CHAPTER 9

Probabilistic models

THE THIRD AND FINAL FAMILY of machine learning models considered in this book are probabilistic models. We have already seen how probabilities can be useful to express a model's expectation about the class of a given instance. For example, a ☞*probability estimation tree* (Section 5.2) attaches a class probability distribution to each leaf of the tree, and each instance that gets filtered down to a particular leaf in a tree model is labelled with that particular class distribution. Similarly, a calibrated linear model translates the distance from the decision boundary into a class probability (Section 7.4). These are examples of what are called *discriminative* probabilistic models. They model the posterior probability distribution $P(Y|X)$, where Y is the target variable and X are the features. That is, given X they return a probability distribution over Y.

The other main class of probabilistic models are called *generative* models. They model the joint distribution $P(Y,X)$ of the target Y and the feature vector X. Once we have access to this joint distribution we can derive any conditional or marginal distribution involving the same variables. In particular, since $P(X) = \sum_y P(Y = y, X)$ it follows that the posterior distribution can be obtained as

$$P(Y|X) = \frac{P(Y,X)}{\sum_y P(Y = y, X)}$$

Alternatively, generative models can be described by the likelihood function $P(X|Y)$, since $P(Y,X) = P(X|Y)P(Y)$ and the target or prior distribution (usually abbreviated to 'prior') can be easily estimated or postulated. Such models are called 'generative' because we can sample from the joint distribution to obtain new data points together with their labels. Alternatively, we can use $P(Y)$ to sample a class and $P(X|Y)$ to sample an instance for that class – this was illustrated for the spam e-mail example on p.29. In contrast, a discriminative model such as a probability estimation tree or a linear classifier models $P(Y|X)$ but not $P(X)$, and hence can be used to label data but not generate it.

Since generative models can do anything that discriminative models do, they may seem preferable. However, they have a number of drawbacks as well. First of all, note that storing the joint distribution requires space exponential in the number of features. This necessitates simplifying assumptions such as independence between features, which may lead to inaccuracies if they are not valid in a particular domain. The most common criticism levied against generative models is that accuracy in modelling $P(X)$ may actually be achieved at the expense of less accurate modelling of $P(Y|X)$. However, the issue is not yet fully understood, and there are certainly situations where knowledge of $P(X)$ provides welcome additional understanding of the domain. For example, we may be less concerned about misclassifying certain instances if they are unlikely according to $P(X)$.

One of the most attractive features of the probabilistic perspective is that it allows us to view learning as a process of reducing uncertainty. For instance, a uniform class prior tells us that, before knowing anything about the instance to be classified, we are maximally uncertain about which class to assign. If the posterior distribution after observing the instance is less uniform, we have reduced our uncertainty in favour of one class or the other. We can repeat this process every time we receive new information, using the posterior obtained in the previous step as the prior for the next step. This process can be applied, in principle, to any unknown quantity that we come across.

Example 9.1 (Spam or not?). Suppose we want to estimate the probability θ that an arbitrary e-mail is spam, so that we can use the appropriate prior distribution. The natural thing to do is to inspect n e-mails, determine the number of spam e-mails d, and set $\hat{\theta} = d/n$; we don't really need any complicated statistics to tell us that. However, while this is the most likely estimate of θ – the maximum a posteriori (MAP) estimate, using the terminology introduced on p.28 – this doesn't mean that other values of θ are completely ruled out. We model this by a probability distribution over θ which is updated each time new information comes in. This is further illustrated in Figure 9.1 for a distribution that is more and more skewed towards spam.

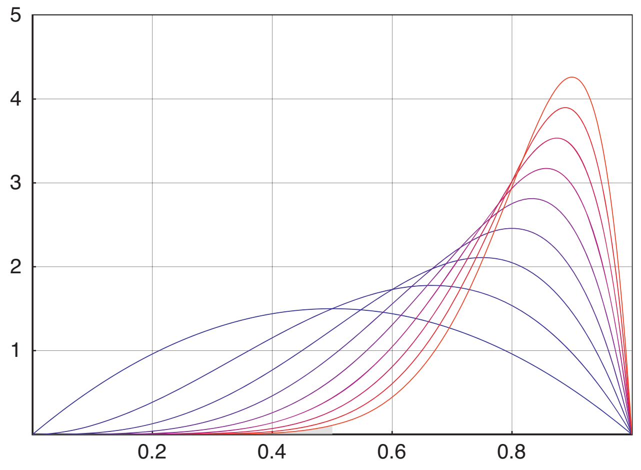

Figure 9.1. Each time we inspect an e-mail, we are reducing our uncertainty regarding the prior spam probability θ. After we inspect two e-mails and observe one spam, the possible θ values are characterised by a symmetric distribution around 1/2. If we inspect a third, fourth, ..., tenth e-mail and each time (except the first one) it is spam, then this distribution narrows and shifts a little bit to the right each time. As you would expect, the distribution for n e-mails reaches its maximum at $\hat{\theta}_{\mathrm{MAP}} = \frac{n-1}{n}$ (e.g., $\hat{\theta}_{\mathrm{MAP}} = 0.8$ for $n = 5$); however, asymmetric distributions like these contain information that cannot be conveyed by single numbers such as the mean or the maximum.

Explicitly modelling the posterior distribution over the parameter θ has a number of advantages that are usually associated with the 'Bayesian' perspective:

- We can precisely characterise the uncertainty that remains about our estimate by quantifying the spread of the posterior distribution.
- We can obtain a generative model for the parameter by sampling from the posterior distribution, which contains much more information than a summary statistic such as the MAP estimate can convey – so, rather than using a single e-mail with $\theta = \theta_{\mathrm{MAP}}$, our generative model can contain a number of e-mails with θ sampled from the posterior distribution.
- We can quantify the probability of statements such as 'e-mails are biased towards ham' (the tiny shaded area in Figure 9.1 demonstrates that after observing one ham and nine spam e-mails this probability is very small, about 0.6%).
- We can use one of these distributions to encode our prior beliefs: e.g., if we believe that the proportions of spam and ham are typically 50–50, we can take the distribution for $n = 2$ (the lowest, symmetric one in Figure 9.1) as our prior.[1]

[1]Statisticians call a prior that has the same mathematical form as a posterior distribution a *conjugate*

The key point is that *probabilities do not have to be interpreted as estimates of relative frequencies, but can carry the more general meaning of (possibly subjective) degrees of belief*. Consequently, we can attach a probability distribution to almost anything: not just features and targets, but also model parameters and even models. For instance, in the example just given we were considering the distribution $P(\theta|D)$, where D represents the data (i.e., the classes of the inspected e-mails).

An important concept related to probabilistic models is *Bayes-optimality*. A classifier is Bayes-optimal if it always assigns $\arg\max_y P^*(Y = y|X = x)$ to an instance x, where P^* denotes the true posterior distribution. Even if we almost never know the true distribution in a practical situation, there are several ways in which we can make this concrete. For example, we can perform experiments with artificially generated data for which we have chosen the true distribution ourselves: this allows us to experimentally evaluate how close the performance of a model is to being Bayes-optimal. Alternatively, the derivation of a probabilistic learning method usually makes certain assumptions about the true distribution, which allows us to prove theoretically that the model will be Bayes-optimal provided these assumptions are met. For example, later on in this chapter we will state the conditions under which the basic linear classifier is Bayes-optimal. The property is therefore best understood as a yardstick by which we measure the performance of probabilistic models.

