for logistic regression

Now let us apply the Gaussian approximation to logistic regression. We will use a a Gaussian prior of the form $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \mathbf{V}_0)$, just as we did in MAP estimation. The approximate posterior is given by

$$p(\mathbf{w}|\mathcal{D}) \approx \mathcal{N}(\mathbf{w}|\hat{\mathbf{w}}, \mathbf{H}^{-1}) \tag{8.58}$$

where $\hat{\mathbf{w}} = \arg\min_{\mathbf{w}} E(\mathbf{w})$, $E(\mathbf{w}) = -(\log p(\mathcal{D}|\mathbf{w}) + \log p(\mathbf{w}))$, and $\mathbf{H} = \nabla^2 E(\mathbf{w})|_{\hat{\mathbf{w}}}$.

As an example, consider the linearly separable 2D data in Figure 8.5(a). There are many parameter settings that correspond to lines that perfectly separate the training data; we show 4 examples. The likelihood surface is shown in Figure 8.5(b), where we see that the likelihood is unbounded as we move up and to the right in parameter space, along a ridge where $w_2/w_1 = 2.35$ (this is indicated by the diagonal line). The reasons for this is that we can maximize the likelihood by driving $||\mathbf{w}||$ to infinity (subject to being on this line), since large regression weights make the sigmoid function very steep, turning it into a step function. Consequently the MLE is not well defined when the data is linearly separable.

To regularize the problem, let us use a vague spherical prior centered at the origin, $\mathcal{N}(\mathbf{w}|\mathbf{0}, 100\mathbf{I})$. Multiplying this spherical prior by the likelihood surface results in a highly skewed posterior, shown in Figure 8.5(c). (The posterior is skewed because the likelihood function "chops off" regions of parameter space (in a "soft" fashion) which disagree with the data.) The MAP estimate is shown by the blue dot. Unlike the MLE, this is not at infinity.

The Gaussian approximation to this posterior is shown in Figure 8.5(d). We see that this is a symmetric distribution, and therefore not a great approximation. Of course, it gets the mode correct (by construction), and it at least represents the fact that there is more uncertainty along the southwest-northeast direction (which corresponds to uncertainty about the orientation of separating lines) than perpendicular to this. Although a crude approximation, this is surely better than approximating the posterior by a delta function, which is what MAP estimation does.

8.4.4 Approximating the posterior predictive

Given the posterior, we can compute credible intervals, perform hypothesis tests, etc., just as we did in Section 7.6.3.3 in the case of linear regression. But in machine learning, interest usually focusses on prediction. The posterior predictive distribution has the form

$$p(y|\mathbf{x}, \mathcal{D}) = \int p(y|\mathbf{x}, \mathbf{w}) p(\mathbf{w}|\mathcal{D}) d\mathbf{w} \tag{8.59}$$

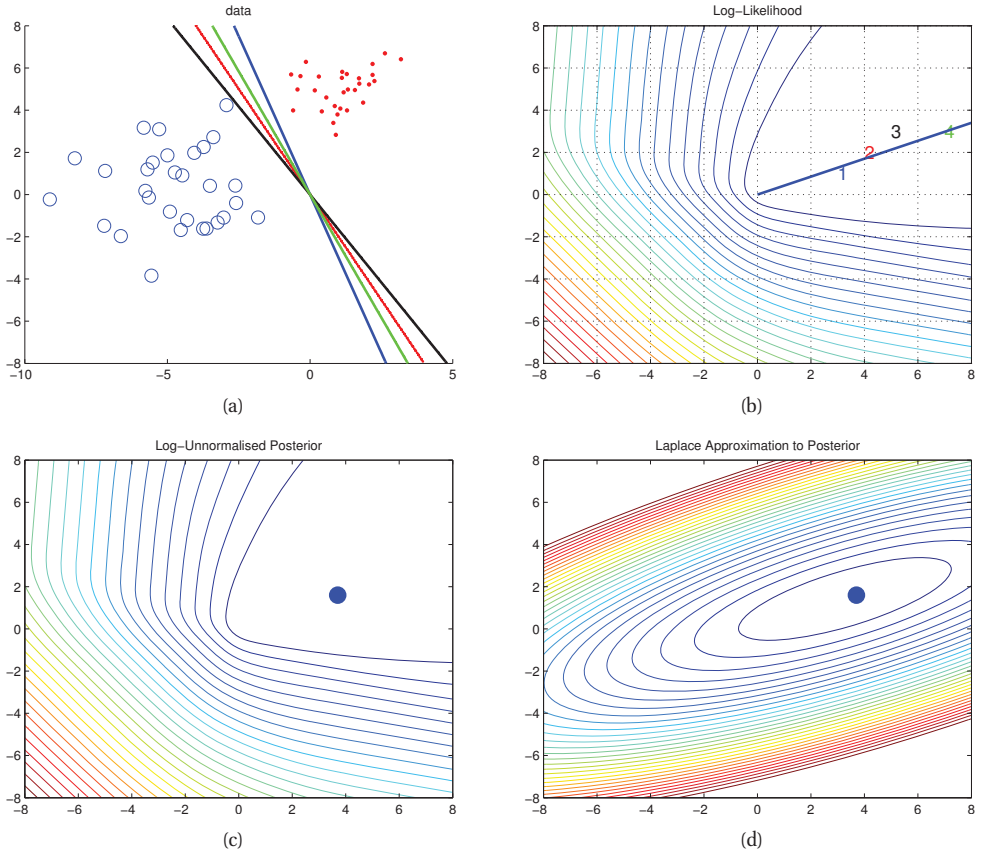


Figure 8.5 (a) Two-class data in 2d. (b) Log-likelihood for a logistic regression model. The line is drawn from the origin in the direction of the MLE (which is at infinity). The numbers correspond to 4 points in parameter space, corresponding to the lines in (a). (c) Unnormalized log posterior (assuming vague spherical prior). (d) Laplace approximation to posterior. Based on a figure by Mark Girolami. Figure generated by `logregLaplaceGirolamiDemo`.

Unfortunately this integral is intractable.

The simplest approximation is the plug-in approximation, which, in the binary case, takes the form

$$p(y = 1|\mathbf{x}, \mathcal{D}) \quad \approx \quad p(y = 1|\mathbf{x}, \mathbb{E}\left[\mathbf{w}\right]) \tag{8.60}$$

where $\mathbb{E}\left[\mathbf{w}\right]$ is the posterior mean. In this context, $\mathbb{E}\left[\mathbf{w}\right]$ is called the **Bayes point**. Of course, such a plug-in estimate underestimates the uncertainty. We discuss some better approximations below.

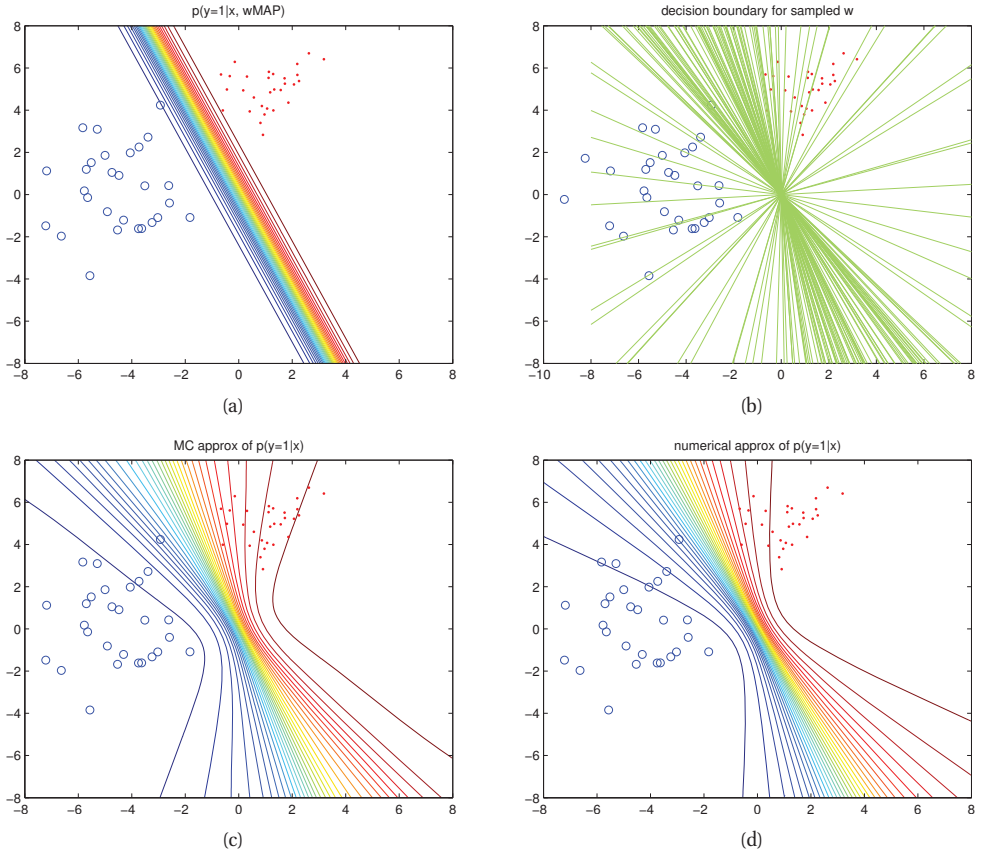


Figure 8.6 Posterior predictive distribution for a logistic regression model in 2d. Top left: contours of $p(y = 1|\mathbf{x}, \hat{\mathbf{w}}_{map})$. Top right: samples from the posterior predictive distribution. Bottom left: Averaging over these samples. Bottom right: moderated output (probit approximation). Based on a figure by Mark Girolami. Figure generated by `logregLaplaceGirolamiDemo`.

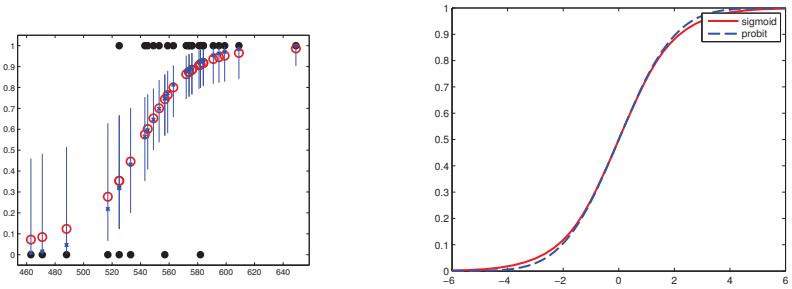
8.4.4.1 Monte Carlo approximation

A better approach is to use a Monte Carlo approximation, as follows:

$$p(y = 1|\mathbf{x}, \mathcal{D}) \approx \frac{1}{S}\sum_{s=1}^{S} \text{sigm}((\mathbf{w}^s)^T\mathbf{x}) \tag{8.61}$$

where $\mathbf{w}^s \sim p(\mathbf{w}|\mathcal{D})$ are samples from the posterior. (This technique can be trivially extended to the multi-class case.) If we have approximated the posterior using Monte Carlo, we can reuse these samples for prediction. If we made a Gaussian approximation to the posterior, we can draw *independent* samples from the Gaussian using standard methods.