Since many models discussed in previous chapters are able to estimate class probabilities and hence are discriminative probabilistic models, it is worth pointing out that the choice of a single model, often referred to as *model selection*, does not necessarily lead to Bayes-optimality – even if the model chosen is the one that performs best under the true distribution. To illustrate this, let m^* be the best probability estimation tree we have learned from a sufficient amount of data. Using m^* we would predict $\arg\max_y P(Y = y|M = m^*, X = x)$ for an instance x, where M is a random variable ranging over the model class m^* was chosen from. However, these predictions are not necessarily Bayes-optimal since

$$\begin{aligned}
P(Y|X = x) &= \sum_{m \in M} P(Y, M = m|X = x) && \text{by marginalising over } M \\
&= \sum_{m \in M} P(Y|M = m, X = x)P(M = m|X = x) && \text{by the chain rule} \\
&= \sum_{m \in M} P(Y|M = m, X = x)P(M = m) && \text{by independence of } M \text{ and } X
\end{aligned}$$

Here, $P(M)$ can be interpreted as a posterior distribution over models after seeing the training data (the MAP model is therefore $m^* = \arg\max_m P(M = m)$). The final

prior – in this case we have used the Beta distribution, which is conjugate to the binomial distribution. Conjugate priors not only simplify the mathematics, but also allow more intuitive interpretations: in this case we pretend we have already inspected two e-mails, one of which was spam – a very useful idea that we have in fact already used in the form of the ☞*Laplace correction* in Section 2.3.

expression in the preceding derivation tells us to average the predictions of all models, weighted by their posterior probabilities. Clearly, this distribution is only equal to $P(Y|M = m^*, X = x)$ if $P(M)$ is zero for all models other than m^*, i.e., if we have seen sufficient training data to rule out all but one remaining model. This is obviously unrealistic.[2]

The outline of the chapter is as follows. In Section 9.1 we will see some useful connections between the geometric perspective and the probabilistic viewpoint, which come about when features are normally distributed. This allows us, as already mentioned, to state the conditions under which the basic linear classifier is Bayes-optimal. In Section 9.2 we consider the case of categorical features, leading to the well-known naive Bayes classifier. Section 9.3 revisits the linear classifier from a probabilistic perspective, which results in a new training algorithm explicitly aimed at optimising the posterior probability of the examples. Section 9.4 discusses ways to deal with hidden variables. Finally, in Section 9.5 we briefly look at compression-based learning methods, which can be given a probabilistic interpretation by means of information-theoretic notions.

9.1 The normal distribution and its geometric interpretations

We can draw a connection between probabilistic and geometric models by considering probability distributions defined over Euclidean spaces. The most common such distributions are *normal distributions*, also called *Gaussians*; Background 9.1 recalls the most important facts concerning univariate and multivariate normal distributions. We start by considering the univariate, two-class case. Suppose the values of $x \in \mathbb{R}$ follow a *mixture model*: i.e., each class has its own probability distribution (a *component* of the mixture model). We will assume a Gaussian mixture model, which means that the components of the mixture are both Gaussians. We thus have

$$P(x|\oplus) = \frac{1}{\sqrt{2\pi}\sigma^{\oplus}} \exp\left(-\frac{1}{2}\left[\frac{x-\mu^{\oplus}}{\sigma^{\oplus}}\right]^2\right) \qquad P(x|\ominus) = \frac{1}{\sqrt{2\pi}\sigma^{\ominus}} \exp\left(-\frac{1}{2}\left[\frac{x-\mu^{\ominus}}{\sigma^{\ominus}}\right]^2\right)$$

where $\mu^{\oplus}$ and $\sigma^{\oplus}$ are the mean and standard deviation for the positive class, and $\mu^{\ominus}$ and $\sigma^{\ominus}$ are the mean and standard deviation for the negative class. This gives the following likelihood ratio:

$$\mathrm{LR}(x) = \frac{P(x|\oplus)}{P(x|\ominus)} = \frac{\sigma^{\ominus}}{\sigma^{\oplus}} \exp\left(-\frac{1}{2}\left[\left(\frac{x-\mu^{\oplus}}{\sigma^{\oplus}}\right)^2 - \left(\frac{x-\mu^{\ominus}}{\sigma^{\ominus}}\right)^2\right]\right) \tag{9.1}$$

[2]Note that we do not require the two distributions to be equal, but rather that they reach the same maximum for Y. It is not hard to demonstrate that this, too, is not generally the case.

The univariate normal or Gaussian distribution has the following probability density function:

$$P(x|\mu,\sigma) = \frac{1}{\sqrt{2\pi}\sigma}\exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right) = \frac{1}{E}\exp\left(-\frac{1}{2}\left[\frac{x-\mu}{\sigma}\right]^2\right) = \frac{1}{E}\exp\left(-z^2/2\right), \quad E=\sqrt{2\pi}\sigma$$

The distribution has two parameters: μ, which is the mean or expected value, as well as the median (i.e., the point where the area under the density function is split in half) and the mode (i.e., the point where the density function reaches its maximum); and σ, which is the standard deviation and determines the width of the bell-shaped curve.

$z = (x-\mu)/\sigma$ is the *z-score* associated with x; it measures the number of standard deviations between x and the mean (it has itself mean 0 and standard deviation 1). It follows that $P(x|\mu,\sigma) = \frac{1}{\sigma}P(z|0,1)$, where $P(z|0,1)$ denotes the *standard normal distribution*. In other words, any normal distribution can be obtained from the standard normal distribution by scaling the x-axis with a factor σ, scaling the y-axis with a factor $1/\sigma$ (so the area under the curve remains 1), and translating the origin over μ.

The *multivariate normal distribution* over d-vectors $\mathbf{x} = (x_1,\ldots,x_d)^{\mathrm{T}} \in \mathbb{R}^d$ is

$$P(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{E_d}\exp\left(-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})\right), \quad E_d = (2\pi)^{d/2}\sqrt{|\boldsymbol{\Sigma}|} \tag{9.2}$$

The parameters are the mean vector $\boldsymbol{\mu} = (\mu_1,\ldots,\mu_d)^{\mathrm{T}}$ and the d-by-d covariance matrix $\boldsymbol{\Sigma}$ (see Background 7.2 on p.200). $\boldsymbol{\Sigma}^{-1}$ is the inverse of the covariance matrix, and $|\boldsymbol{\Sigma}|$ is its determinant. The components of $\mathbf{x}$ may be thought of as d features that are possibly correlated.

If $d=1$, then $\boldsymbol{\Sigma} = \sigma^2 = |\boldsymbol{\Sigma}|$ and $\boldsymbol{\Sigma}^{-1} = 1/\sigma^2$, which gives us the univariate Gaussian as a special case. For $d=2$ we have $\boldsymbol{\Sigma} = \begin{pmatrix} \sigma_1^2 & \sigma_{12} \\ \sigma_{12} & \sigma_2^2 \end{pmatrix}$, $|\boldsymbol{\Sigma}| = \sigma_1^2\sigma_2^2 - (\sigma_{12})^2$ and $\boldsymbol{\Sigma}^{-1} = \frac{1}{|\boldsymbol{\Sigma}|}\begin{pmatrix} \sigma_2^2 & -\sigma_{12} \\ -\sigma_{12} & \sigma_1^2 \end{pmatrix}$. Using z-scores we derive the following expression for the bivariate normal distribution:

$$P(x_1,x_2|\mu_1,\mu_2,\sigma_1,\sigma_2,\rho) = \frac{1}{E_2}\exp\left(-\frac{1}{2(1-\rho^2)}(z_1^2+z_2^2-2\rho z_1 z_2)\right), \quad E_2 = 2\pi\sigma_1\sigma_2\sqrt{1-\rho^2} \tag{9.3}$$

where $z_i = (x_i-\mu_i)/\sigma_i$ for $i=1,2$, and $\rho = \sigma_{12}/\sigma_1\sigma_2$ is the *correlation coefficient* between the two features.

The *multivariate standard normal distribution* has $\boldsymbol{\mu} = \mathbf{0}$ (a d-vector with all 0s) and $\boldsymbol{\Sigma} = \mathbf{I}$ (the d-by-d identity matrix), and thus $P(\mathbf{x}|\mathbf{0},\mathbf{I}) = \frac{1}{(2\pi)^{d/2}}\exp\left(-\frac{1}{2}\mathbf{x}\cdot\mathbf{x}\right)$.

Background 9.1. The normal distribution.

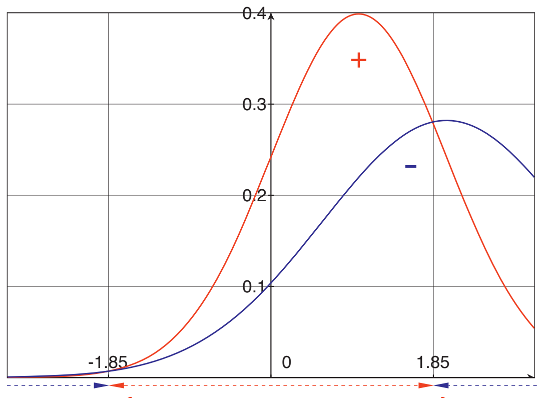

Figure 9.2. If positive examples are drawn from a Gaussian with mean and standard deviation 1 and negatives from a Gaussian with mean and standard deviation 2, then the two distributions cross at $x = \pm 1.85$. This means that the maximum-likelihood region for positives is the closed interval $[-1.85, 1.85]$, and hence the negative region is non-contiguous.

Let's first consider the case that both components have the same standard deviation, i.e., $\sigma^{\oplus} = \sigma^{\ominus} = \sigma$. We can then simplify the exponent in Equation 9.1 as follows:

$$\begin{aligned}
-\frac{1}{2\sigma^2}\left[(x-\mu^{\oplus})^2-(x-\mu^{\ominus})^2\right] &= -\frac{1}{2\sigma^2}\left[x^2-2\mu^{\oplus}x+\mu^{\oplus 2}-(x^2-2\mu^{\ominus}x+\mu^{\ominus 2})\right] \\
&= -\frac{1}{2\sigma^2}\left[-2(\mu^{\oplus}-\mu^{\ominus})x+(\mu^{\oplus 2}-\mu^{\ominus 2})\right] \\
&= \frac{\mu^{\oplus}-\mu^{\ominus}}{\sigma^2}\left[x-\frac{\mu^{\oplus}+\mu^{\ominus}}{2}\right]
\end{aligned}$$

The likelihood ratio can thus be written as $\mathrm{LR}(x) = \exp\big(\gamma(x-\mu)\big)$, with two parameters: $\gamma = (\mu^{\oplus}-\mu^{\ominus})/\sigma^2$ is the difference between the means in proportion to the variance, and $\mu = (\mu^{\oplus}+\mu^{\ominus})/2$ is the midpoint between the two class means. It follows that the maximum-likelihood decision threshold (the value of x such that $\mathrm{LR}(x) = 1$) is $x_{\mathrm{ML}} = \mu$.

If $\sigma^{\oplus} \neq \sigma^{\ominus}$, the x^2 terms in Equation 9.1 do not cancel. This results in two decision boundaries and a non-contiguous decision region for one of the classes.

Example 9.2 (Univariate mixture model with unequal variances). Suppose $\mu^{\oplus} = 1$, $\mu^{\ominus} = 2$ and $\sigma^{\ominus} = 2\sigma^{\oplus} = 2$, then $\mathrm{LR}(x) = 2\exp\big(-[(x-1)^2-(x-2)^2/4]/2\big) = 2\exp\big(3x^2/8\big)$. It follows that the ML decision boundaries are $x = \pm(8/3)\ln 2 = \pm 1.85$. As can be observed in Figure 9.2, these are the points where the two Gaussians cross. In contrast, if $\sigma^{\ominus} = \sigma^{\oplus}$ then we get a single ML decision boundary at $x = 1.5$.

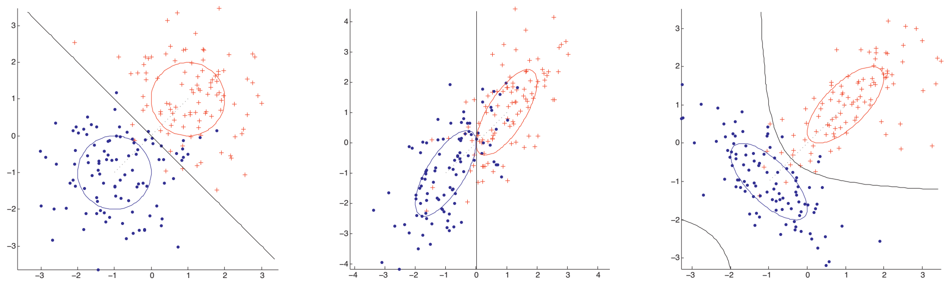

Figure 9.3. **(left)** If the features are uncorrelated and have the same variance, maximum-likelihood classification leads to the basic linear classifier, whose decision boundary is orthogonal to the line connecting the means. **(middle)** As long as the per-class covariance matrices are identical, the Bayes-optimal decision boundary is linear – if we were to decorrelate the features by rotation and scaling, we would again obtain the basic linear classifier. **(right)** Unequal covariance matrices lead to hyperbolic decision boundaries, which means that one of the decision regions is non-contiguous.

Non-contiguous decision regions can also occur in higher-dimensional spaces. The following example demonstrates this for $m = 2$.

Example 9.3 (Bivariate Gaussian mixture). We use Equation 9.3 on p.267 to obtain explicit expressions for the ML decision boundary in the bivariate case. Throughout the example we assume $\mu_1{}^{\oplus} = \mu_2{}^{\oplus} = 1$ and $\mu_1{}^{\ominus} = \mu_2{}^{\ominus} = -1$.

(*i*) If all variances are 1 and both correlations are 0, then the ML decision boundary is given by $(x_1-1)^2+(x_2-1)^2-(x_1+1)^2-(x_2+1)^2 = -2x_1-2x_2-2x_1-2x_2 = 0$, i.e., $x_1+x_2=0$ (Figure 9.3 (left)).

(*ii*) If $\sigma_1{}^{\oplus} = \sigma_1{}^{\ominus} = 1$, $\sigma_2{}^{\oplus} = \sigma_2{}^{\ominus} = \sqrt{2}$ and $\rho^{\oplus} = \rho^{\ominus} = \sqrt{2}/2$, then the ML decision boundary is $(x_1-1)^2+(x_2-1)^2/2-\sqrt{2}(x_1-1)(x_2-1)/\sqrt{2}-(x_1+1)^2-(x_2+1)^2/2+\sqrt{2}(x_1+1)(x_2+1)/\sqrt{2} = -2x_1 = 0$ (Figure 9.3 (middle)).