Figure 8.6(b) shows samples from the posteiror predictive for our 2d example. Figure 8.6(c)



(a) (b)

Figure 8.7 (a) Posterior predictive density for SAT data. The red circle denotes the posterior mean, the blue cross the posterior median, and the blue lines denote the 5th and 95th percentiles of the predictive distribution. Figure generated by `logregSATdemoBayes`. (b) The logistic (sigmoid) function $\text{sigm}(x)$ in solid red, with the rescaled probit function $\Phi(\lambda x)$ in dotted blue superimposed. Here $\lambda = \sqrt{\pi/8}$, which was chosen so that the derivatives of the two curves match at $x = 0$. Based on Figure 4.9 of (Bishop 2006b). Figure generated by `probitPlot`. Figure generated by `probitRegDemo`.

shows the average of these samples. By averaging over multiple predictions, we see that the uncertainty in the decision boundary "splays out" as we move further from the training data. So although the decision boundary is linear, the posterior predictive density is not linear. Note also that the posterior mean decision boundary is roughly equally far from both classes; this is the Bayesian analog of the large margin principle discussed in Section 14.5.2.2.

Figure 8.7(a) shows an example in 1d. The red dots denote the mean of the posterior predictive evaluated at the training data. The vertical blue lines denote 95% credible intervals for the posterior predictive; the small blue star is the median. We see that, with the Bayesian approach, we are able to model our uncertainty about the probability a student will pass the exam based on his SAT score, rather than just getting a point estimate.

8.4.4.2 Probit approximation (moderated output) *

If we have a Gaussian approximation to the posterior $p(\mathbf{w}|\mathcal{D}) \approx \mathcal{N}(\mathbf{w}|\mathbf{m}_N, \mathbf{V}_N)$, we can also compute a deterministic approximation to the posterior predictive distribution, at least in the binary case. We proceed as follows:

$$p(y=1|\mathbf{x},\mathcal{D}) \approx \int \text{sigm}(\mathbf{w}^T\mathbf{x}) p(\mathbf{w}|\mathcal{D}) d\mathbf{w} = \int \text{sigm}(a)\mathcal{N}(a|\mu_a, \sigma_a^2) da \tag{8.62}$$

$$a \triangleq \mathbf{w}^T\mathbf{x} \tag{8.63}$$

$$\mu_a \triangleq \mathbb{E}[a] = \mathbf{m}_N^T\mathbf{x} \tag{8.64}$$

$$\sigma_a^2 \triangleq \text{var}[a] = \int p(a|\mathcal{D})[a^2 - \mathbb{E}[a^2]] da \tag{8.65}$$

$$= \int p(\mathbf{w}|\mathcal{D})[(\mathbf{w}^T\mathbf{x})^2 - (\mathbf{m}_N^T\mathbf{x})^2] d\mathbf{w} = \mathbf{x}^T\mathbf{V}_N\mathbf{x} \tag{8.66}$$

Thus we see that we need to evaluate the expectation of a sigmoid with respect to a Gaussian. This can be approximated by exploiting the fact that the sigmoid function is similar to the **probit** function, which is given by the cdf of the standard normal:

$$\Phi(a) \triangleq \int_{-\infty}^{a} \mathcal{N}(x|0,1)dx \tag{8.67}$$

Figure 8.7(b) plots the sigmoid and probit functions. We have rescaled the axes so that $\text{sigm}(a)$ has the same slope as $\Phi(\lambda a)$ at the origin, where $\lambda^2 = \pi/8$.

The advantage of using the probit is that one can convolve it with a Gaussian analytically:

$$\int \Phi(\lambda a)\mathcal{N}(a|\mu,\sigma^2)da = \Phi\left(\frac{a}{(\lambda^{-2}+\sigma^2)^{\frac{1}{2}}}\right) \tag{8.68}$$

We now plug in the approximation $\text{sigm}(a) \approx \Phi(\lambda a)$ to both sides of this equation to get

$$\int \text{sigm}(a)\mathcal{N}(a|\mu,\sigma^2)da \approx \text{sigm}(\kappa(\sigma^2)\mu) \tag{8.69}$$

$$\kappa(\sigma^2) \triangleq (1+\pi\sigma^2/8)^{-\frac{1}{2}} \tag{8.70}$$

Applying this to the logistic regression model we get the following expression (first suggested in (Spiegelhalter and Lauritzen 1990)):

$$p(y=1|\mathbf{x},\mathcal{D}) \approx \text{sigm}(\kappa(\sigma_a^2)\mu_a) \tag{8.71}$$

Figure 8.6(d) indicates that this gives very similar results to the Monte Carlo approximation.

Using Equation 8.71 is sometimes called a **moderated output**, since it is less extreme than the plug-in estimate. To see this, note that $0 \leq \kappa(\sigma^2) \leq 1$ and hence

$$\text{sigm}(\kappa(\sigma^2)\mu) \leq \text{sigm}(\mu) = p(y=1|\mathbf{x},\hat{\mathbf{w}}) \tag{8.72}$$

where the inequality is strict if $\mu \neq 0$. If $\mu > 0$, we have $p(y=1|\mathbf{x},\hat{\mathbf{w}}) > 0.5$, but the moderated prediction is always closer to 0.5, so it is less confident. However, the decision boundary occurs whenever $p(y=1|\mathbf{x},\mathcal{D}) = \text{sigm}(\kappa(\sigma^2)\mu) = 0.5$, which implies $\mu = \hat{\mathbf{w}}^T\mathbf{x} = 0$. Hence the decision boundary for the moderated approximation is the same as for the plug-in approximation. So the number of misclassifications will be the same for the two methods, but the log-likelihood will not. (Note that in the multiclass case, taking into account posterior covariance gives different answers than the plug-in approach: see Exercise 3.10.3 of (Rasmussen and Williams 2006).)

8.4.5 Residual analysis (outlier detection) *

It is sometimes useful to detect data cases which are "outliers". This is called **residual analysis** or **case analysis**. In a regression setting, this can be performed by computing $r_i = y_i - \hat{y}_i$, where $\hat{y}_i = \hat{\mathbf{w}}^T\mathbf{x}_i$. These values should follow a $\mathcal{N}(0,\sigma^2)$ distribution, if the modelling assumptions are correct. This can be assessed by creating a **qq-plot**, where we plot the N theoretical quantiles of a Gaussian distribution against the N empirical quantiles of the r_i. Points that deviate from the straightline are potential outliers.

Classical methods, based on residuals, do not work well for binary data, because they rely on asymptotic normality of the test statistics. However, adopting a Bayesian approach, we can just define outliers to be points which which $p(y_i|\hat{y}_i)$ is small, where we typically use $\hat{y}_i = \text{sigm}(\hat{\mathbf{w}}^T\mathbf{x}_i)$. Note that $\hat{\mathbf{w}}$ was estimated from all the data. A better method is to exclude $(\mathbf{x}_i, y_i)$ from the estimate of $\mathbf{w}$ when predicting y_i. That is, we define outliers to be points which have low probability under the cross-validated posterior predictive distribution, defined by

$$p(y_i|\mathbf{x}_i, \mathbf{x}_{-i}, \mathbf{y}_{-i}) = \int p(y_i|\mathbf{x}_i, \mathbf{w}) \prod_{i' \neq i} p(y_{i'}|\mathbf{x}_{i'}, \mathbf{w}) p(\mathbf{w}) d\mathbf{w} \tag{8.73}$$

This can be efficiently approximated by sampling methods (Gelfand 1996). For further discussion of residual analysis in logistic regression models, see e.g.,(Johnson and Albert 1999, Sec 3.4).

8.5 Online learning and stochastic optimization

Traditionally machine learning is performed **offline**, which means we have a **batch** of data, and we optimize an equation of the following form

$$f(\boldsymbol{\theta}) = \frac{1}{N} \sum_{i=1}^{N} f(\boldsymbol{\theta}, \mathbf{z}_i) \tag{8.74}$$

where $\mathbf{z}_i = (\mathbf{x}_i, y_i)$ in the supervised case, or just $\mathbf{x}_i$ in the unsupervised case, and $f(\boldsymbol{\theta}, \mathbf{z}_i)$ is some kind of loss function. For example, we might use

$$f(\boldsymbol{\theta}, \mathbf{z}_i) = -\log p(y_i|\mathbf{x}_i, \boldsymbol{\theta}) \tag{8.75}$$

in which case we are trying to maximize the likelihood. Alternatively, we might use

$$f(\boldsymbol{\theta}, \mathbf{z}_i) = L(y_i, h(\mathbf{x}_i, \boldsymbol{\theta})) \tag{8.76}$$

where $h(\mathbf{x}_i, \boldsymbol{\theta})$ is a prediction function, and $L(y, \hat{y})$ is some other loss function such as squared error or the Huber loss. In frequentist decision theory, the average loss is called the risk (see Section 6.3), so this overall approach is called empirical risk minimization or ERM (see Section 6.5 for details).

However, if we have **streaming data**, we need to perform **online learning**, so we can update our estimates as each new data point arrives rather than waiting until "the end" (which may never occur). And even if we have a batch of data, we might want to treat it like a stream if it is too large to hold in main memory. Below we discuss learning methods for this kind of scenario.[1]

1. A simple implementation trick can be used to speed up batch learning algorithms when applied to data sets that are too large to hold in memory. First note that the naive implementation makes a pass over the data file, from the beginning to end, accumulating the sufficient statistics and gradients as it goes; then an update is performed and the process repeats. Unfortunately, at the end of each pass, the data from the beginning of the file will have been evicted from the cache (since are are assuming it cannot all fit into memory). Rather than going back to the beginning of the file and reloading it, we can simply work backwards from the end of the file, which is already in memory. We then repeat this forwards-backwards pattern over the data. This simple trick is known as **rocking**.

8.5.1 Online learning and regret minimization

Suppose that at each step, "nature" presents a sample $\mathbf{z}_k$ and the "learner" must respond with a parameter estimate $\boldsymbol{\theta}_k$. In the theoretical machine learning community, the objective used in online learning is the **regret**, which is the averaged loss incurred relative to the best we could have gotten in hindsight using a single fixed parameter value:

$$\text{regret}_k \triangleq \frac{1}{k}\sum_{t=1}^{k} f(\boldsymbol{\theta}_t, \mathbf{z}_t) - \min_{\boldsymbol{\theta}^* \in \Theta} \frac{1}{k}\sum_{t=1}^{k} f(\boldsymbol{\theta}_*, \mathbf{z}_t) \tag{8.77}$$

For example, imagine we are investing in the stock-market. Let θ_j be the amount we invest in stock j, and let z_j be the return on this stock. Our loss function is $f(\boldsymbol{\theta}, \mathbf{z}) = -\boldsymbol{\theta}^T\mathbf{z}$. The regret is how much better (or worse) we did by trading at each step, rather than adopting a "buy and hold" strategy using an oracle to choose which stocks to buy.

One simple algorithm for online learning is **online gradient descent** (Zinkevich 2003), which is as follows: at each step k, update the parameters using

$$\boldsymbol{\theta}_{k+1} = \text{proj}_\Theta(\boldsymbol{\theta}_k - \eta_k \mathbf{g}_k) \tag{8.78}$$

where $\text{proj}_\mathcal{V}(\mathbf{v}) = \text{argmin}_{\mathbf{w}\in\mathbf{V}} ||\mathbf{w} - \mathbf{v}||_2$ is the projection of vector $\mathbf{v}$ onto space $\mathcal{V}$, $\mathbf{g}_k = \nabla f(\boldsymbol{\theta}_k, \mathbf{z}_k)$ is the gradient, and η_k is the step size. (The projection step is only needed if the parameter must be constrained to live in a certain subset of $\mathbb{R}^D$. See Section 13.4.3 for details.) Below we will see how this approach to regret minimization relates to more traditional objectives, such as MLE.

There are a variety of other approaches to regret minimization which are beyond the scope of this book (see e.g., Cesa-Bianchi and Lugosi (2006) for details).