(*iii*) If all variances are 1 and $\rho^{\oplus} = -\rho^{\ominus} = \rho$, then the ML decision boundary is given by $(x_1-1)^2+(x_2-1)^2-2\rho(x_1-1)(x_2-1)-(x_1+1)^2-(x_2+1)^2-2\rho(x_1+1)(x_2+1) = -4x_1-4x_2-4\rho x_1x_2-4\rho = 0$, i.e., $x_1+x_2+\rho x_1x_2+\rho=0$, which is a hyperbole. Figure 9.3 (right) illustrates this for $\rho = 0.7$. Notice that the bottom left of the instance space is a positive decision region, even though it contains no training examples and it is closer to the negative mean than to the positive mean.

Notice the circles and ellipses in Figure 9.3, which provide a visual summary of the covariance matrix. By projecting the shape for the positive class down to the x-axis we obtain the interval $[\mu_1{}^{\oplus} - \sigma_1{}^{\oplus}, \mu_1{}^{\oplus} + \sigma_1{}^{\oplus}]$ – i.e., one standard deviation around the mean – and similar for the negative class and the y-axis. Three cases can be distinguished: *(i)* both x and y standard deviations are equal and the correlation coefficient is zero, in which case the shape is a circle; *(ii)* the standard deviations are different and the correlation coefficient is zero, which means the shape is an ellipse parallel to the axis with the largest standard deviation; *(iii)* the correlation coefficient is non-zero: the orientation of the ellipse gives the sign of the correlation coefficient, and its width varies with the magnitude of the correlation coefficient.[3] Mathematically, these shapes are defined by setting $f(\mathbf{x})$ in $\frac{1}{E_d}\exp\left(-\frac{1}{2}f(\mathbf{x})\right)$ to 1 and solving for $\mathbf{x}$, in order to capture the points that are one standard deviation away from the mean. For the bivariate case this leads to $(z_1^2 + z_2^2 - 2\rho z_1 z_2) = 1 - \rho^2$, which can be translated into an elliptic equation for x_1 and x_2 by expanding the z-scores. Notice that for $\rho = 0$ this is a circle around the origin, and when $\rho \to 1$ this approaches the line $z_2 = z_1$ (we can't put $\rho = 1$ because this leads to a singular covariance matrix).

In the general multivariate case the condition $(\mathbf{x} - \boldsymbol{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}) = 1$ defines a hyper-ellipse, because $\boldsymbol{\Sigma}^{-1}$ satisfies certain properties.[4] For a standard normal distribution, one-standard-deviation contours lie on a hyper-sphere (a circle in d dimensions) defined by $\mathbf{x} \cdot \mathbf{x} = 1$. A very useful geometric intuition is that, just as hyper-spheres can be turned into arbitrary hyper-ellipses by scaling and rotation, any multivariate Gaussian can be obtained from the standard Gaussian by scaling and rotation (to obtain the desired covariance matrix) and translation (to obtain the desired mean). Conversely, we can turn an arbitrary multivariate Gaussian into a standard normal distribution by translation, rotation and scaling, as was already suggested in Background 1.2 on p.24. This results in decorrelated and normalised features.

The general form of the likelihood ratio can be derived from Equation 9.2 on p.267 as

$$\mathrm{LR}(\mathbf{x}) = \sqrt{\frac{|\boldsymbol{\Sigma}^{\ominus}|}{|\boldsymbol{\Sigma}^{\oplus}|}} \exp\left(-\frac{1}{2}\left[(\mathbf{x} - \boldsymbol{\mu}^{\oplus})^{\mathrm{T}}(\boldsymbol{\Sigma}^{\oplus})^{-1}(\mathbf{x} - \boldsymbol{\mu}^{\oplus}) - (\mathbf{x} - \boldsymbol{\mu}^{\ominus})^{\mathrm{T}}(\boldsymbol{\Sigma}^{\ominus})^{-1}(\mathbf{x} - \boldsymbol{\mu}^{\ominus})\right]\right)$$

where $\boldsymbol{\mu}^{\oplus}$ and $\boldsymbol{\mu}^{\ominus}$ are the class means, and $\boldsymbol{\Sigma}^{\oplus}$ and $\boldsymbol{\Sigma}^{\ominus}$ are the covariance matrices for each class. To understand this a bit better, assume that $\boldsymbol{\Sigma}^{\oplus} = \boldsymbol{\Sigma}^{\ominus} = \mathbf{I}$ (i.e., in each class the features are uncorrelated and have unit variance), then we have

$$\begin{aligned}\mathrm{LR}(\mathbf{x}) &= \exp\left(-\frac{1}{2}\left[(\mathbf{x} - \boldsymbol{\mu}^{\oplus})^{\mathrm{T}}(\mathbf{x} - \boldsymbol{\mu}^{\oplus}) - (\mathbf{x} - \boldsymbol{\mu}^{\ominus})^{\mathrm{T}}(\mathbf{x} - \boldsymbol{\mu}^{\ominus})\right]\right) \\ &= \exp\left(-\frac{1}{2}\left[||\mathbf{x} - \boldsymbol{\mu}^{\oplus}||^2 - ||\mathbf{x} - \boldsymbol{\mu}^{\ominus}||^2\right]\right)\end{aligned}$$

[3]A common mistake is to think that the angle of rotation of the ellipse depends on the correlation coefficient; in fact, it is solely determined by the relative magnitudes of the marginal standard deviations.

[4]Specifically, $\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x}$ defines a hyper-ellipse if $\mathbf{A}$ is symmetric and positive definite. Both properties are satisfied if $\mathbf{A}$ is the inverse of a non-singular covariance matrix.

It follows that $\mathrm{LR}(\mathbf{x}) = 1$ for any $\mathbf{x}$ equidistant from $\boldsymbol{\mu}^{\oplus}$ and $\boldsymbol{\mu}^{\ominus}$. But this means that the ML decision boundary is a straight line at equal distances from the class means – in which we recognise our old friend, the basic linear classifier! In other words, *for uncorrelated, unit-variance Gaussian features, the basic linear classifier is Bayes-optimal*. This is a good example of how a probabilistic viewpoint can justify particular models.

More generally, as long as the per-class covariance matrices are equal, the ML decision boundary will be linear, intersecting $\boldsymbol{\mu}^{\oplus} - \boldsymbol{\mu}^{\ominus}$ in the middle, but not at right angles if the features are correlated. This means that the basic linear classifier is only Bayes-optimal in this case if we first decorrelate and normalise the features. With non-equal class covariances the decision boundary will be hyperbolic. So, the three cases in Figure 9.3 generalise to the multivariate case.

We have now seen several examples of how the normal distribution links the probabilistic and geometric viewpoints. The multivariate normal distribution essentially translates distances into probabilities. This becomes obvious when we plug the definition of ☞*Mahalanobis distance* (Equation 8.1 on p.237) into Equation 9.2:

$$P(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \frac{1}{E_d}\exp\left(-\frac{1}{2}\left(\mathrm{Dis}_M(\mathbf{x},\boldsymbol{\mu}|\boldsymbol{\Sigma})\right)^2\right) \tag{9.4}$$

Similarly, the standard normal distribution translates Euclidean distances into probabilities:

$$P(\mathbf{x}|\mathbf{0},\mathbf{I}) = \frac{1}{(2\pi)^{d/2}}\exp\left(-\frac{1}{2}\left(\mathrm{Dis}_2(\mathbf{x},\mathbf{0})\right)^2\right)$$

Conversely, we see that *the negative logarithm of the Gaussian likelihood can be interpreted as a squared distance*:

$$-\ln P(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) = \ln E_d + \frac{1}{2}\left(\mathrm{Dis}_M(\mathbf{x},\boldsymbol{\mu}|\boldsymbol{\Sigma})\right)^2$$

The intuition is that the logarithm transforms the multiplicative probability scale into an additive scale (which, in the case of Gaussian distributions, corresponds to a squared distance). Since additive scales are often easier to handle, log-likelihoods are a common concept in statistics.

Another example of the link between the geometric and the probabilistic perspective occurs when we consider the question of estimating the parameters of a normal distribution. For example, suppose we want to estimate the mean $\boldsymbol{\mu}$ of a multivariate Gaussian distribution with given covariance matrix $\boldsymbol{\Sigma}$ from a set of data points X. The principle of *maximum-likelihood estimation* states that we should find the value of $\boldsymbol{\mu}$ that maximises the joint likelihood of X. Assuming that the elements of X were independently sampled, the joint likelihood decomposes into a product over the individual data points in X, and the maximum-likelihood estimate can be found as follows:

$$\begin{aligned}
\hat{\boldsymbol{\mu}} &= \arg\max_{\boldsymbol{\mu}} \prod_{\mathbf{x}\in X} P(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma}) \\
&= \arg\max_{\boldsymbol{\mu}} \prod_{\mathbf{x}\in X} \frac{1}{E_d} \exp\left(-\frac{1}{2}\left(\mathrm{Dis}_M(\mathbf{x},\boldsymbol{\mu}|\boldsymbol{\Sigma})\right)^2\right) && \text{using Equation 9.4} \\
&= \arg\min_{\boldsymbol{\mu}} \sum_{\mathbf{x}\in X} \left[\ln E_d + \frac{1}{2}\left(\mathrm{Dis}_M(\mathbf{x},\boldsymbol{\mu}|\boldsymbol{\Sigma})\right)^2\right] && \text{taking negative logarithms} \\
&= \arg\min_{\boldsymbol{\mu}} \sum_{\mathbf{x}\in X} \left(\mathrm{Dis}_M(\mathbf{x},\boldsymbol{\mu}|\boldsymbol{\Sigma})\right)^2 && \text{dropping constant term and factor}
\end{aligned}$$

We thus find that the maximum-likelihood estimate of the mean of a multivariate distribution is the point that minimises the total squared Mahalanobis distance to all points in X. For the identity covariance matrix $\boldsymbol{\Sigma} = \mathbf{I}$ we can replace Mahalanobis distance with Euclidean distance, and by Theorem 8.1 the point minimising total squared Euclidean distance to all points in X is the arithmetic mean $\frac{1}{|X|}\sum_{\mathbf{x}\in X}\mathbf{x}$.

As a final example of how geometric and probabilistic views of the same problem can be strongly connected I will now demonstrate how the ☞*least-squares solution to a linear regression problem* (Section 7.1) can be derived as a maximum-likelihood estimate. For ease of notation we will look at the univariate case discussed in Example 7.1. The starting point is the assumption that our training examples (h_i, y_i) are noisy measurements of true function points $(x_i, f(x_i))$: i.e., $y_i = f(x_i) + \epsilon_i$, where the ϵ_i are independently and identically distributed errors. (Notice the slight change of notation as y_i is now no longer the true function value.) We want to derive the maximum-likelihood estimates $\hat{y}_i$ of $f(x_i)$. We can derive this if we assume a particular noise distribution, for example Gaussian with variance σ_2. It then follows that each y_i is normally distributed with mean $a + bx_i$ and variance σ^2, and thus

$$P(y_i|a,b,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(y_i - (a+bx_i)\right)^2}{2\sigma^2}\right)$$

Since the noise terms ϵ_i are independent for different i, so are the y_i and so the joint probability over all i is simply the product of n of these Gaussians:

$$\begin{aligned}
P(y_1,\dots,y_n|a,b,\sigma^2) &= \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{\left(y_i - (a+bx_i)\right)^2}{2\sigma^2}\right) \\
&= \left(\frac{1}{\sqrt{2\pi\sigma^2}}\right)^n \exp\left(-\frac{\sum_{i=1}^{n}\left(y_i - (a+bx_i)\right)^2}{2\sigma^2}\right)
\end{aligned}$$

For ease of algebraic manipulation we take the negative natural logarithm:

$$-\ln P(y_1,\dots,y_n|a,b,\sigma^2) = \frac{n}{2}\ln 2\pi + \frac{n}{2}\ln\sigma^2 + \frac{\sum_{i=1}^{n}\left(y_i - (a+bx_i)\right)^2}{2\sigma^2}$$

Taking the partial derivatives with respect to a, b and σ^2 and setting to zero in order to maximise the negative log likelihood gives the following three equations:

$$\sum_{i=1}^{n} y_i - (a + bx_i) = 0$$

$$\sum_{i=1}^{n} \left(y_i - (a + bx_i)\right) x_i = 0$$

$$\frac{n}{2}\frac{1}{\sigma^2} - \frac{\sum_{i=1}^{n} \left(y_i - (a + bx_i)\right)^2}{2(\sigma^2)^2} = 0$$

The first two equations are essentially the same as derived in Example 7.1 and give us $\hat{a} = \overline{y} - \hat{b}\overline{x}$ and $\hat{b} = \sigma_{xy}/\sigma_{xx}$, respectively. The third equation tells us that the sum of squared residuals is equal to $n\sigma^2$ and gives the maximum-likelihood estimate of the noise variance as $\left(\sum_{i=1}^{n} \left(y_i - (a + bx_i)\right)^2\right)/n$.

It is reassuring that the probabilistic viewpoint allows us to derive (ordinary) east-squares regression from first principles. On the other hand, a full treatment would require noise on the x-values as well (total least squares), but this complicates the mathematics and does not necessarily have a unique solution. This illustrates that *a good probabilistic treatment of a machine learning problem achieves a balance between solid theoretical foundations and the pragmatism required to obtain a workable solution*.

9.2 Probabilistic models for categorical data

To kill time during long drives to some faraway holiday destination, my sisters and I would often play games involving passing cars. For example, we would ask each other to look out for cars that had a particular colour, were from a particular country or had a particular letter on the numberplate. A binary question such as ‘is the car blue?’ is called a *Bernoulli trial* by statisticians. They are modelled as a binary random variable whose probability of success is fixed over each independent trial. We used a Bernoulli distribution to model the event of an e-mail being ham in Example 9.1. On top of such a random variable, other probability distributions can be built. For example, we may want to guess how many of the next n cars are blue: this is governed by the binomial distribution. Or the task may be to estimate how many cars we need to see until the first Dutch one: this number follows a geometric definition. Background 9.2 will help to refresh your memory regarding the main definitions.