8.5.2 Stochastic optimization and risk minimization

Now suppose that instead of minimizing regret with respect to the past, we want to minimize expected loss in the future, as is more common in (frequentist) statistical learning theory. That is, we want to minimize

$$f(\boldsymbol{\theta}) = \mathbb{E}\left[f(\boldsymbol{\theta}, \mathbf{z})\right] \tag{8.79}$$

where the expectation is taken over future data. Optimizing functions where some of the variables in the objective are random is called **stochastic optimization**.[2]

Suppose we receive an infinite stream of samples from the distribution. One way to optimize stochastic objectives such as Equation 8.79 is to perform the update in Equation 8.78 at each step. This is called **stochastic gradient descent** or **SGD** (Nemirovski and Yudin 1978). Since we typically want a single parameter estimate, we can use a running average:

$$\overline{\boldsymbol{\theta}}_k = \frac{1}{k}\sum_{t=1}^{k} \boldsymbol{\theta}_t \tag{8.80}$$

2. Note that in stochastic optimization, the objective is stochastic, and therefore the algorithms will be, too. However, it is also possible to apply stochastic optimization algorithms to deterministic objectives. Examples include simulated annealing (Section 24.6.1) and stochastic gradient descent applied to the empirical risk minimization problem. There are some interesting theoretical connections between online learning and stochastic optimization (Cesa-Bianchi and Lugosi 2006), but this is beyond the scope of this book.

This is called **Polyak-Ruppert averaging**, and can be implemented recursively as follows:

$$\overline{\boldsymbol{\theta}}_k = \overline{\boldsymbol{\theta}}_{k-1} - \frac{1}{k}(\overline{\boldsymbol{\theta}}_{k-1} - \boldsymbol{\theta}_k) \tag{8.81}$$

See e.g., (Spall 2003; Kushner and Yin 2003) for details.

8.5.2.1 Setting the step size

We now discuss some sufficient conditions on the learning rate to guarantee convergence of SGD. These are known as the **Robbins-Monro** conditions:

$$\sum_{k=1}^{\infty} \eta_k = \infty, \ \sum_{k=1}^{\infty} \eta_k^2 < \infty. \tag{8.82}$$

The set of values of η_k over time is called the learning rate **schedule**. Various formulas are used, such as $\eta_k = 1/k$, or the following (Bottou 1998; Bach and Moulines 2011):

$$\eta_k = (\tau_0 + k)^{-\kappa} \tag{8.83}$$

where $\tau_0 \geq 0$ slows down early iterations of the algorithm, and $\kappa \in (0.5, 1]$ controls the rate at which old values of are forgotten.

The need to adjust these tuning parameters is one of the main drawback of stochastic optimization. One simple heuristic (Bottou 2007) is as follows: store an initial subset of the data, and try a range of η values on this subset; then choose the one that results in the fastest decrease in the objective and apply it to all the rest of the data. Note that this may not result in convergence, but the algorithm can be terminated when the performance improvement on a hold-out set plateaus (this is called **early stopping**).

8.5.2.2 Per-parameter step sizes

One drawback of SGD is that it uses the same step size for all parameters. We now briefly present a method known as **adagrad** (short for adaptive gradient) (Duchi et al. 2010), which is similar in spirit to a diagonal Hessian approximation. (See also (Schaul et al. 2012) for a similar approach.) In particular, if $\theta_i(k)$ is parameter i at time k, and $g_i(k)$ is its gradient, then we make an update as follows:

$$\theta_i(k+1) = \theta_i(k) - \eta \frac{g_i(k)}{\tau_0 + \sqrt{s_i(k)}} \tag{8.84}$$

where the diagonal step size vector is the gradient vector squared, summed over all time steps. This can be recursively updated as follows:

$$s_i(k) = s_i(k-1) + g_i(k)^2 \tag{8.85}$$

The result is a per-parameter step size that adapts to the curvature of the loss function. This method was original derived for the regret minimization case, but it can be applied more generally.

8.5.2.3 SGD compared to batch learning

If we don't have an infinite data stream, we can "simulate" one by sampling data points at random from our training set. Essentially we are optimizing Equation 8.74 by treating it as an expectation with respect to the empirical distribution.

Algorithm 8.3: Stochastic gradient descent

```
Initialize θ, η;
repeat
    Randomly permute data;
    for i = 1 : N do
        g = ∇f(θ, z_i);
        θ ← proj_Θ(θ − ηg);
        Update η;
until converged;
```

In theory, we should sample with replacement, although in practice it is usually better to randomly permute the data and sample without replacement, and then to repeat. A single such pass over the entire data set is called an **epoch**. See Algorithm 8 for some pseudocode.

In this offline case, it is often better to compute the gradient of a **mini-batch** of B data cases. If $B = 1$, this is standard SGD, and if $B = N$, this is standard **steepest descent**. Typically $B \sim 100$ is used.

Although a simple first-order method, SGD performs surprisingly well on some problems, especially ones with large data sets (Bottou 2007). The intuitive reason for this is that one can get a fairly good estimate of the gradient by looking at just a few examples. Carefully evaluating precise gradients using large datasets is often a waste of time, since the algorithm will have to recompute the gradient again anyway at the next step. It is often a better use of computer time to have a noisy estimate and to move rapidly through parameter space. As an extreme example, suppose we double the training set by duplicating every example. Batch methods will take twice as long, but online methods will be unaffected, since the direction of the gradient has not changed (doubling the size of the data changes the magnitude of the gradient, but that is irrelevant, since the gradient is being scaled by the step size anyway).

In addition to enhanced speed, SGD is often less prone to getting stuck in shallow local minima, because it adds a certain amount of "noise". Consequently it is quite popular in the machine learning community for fitting models with non-convex objectives, such as neural networks (Section 16.5) and deep belief networks (Section 28.1).