Categorical variables or features (also called discrete or nominal) are ubiquitous in machine learning. Perhaps the most common form of the Bernoulli distribution models whether or not a word occurs in a document. That is, for the i-th word in our vocabulary we have a random variable X_i governed by a Bernoulli distribution. The joint distribution over the *bit vector* $X = (X_1, \ldots, X_k)$ is called a *multivariate Bernoulli distribution*. Variables with more than two outcomes are also common: for example, The *Bernoulli distribution*, named after the Swiss seventeenth century mathematician Jacob Bernoulli, concerns Boolean or binary events with two possible outcomes: success or 1, and failure or 0. A Bernoulli distribution has a single parameter θ which gives the probability of success: hence $P(X=1)=\theta$ and $P(X=0)=1-\theta$. The Bernoulli distribution has expected value $\mathbb{E}[X]=\theta$ and variance $\mathbb{E}\left[(X-\mathbb{E}[X])^2\right]=\theta(1-\theta)$.

The *binomial distribution* arises when counting the number of successes S in n independent Bernoulli trials with the same parameter θ. It is described by

$$P(S=s)=\binom{n}{s}\theta^s(1-\theta)^{n-s} \text{ for } s\in\{0,\dots,n\}$$

This distribution has expected value $\mathbb{E}[S]=n\theta$ and variance $\mathbb{E}\left[(S-\mathbb{E}[S])^2\right]=n\theta(1-\theta)$.

The *categorical distribution* generalises the Bernoulli distribution to $k\geq 2$ outcomes. The parameter of the distribution is a k-vector $\boldsymbol{\theta}=(\theta_1,\dots,\theta_k)$ such that $\sum_{i=1}^k \theta_i=1$.

Finally, the *multinomial distribution* tabulates the outcomes of n independent and identically distributed (i.i.d.) categorical trials. That is, $\mathbf{X}=(X_1,\dots,X_k)$ is a k-vector of integer counts, and

$$P(\mathbf{X}=(x_1,\dots,x_k))=n!\frac{\theta_1^{x_1}}{x_1!}\cdots\frac{\theta_k^{x_k}}{x_k!}$$

with $\sum_{i=1}^k x_i=n$. Notice that setting $n=1$ gives us an alternative way of stating the categorical distribution as $P(\mathbf{X}=(x_1,\dots,x_k))=\theta_1^{x_1}\cdots\theta_k^{x_k}$, with exactly one of the x_i equal to 1 and the rest set to 0. Furthermore, setting $k=2$ gives an alternative expression for the Bernoulli distribution as $P(X=x)=\theta^x(1-\theta)^{1-x}$ for $x\in\{0,1\}$. It is also useful to note that if $\mathbf{X}$ follows a multinomial distribution, then each component X_i follows a binomial distribution with parameter θ_i.

We can estimate the parameters of these distributions by counting in a straightforward way. Suppose $a\ b\ a\ c\ c\ b\ a\ a\ b\ c$ is a sequence of words. We might be interested in individual words being a or not, and interpret the data as coming from 10 i.i.d. Bernoulli trials, which would allow us to estimate $\hat{\theta}_a=4/10=0.4$. This same parameter generates a binomial distribution of the number of occurrences of the word a in similar sequences. Alternatively, we can estimate the parameters of the categorical (word occurrences) and multinomial (word counts) distributions as $\hat{\boldsymbol{\theta}}=(0.4,0.3,0.3)$.

It is almost always a good idea to smooth these distributions by including *pseudo-counts*. Imagine our vocabulary includes the word d but we haven't yet observed it, then a maximum-likelihood estimate would set $\hat{\theta}_d=0$. We can smooth this by adding a virtual occurrence of each word to our observations, leading to $\hat{\boldsymbol{\theta}}'=(5/14,4/14,4/14,1/14)$. In the case of a binomial this is the Laplace correction.

Background 9.2. Probability distributions for categorical data.

every word position in an e-mail corresponds to a categorical variable with k outcomes, where k is the size of the vocabulary. The multinomial distribution manifests itself as a *count vector*: a histogram of the number of occurrences of all vocabulary words in a document. This establishes an alternative way of modelling text documents that allows the number of occurrences of a word to influence the classification of a document.

Both these document models are in common use. Despite their differences, they both assume independence between word occurrences, generally referred to as the *naive Bayes assumption*. In the multinomial document model, this follows from the very use of the multinomial distribution, which assumes that words at different word positions are drawn independently from the same categorical distribution. In the multivariate Bernoulli model we assume that the bits in a bit vector are statistically independent, which allows us to compute the joint probability of a particular bit vector $(x_1,\ldots,x_k)$ as the product of the probabilities of each component $P(X_i = x_i)$. In practice, such word independence assumptions are often not true: if we know that an e-mail contains the word 'Viagra', we can be quite sure that it will also contain the word 'pill'. In any case, the experience is that, while the naive Bayes assumption almost certainly leads to poor probability estimates, it often doesn't harm ranking performance. This means that, provided the classification threshold is chosen with some care, we can usually get good classification performance too.

Using a naive Bayes model for classification

Assume that we have chosen one of the possible distributions to model our data X. In a classification context, we furthermore assume that the distribution depends on the class, so that $P(X|Y = \mathsf{spam})$ and $P(X|Y = \mathsf{ham})$ are different distributions. The more different these two distributions are, the more useful the features X are for classification. Thus, for a specific e-mail x we calculate both $P(X = x|Y = \mathsf{spam})$ and $P(X = x|Y = \mathsf{ham})$, and apply one of several possible decision rules:

maximum likelihood (ML)	– predict $\arg\max_y P(X = x|Y = y)$;
maximum a posteriori (MAP)	– predict $\arg\max_y P(X = x|Y = y)P(Y = y)$;
recalibrated likelihood	– predict $\arg\max_y w_y P(X = x|Y = y)$.

The relation between the first two decision rules is that ML classification is equivalent to MAP classification with a uniform class distribution. The third decision rule generalises the first two in that it replaces the class distribution with a set of weights learned from the data: this makes it possible to correct for estimation errors in the likelihoods, as we shall see later.

Example 9.4 (Prediction using a naive Bayes model). Suppose our vocabulary contains three words a, b and c, and we use a multivariate Bernoulli model for our e-mails, with parameters

$$\boldsymbol{\theta}^{\oplus} = (0.5, 0.67, 0.33) \qquad \boldsymbol{\theta}^{\ominus} = (0.67, 0.33, 0.33)$$

This means, for example, that the presence of b is twice as likely in spam (+), compared with ham.

The e-mail to be classified contains words a and b but not c, and hence is described by the bit vector $\mathbf{x} = (1,1,0)$. We obtain likelihoods

$$P(\mathbf{x}|\oplus) = 0.5 \cdot 0.67 \cdot (1-0.33) = 0.222 \qquad P(\mathbf{x}|\ominus) = 0.67 \cdot 0.33 \cdot (1-0.33) = 0.148$$

The ML classification of $\mathbf{x}$ is thus spam. In the case of two classes it is often convenient to work with likelihood ratios and odds. The likelihood ratio can be calculated as $\frac{P(\mathbf{x}|\oplus)}{P(\mathbf{x}|\ominus)} = \frac{0.5}{0.67}\frac{0.67}{0.33}\frac{1-0.33}{1-0.33} = 3/2 > 1$. This means that the MAP classification of $\mathbf{x}$ is also spam if the prior odds are more than $2/3$, but ham if they are less than that. For example, with 33% spam and 67% ham the prior odds are $\frac{P(\oplus)}{P(\ominus)} = \frac{0.33}{0.67} = 1/2$, resulting in a posterior odds of $\frac{P(\oplus|\mathbf{x})}{P(\ominus|\mathbf{x})} = \frac{P(\mathbf{x}|\oplus)}{P(\mathbf{x}|\ominus)}\frac{P(\oplus)}{P(\ominus)} = 3/2 \cdot 1/2 = 3/4 < 1$. In this case the likelihood ratio for $\mathbf{x}$ is not strong enough to push the decision away from the prior.

Alternatively, we can employ a multinomial model. The parameters of a multinomial establish a distribution over the words in the vocabulary, say

$$\boldsymbol{\theta}^{\oplus} = (0.3, 0.5, 0.2) \qquad \boldsymbol{\theta}^{\ominus} = (0.6, 0.2, 0.2)$$

The e-mail to be classified contains three occurrences of word a, one single occurrence of word b and no occurrences of word c, and hence is described by the count vector $\mathbf{x} = (3,1,0)$. The total number of vocabulary word occurrences is $n = 4$. We obtain likelihoods

$$P(\mathbf{x}|\oplus) = 4!\frac{0.3^3}{3!}\frac{0.5^1}{1!}\frac{0.2^0}{0!} = 0.054 \qquad P(\mathbf{x}|\ominus) = 4!\frac{0.6^3}{3!}\frac{0.2^1}{1!}\frac{0.2^0}{0!} = 0.1728$$

The likelihood ratio is $\left(\frac{0.3}{0.6}\right)^3\left(\frac{0.5}{0.2}\right)^1\left(\frac{0.2}{0.2}\right)^0 = 5/16$. The ML classification of $\mathbf{x}$ is thus ham, the opposite of the multivariate Bernoulli model. This is mainly because of the three occurrences of word a, which provide strong evidence for ham.

Notice how the likelihood ratio for the multivariate Bernoulli model is a product of factors $\theta_i^{\oplus}/\theta_i^{\ominus}$ if $x_i = 1$ in the bit vector to be classified, and $(1-\theta_i^{\oplus})/(1-\theta_i^{\ominus})$ if $x_i = 0$. For the multinomial model the factors are $\left(\theta_i^{\oplus}/\theta_i^{\ominus}\right)^{x_i}$. One consequence of this is that the multinomial model only takes the presence of words into account, whereas in the multivariate Bernoulli model absent words can make a difference. In the previous example, not containing word b corresponds to a factor of $(1-0.67)/(1-0.33) = 1/2$ in the likelihood ratio. The other main difference between the two models is that multiple occurrences of words are treated like duplicated features in the multinomial model, through the exponential 'weight' x_i. This becomes clearer by taking the logarithm of the likelihood ratio, which is $\sum_i x_i(\ln\theta_i^{\oplus} - \ln\theta_i^{\ominus})$: this expression is linear in $\ln\theta_i^{\oplus}$ and $\ln\theta_i^{\ominus}$ with x_i as weights. Notice that this does not mean that naive Bayes classifiers are linear in the sense discussed in Chapter 7 unless we can demonstrate a linear relationship between $\ln\theta$ and the corresponding feature value. But we can say that naive Bayes models are linear in a particular space (the 'log-odds' space) obtained by applying a well-defined transformation to the features. We will return to this point when we discuss ☞*feature calibration* in Section 10.2.

The fact that the joint likelihood ratio of a naive Bayes model factorises as a product of likelihood ratios of individual words is a direct consequence of the naive Bayes assumption. In other words, the learning task decomposes into univariate tasks, one for each word in the vocabulary. We have encountered such a decomposition before when we discussed ☞*multivariate linear regression* in Section 7.1. There, we saw an example of how ignoring feature correlation could be harmful. Can we come up with similar examples for naive Bayes classifiers? Consider the situation when a particular word occurs twice in the vocabulary. In that case, we have the same factor occurring twice in the product for the likelihood ratio, and are effectively giving the word in question twice the weight of other words. While this is an extreme example, such double-counting does have noticeable effects in practice. I previously gave the example that if a spam e-mail contains the word 'Viagra', it is also expected to contain the word 'pill', so seeing the two words together should not give much more evidence for spam than seeing the first word on its own, and the likelihood ratio for the two words should not be much higher than that of the first word. However, multiplying two likelihood ratios larger than 1 will result in an even larger likelihood ratio. As a result, the probability estimates of a naive Bayes classifier are often pushed too far towards 0 or 1.

This may not seem such a big deal if we are only interested in classification, and not in the probability estimates as such. However, *an often overlooked consequence of having uncalibrated probability estimates such as those produced by naive Bayes is that both the ML and MAP decision rules become inadequate*. Unless we have evidence that the model assumptions are satisfied, the only sensible thing to do in this case is to invoke the *recalibrated likelihood decision rule*, which requires one to learn a weight vector

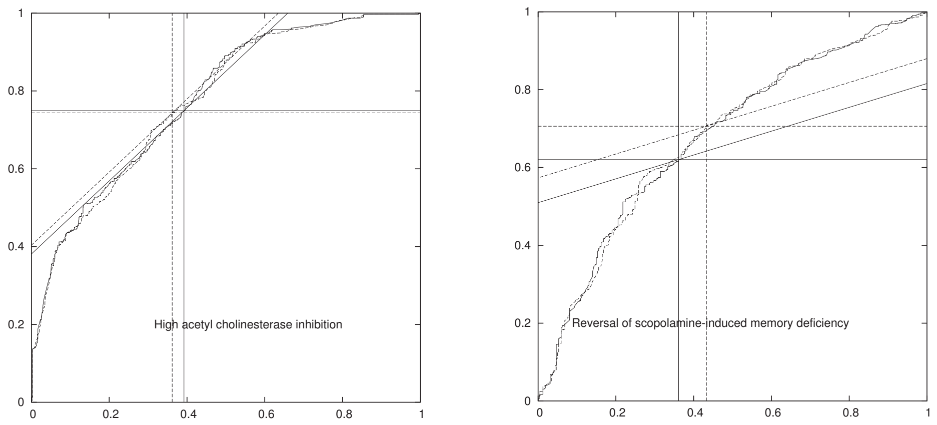

Figure 9.4. **(left)** ROC curves produced by two naive Bayes classifiers (solid line: a variant of the multivariate Bernoulli model; dashed line: a variant of the multinomial model). Both models have similar ranking performance and yield almost the same – more or less optimal – MAP decision threshold. **(right)** On a different data set from the same domain, the multinomial model's MAP threshold is slightly better, hinting at somewhat better calibrated probability estimates. But since the slope of the accuracy isometrics indicates that there are about four positives for every negative, the optimal decision rule is in fact to always predict positive.

over the classes, in order to correct for the estimation errors in the likelihoods. Specifically, we want to find weights w_i such that predicting $\arg\max_y w_y P(X=x|Y=y)$ results in the smallest possible loss – e.g., the number of misclassified examples – over a test set. For two classes this can be solved by the same procedure we considered for ☞*turning rankers into classifiers* in Section 2.2. To see this, notice that for two classes the recalibrated likelihood decision rule can be rewritten as

- ☞ predict positive if $w^{\oplus} P(X=x|Y=\oplus) > w^{\ominus} P(X=x|Y=\ominus)$ and negative otherwise; which is equivalent to

- ☞ predict positive if $P(X=x|Y=\oplus)/P(X=x|Y=\ominus) > w^{\ominus}/w^{\oplus}$ and negative otherwise

This demonstrates that in the two-class case we really have just one degree of freedom, as multiplying the weights by a constant does not affect the decisions. In other words, what we are interested in is finding the best threshold $t = w^{\ominus}/w^{\oplus}$ on the likelihood ratio, which is essentially the same problem as finding the best operating point on an ROC curve. The solution is given by the point on the highest accuracy isometric. Figure 9.4 illustrates this on two real-life data sets: in the left figure we see that the MAP decision threshold is more or less optimal, whereas in the right figure the optimal point is in the top right-hand corner.