8.5.3 The LMS algorithm

As an example of SGD, let us consider how to compute the MLE for linear regression in an online fashion. We derived the batch gradient in Equation 7.14. The online gradient at iteration k is given by

$$\mathbf{g}_k = \mathbf{x}_i(\boldsymbol{\theta}_k^T \mathbf{x}_i - y_i) \tag{8.86}$$

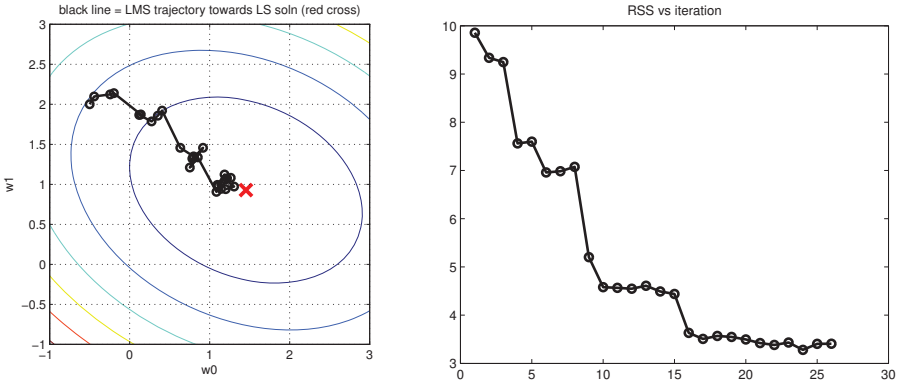


Figure 8.8 Illustration of the LMS algorithm. Left: we start from $\boldsymbol{\theta} = (-0.5, 2)$ and slowly converging to the least squares solution of $\hat{\boldsymbol{\theta}} = (1.45, 0.92)$ (red cross). Right: plot of objective function over time. Note that it does not decrease monotonically. Figure generated by `LMSdemo`.

where $i = i(k)$ is the training example to use at iteration k. If the data set is streaming, we use $i(k) = k$; we shall assume this from now on, for notational simplicity. Equation 8.86 is easy to interpret: it is the feature vector $\mathbf{x}_k$ weighted by the difference between what we predicted, $\hat{y}_k = \boldsymbol{\theta}_k^T \mathbf{x}_k$, and the true response, y_k; hence the gradient acts like an error signal.

After computing the gradient, we take a step along it as follows:

$$\boldsymbol{\theta}_{k+1} = \boldsymbol{\theta}_k - \eta_k(\hat{y}_k - y_k)\mathbf{x}_k \tag{8.87}$$

(There is no need for a projection step, since this is an unconstrained optimization problem.) This algorithm is called the **least mean squares** or **LMS** algorithm, and is also known as the **delta rule**, or the **Widrow-Hoff rule**.

Figure 8.8 shows the results of applying this algorithm to the data shown in Figure 7.2. We start at $\boldsymbol{\theta} = (-0.5, 2)$ and converge (in the sense that $||\boldsymbol{\theta}_k - \boldsymbol{\theta}_{k-1}||_2^2$ drops below a threshold of 10^{-2}) in about 26 iterations.

Note that LMS may require multiple passes through the data to find the optimum. By contrast, the recursive least squares algorithm, which is based on the Kalman filter and which uses second-order information, finds the optimum in a single pass (see Section 18.2.3). See also Exercise 7.7.

8.5.4 The perceptron algorithm

Now let us consider how to fit a binary logistic regression model in an online manner. The batch gradient was given in Equation 8.5. In the online case, the weight update has the simple form

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} - \eta_k \mathbf{g}_i = \boldsymbol{\theta}_{k-1} - \eta_k(\mu_i - y_i)\mathbf{x}_i \tag{8.88}$$

where $\mu_i = p(y_i = 1|\mathbf{x}_i, \boldsymbol{\theta}_k) = \mathbb{E}[y_i|\mathbf{x}_i, \boldsymbol{\theta}_k]$. We see that this has exactly the same form as the LMS algorithm. Indeed, this property holds for all generalized linear models (Section 9.3).

We now consider an approximation to this algorithm. Specifically, let

$$\hat{y}_i = \arg\max_{y \in \{0,1\}} p(y|\mathbf{x}_i, \boldsymbol{\theta}) \tag{8.89}$$

represent the most probable class label. We replace $\mu_i = p(y = 1|\mathbf{x}_i, \boldsymbol{\theta}) = \text{sigm}(\boldsymbol{\theta}^T \mathbf{x}_i)$ in the gradient expression with $\hat{y}_i$. Thus the approximate gradient becomes

$$\mathbf{g}_i \approx (\hat{y}_i - y_i)\mathbf{x}_i \tag{8.90}$$

It will make the algebra simpler if we assume $y \in \{-1, +1\}$ rather than $y \in \{0, 1\}$. In this case, our prediction becomes

$$\hat{y}_i = \text{sign}(\boldsymbol{\theta}^T \mathbf{x}_i) \tag{8.91}$$

Then if $\hat{y}_i y_i = -1$, we have made an error, but if $\hat{y}_i y_i = +1$, we guessed the right label.

At each step, we update the weight vector by adding on the gradient. The key observation is that, if we predicted correctly, then $\hat{y}_i = y_i$, so the (approximate) gradient is zero and we do not change the weight vector. But if $\mathbf{x}_i$ is misclassified, we update the weights as follows: If $\hat{y}_i = 1$ but $y_i = -1$, then the negative gradient is $-(\hat{y}_i - y_i)\mathbf{x}_i = -2\mathbf{x}_i$; and if $\hat{y}_i = -1$ but $y_i = 1$, then the negative gradient is $-(\hat{y}_i - y_i)\mathbf{x}_i = 2\mathbf{x}_i$. We can absorb the factor of 2 into the learning rate η and just write the update, in the case of a misclassification, as

$$\boldsymbol{\theta}_k = \boldsymbol{\theta}_{k-1} + \eta_k y_i \mathbf{x}_i \tag{8.92}$$

Since it is only the sign of the weights that matter, not the magnitude, we will set $\eta_k = 1$. See Algorithm 11 for the pseudocode.

One can show that this method, known as the **perceptron algorithm** (Rosenblatt 1958), will converge, provided the data is linearly separable, i.e., that there exist parameters $\boldsymbol{\theta}$ such that predicting with $\text{sign}(\boldsymbol{\theta}^T \mathbf{x})$ achieves 0 error on the training set. However, if the data is not linearly separable, the algorithm will not converge, and even if it does converge, it may take a long time. There are much better ways to train logistic regression models (such as using proper SGD, without the gradient approximation, or IRLS, discussed in Section 8.3.4). However, the perceptron algorithm is historically important: it was one of the first machine learning algorithms ever derived (by Frank Rosenblatt in 1957), and was even implemented in analog hardware. In addition, the algorithm can be used to fit models where computing marginals $p(y_i|\mathbf{x}, \boldsymbol{\theta})$ is more expensive than computing the MAP output, $\arg\max_{\mathbf{y}} p(\mathbf{y}|\mathbf{x}, \boldsymbol{\theta})$; this arises in some structured-output classification problems. See Section 19.7 for details.

8.5.5 A Bayesian view

Another approach to online learning is to adopt a Bayesian view. This is conceptually quite simple: we just apply Bayes rule recursively:

$$p(\boldsymbol{\theta}|\mathcal{D}_{1:k}) \propto p(\mathcal{D}_k|\boldsymbol{\theta}) p(\boldsymbol{\theta}|\mathcal{D}_{1:k-1}) \tag{8.93}$$

This has the obvious advantage of returning a posterior instead of just a point estimate. It also allows for the online adaptation of hyper-parameters, which is important since cross-validation cannot be used in an online setting. Finally, it has the (less obvious) advantage that it can be

Algorithm 8.4: Perceptron algorithm

Input: linearly separable data set $\mathbf{x}_i \in \mathbb{R}^D$, $y_i \in \{-1, +1\}$ for $i = 1 : N$;
Initialize $\boldsymbol{\theta}_0$;
$k \leftarrow 0$;
repeat
  $k \leftarrow k + 1$;
  $i \leftarrow k \bmod N$;
  if $\hat{y}_i \neq y_i$ **then**
    $\boldsymbol{\theta}_{k+1} \leftarrow \boldsymbol{\theta}_k + y_i \mathbf{x}_i$
  else
    no-op
until *converged*;

quicker than SGD. To see why, note that by modeling the posterior variance of each parameter in addition to its mean, we effectively associate a different learning rate for each parameter (de Freitas et al. 2000), which is a simple way to model the curvature of the space. These variances can then be adapted using the usual rules of probability theory. By contrast, getting second-order optimization methods to work online is more tricky (see e.g., (Schraudolph et al. 2007; Sunehag et al. 2009; Bordes et al. 2009, 2010)).

As a simple example, in Section 18.2.3 we show how to use the Kalman filter to fit a linear regression model online. Unlike the LMS algorithm, this converges to the optimal (offline) answer in a single pass over the data. An extension which can learn a robust non-linear regression model in an online fashion is described in (Ting et al. 2010). For the GLM case, we can use an assumed density filter (Section 18.5.3), where we approximate the posterior by a Gaussian with a diagonal covariance; the variance terms serve as a per-parameter step-size. See Section 18.5.3.2 for details. Another approach is to use particle filtering (Section 23.5); this was used in (Andrieu et al. 2000) for sequentially learning a kernelized linear/logistic regression model.

8.6 Generative vs discriminative classifiers

In Section 4.2.2, we showed that the posterior over class labels induced by Gaussian discriminant analysis (GDA) has exactly the same form as logistic regression, namely $p(y = 1|\mathbf{x}) = \mathrm{sigm}(\mathbf{w}^T\mathbf{x})$. The decision boundary is therefore a linear function of $\mathbf{x}$ in both cases. Note, however, that many generative models can give rise to a logistic regression posterior, e.g., if each class-conditional density is Poisson, $p(x|y = c) = \mathrm{Poi}(x|\lambda_c)$. So the assumptions made by GDA are much stronger than the assumptions made by logistic regression.

A further difference between these models is the way they are trained. When fitting a discriminative model, we usually maximize the conditional log likelihood $\sum_{i=1}^N \log p(y_i|\mathbf{x}_i, \boldsymbol{\theta})$, whereas when fitting a generative model, we usually maximize the joint log likelihood, $\sum_{i=1}^N \log p(y_i, \mathbf{x}_i|\boldsymbol{\theta})$. It is clear that these can, in general, give different results (see Exercise 4.20).

When the Gaussian assumptions made by GDA are correct, the model will need less training data than logistic regression to achieve a certain level of performance, but if the Gaussian

assumptions are incorrect, logistic regression will do better (Ng and Jordan 2002). This is because discriminative models do not need to model the distribution of the features. This is illustrated in Figure 8.10. We see that the class conditional densities are rather complex; in particular, $p(x|y = 1)$ is a multimodal distribution, which might be hard to estimate. However, the class posterior, $p(y = c|x)$, is a simple sigmoidal function, centered on the threshold value of 0.55. This suggests that, in general, discriminative methods will be more accurate, since their "job" is in some sense easier. However, accuracy is not the only important factor when choosing a method. Below we discuss some other advantages and disadvantages of each approach.

8.6.1 Pros and cons of each approach

- **Easy to fit?** As we have seen, it is usually very easy to fit generative classifiers. For example, in Sections 3.5.1.1 and 4.2.4, we show that we can fit a naive Bayes model and an LDA model by simple counting and averaging. By contrast, logistic regression requires solving a convex optimization problem (see Section 8.3.4 for the details), which is much slower.
- **Fit classes separately?** In a generative classifier, we estimate the parameters of each class conditional density independently, so we do not have to retrain the model when we add more classes. In contrast, in discriminative models, all the parameters interact, so the whole model must be retrained if we add a new class. (This is also the case if we train a generative model to maximize a discriminative objective Salojarvi et al. (2005).)
- **Handle missing features easily?** Sometimes some of the inputs (components of $\mathbf{x}$) are not observed. In a generative classifier, there is a simple method for dealing with this, as we discuss in Section 8.6.2. However, in a discriminative classifier, there is no principled solution to this problem, since the model assumes that $\mathbf{x}$ is always available to be conditioned on (although see (Marlin 2008) for some heuristic approaches).
- **Can handle unlabeled training data?** There is much interest in **semi-supervised learning**, which uses unlabeled data to help solve a supervised task. This is fairly easy to do using generative models (see e.g., (Lasserre et al. 2006; Liang et al. 2007)), but is much harder to do with discriminative models.
- **Symmetric in inputs and outputs?** We can run a generative model "backwards", and infer probable inputs given the output by computing $p(\mathbf{x}|y)$. This is not possible with a discriminative model. The reason is that a generative model defines a joint distribution on $\mathbf{x}$ and y, and hence treats both inputs and outputs symmetrically.
- **Can handle feature preprocessing?** A big advantage of discriminative methods is that they allow us to preprocess the input in arbitrary ways, e.g., we can replace $\mathbf{x}$ with $\boldsymbol{\phi}(\mathbf{x})$, which could be some basis function expansion, as illustrated in Figure 8.9. It is often hard to define a generative model on such pre-processed data, since the new features are correlated in complex ways.
- **Well-calibrated probabilities?** Some generative models, such as naive Bayes, make strong independence assumptions which are often not valid. This can result in very extreme posterior class probabilities (very near 0 or 1). Discriminative models, such as logistic regression, are usually better calibrated in terms of their probability estimates.

We see that there are arguments for and against both kinds of models. It is therefore useful to have both kinds in your "toolbox". See Table 8.1 for a summary of the classification and

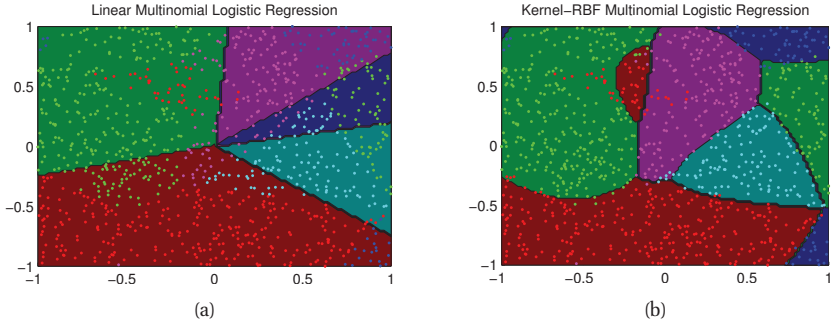


Figure 8.9 (a) Multinomial logistic regression for 5 classes in the original feature space. (b) After basis function expansion, using RBF kernels with a bandwidth of 1, and using all the data points as centers. Figure generated by `logregMultinomKernelDemo`.

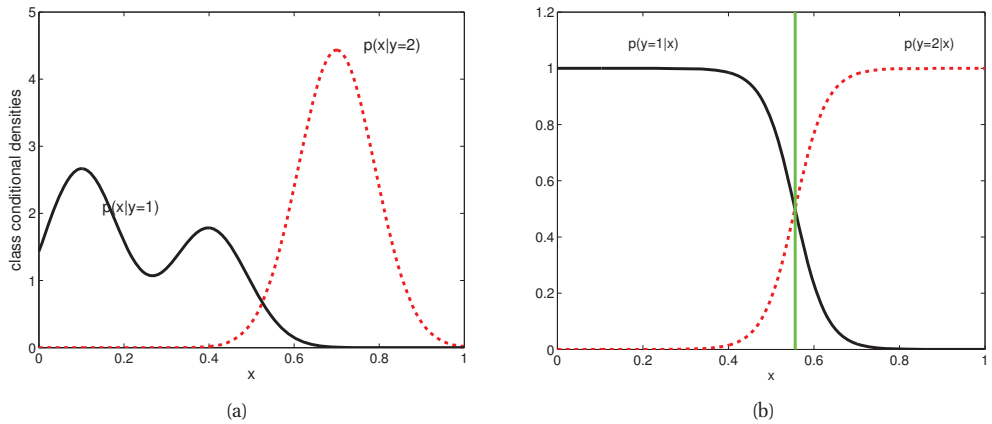


Figure 8.10 The class-conditional densities $p(x|y = c)$ (left) may be more complex than the class posteriors $p(y = c|x)$ (right). Based on Figure 1.27 of (Bishop 2006a). Figure generated by `generativeVsDiscrim`.

regression techniques we cover in this book.

8.6.2 Dealing with missing data

Sometimes some of the inputs (components of $\mathbf{x}$) are not observed; this could be due to a sensor failure, or a failure to complete an entry in a survey, etc. This is called the **missing data problem** (Little. and Rubin 1987). The ability to handle missing data in a principled way is one of the biggest advantages of generative models.

To formalize our assumptions, we can associate a binary response variable $r_i \in \{0, 1\}$, that specifies whether each value $\mathbf{x}_i$ is observed or not. The joint model has the form $p(\mathbf{x}_i, r_i|\boldsymbol{\theta}, \boldsymbol{\phi}) = p(r_i|\mathbf{x}_i, \boldsymbol{\phi})p(\mathbf{x}_i|\boldsymbol{\theta})$, where $\boldsymbol{\phi}$ are the parameters controlling whether the item