For more than two classes, finding a globally optimal weight vector is computationally intractable, which means that we need to resort to a heuristic method. In Section 3.1 such a method was demonstrated for three classes. The idea is to fix the weights one by one, using some ordering of the classes. That is, we use the two-class procedure to optimally separate the i-th class from the previous $i-1$ classes.

Training a naive Bayes model

Training a probabilistic model usually involves estimating the parameters of the distributions used in the model. The parameter of a Bernoulli distribution can be estimated by counting the number of successes d in n trials and setting $\hat{\theta} = d/n$. In other words, we count, for each class, how many e-mails contain the word in question. Such relative frequency estimates are usually smoothed by including *pseudo-counts*, representing the outcome of virtual trials according to some fixed distributions. In the case of a Bernoulli distribution the most common smoothing operation is the Laplace correction, which involves two virtual trials, one of which results in success and the other in failure. Consequently, the relative frequency estimate is changed to $(d+1)/(n+2)$. From a Bayesian perspective this amounts to adopting a uniform prior, representing our initial belief that success and failure are equally likely. If appropriate, we can strengthen the influence of the prior by including a larger number of virtual trials, which means that more data is needed to move the estimate away from the prior. For a categorical distribution smoothing adds one pseudo-count to each of the k categories, leading to the smoothed estimate $(d+1)/(n+k)$. The *m-estimate* generalises this further by making both the total number of pseudo-counts m and the way they are distributed over the categories into parameters. The estimate for the i-th category is defined as $(d+p_i m)/(n+m)$, where p_i is a distribution over the categories (i.e., $\sum_{i=1}^{k} p_i = 1$). Notice that smoothed relative frequency estimates – and hence products of such estimates – can never attain the extreme values $\hat{\theta} = 0$ or $\hat{\theta} = 1$.

Example 9.5 (Training a naive Bayes model). We now show how the parameter vectors in the previous example might have been obtained. Consider the following e-mails consisting of five words a, b, c, d, e:

e_1: *b d e b b d e*	e_5: *a b a b a b a e d*
e_2: *b c e b b d d e c c*	e_6: *a c a c a c a e d*
e_3: *a d a d e a e e*	e_7: *e a e d a e a*
e_4: *b a d b e d a b*	e_8: *d e d e d*

We are told that the e-mails on the left are spam and those on the right are ham, and so we use them as a small training set to train our Bayesian classifier. First,

E-mail	$\#a$	$\#b$	$\#c$	Class
e_1	0	3	0	+
e_2	0	3	3	+
e_3	3	0	0	+
e_4	2	3	0	+
e_5	4	3	0	–
e_6	4	0	3	–
e_7	3	0	0	–
e_8	0	0	0	–

E-mail	$a?$	$b?$	$c?$	Class
e_1	0	1	0	+
e_2	0	1	1	+
e_3	1	0	0	+
e_4	1	1	0	+
e_5	1	1	0	–
e_6	1	0	1	–
e_7	1	0	0	–
e_8	0	0	0	–

Table 9.1. (left) A small e-mail data set described by count vectors. **(right)** The same data set described by bit vectors.

we decide that d and e are so-called *stop words* that are too common to convey class information. The remaining words, a, b and c, constitute our vocabulary.

For the multinomial model, we represent each e-mail as a count vector, as in Table 9.1 (left). In order to estimate the parameters of the multinomial, we sum up the count vectors for each class, which gives $(5,9,3)$ for spam and $(11,3,3)$ for ham. To smooth these probability estimates we add one pseudo-count for each vocabulary word, which brings the total number of occurrences of vocabulary words to 20 for each class. The estimated parameter vectors are thus $\hat{\boldsymbol{\theta}}^{\oplus} = (6/20, 10/20, 4/20) = (0.3, 0.5, 0.2)$ for spam and $\hat{\boldsymbol{\theta}}^{\ominus} = (12/20, 4/20, 4/20) = (0.6, 0.2, 0.2)$ for ham.

In the multivariate Bernoulli model e-mails are represented by bit vectors, as in Table 9.1 (right). Adding the bit vectors for each class results in $(2,3,1)$ for spam and $(3,1,1)$ for ham. Each count is to be divided by the number of documents in a class, in order to get an estimate of the probability of a document containing a particular vocabulary word. Probability smoothing now means adding two pseudo-documents, one containing each word and one containing none of them. This results in the estimated parameter vectors $\hat{\boldsymbol{\theta}}^{\oplus} = (3/6, 4/6, 2/6) = (0.5, 0.67, 0.33)$ for spam and $\hat{\boldsymbol{\theta}}^{\ominus} = (4/6, 2/6, 2/6) = (0.67, 0.33, 0.33)$ for ham.

Many other variations of the naive Bayes classifier exist. In fact, what is normally understood as 'the' naive Bayes classifier employs neither a multinomial nor a multivariate Bernoulli model, but rather a multivariate categorical model. This means that features are categorical, and the probability of the i-th feature taking on its l-th value for class c examples is given by $\theta_{il}^{(c)}$, under the constraint that $\sum_{l=1}^{k_i} \theta_{il}^{(c)} = 1$, where k_i

is the number of values of the i-th feature. These parameters can be estimated by smoothed relative frequencies in the training set, as in the multivariate Bernoulli case. We again have that the joint probability of the feature vector is the product of the individual feature probabilities, and hence $P(F_i, F_j|C) = P(F_i|C)P(F_j|C)$ for all pairs of features and for all classes.

Notice, by the way, that conditional independence is quite different from unconditional independence: neither implies the other. To see that conditional independence does not imply unconditional independence, imagine two words that are very likely to occur in spam, but they are independent (i.e., the probability of both of them occurring in a spam e-mail is the product of the marginal probabilities). Imagine further that they are very unlikely – but also independent – in ham. Suppose I tell you an unclassified e-mail contains one of the words: you would probably guess that it is a spam e-mail, from which you would further guess that it also contains the other word – demonstrating that the words are not unconditionally independent. To see that unconditional independence does not imply conditional independence, consider two different independent words, and let an e-mail be spam if it contains at least one of the words and ham otherwise, then among spam e-mails the two words are dependent (since if I know that a spam e-mail doesn't contain one of the words, then it must contain the other).

Another extension of the naive Bayes model is required when some of the features are real-valued. One option is to discretise the real-valued features in a pre-processing stage: this will be discussed in Chapter 10. Another option is to assume that the feature values are normally distributed within each class, as discussed in the previous section. In this context it is worth noting that the naive Bayes assumption boils down to assuming a diagonal covariance matrix within each class, so that each feature can be treated independently. A third option that is also used in practice is to model the class-conditional likelihood of each feature by a non-parametric density estimator. These three options are illustrated in Figure 9.5.

In summary, the naive Bayes model is a popular model for dealing with textual, categorical and mixed categorical/real-valued data. Its main shortcoming as a probabilistic model – poorly calibrated probability estimates – are outweighed by generally good ranking performance. Another apparent paradox with naive Bayes is that it isn't particularly Bayesian at all! For one thing, we have seen that the poor probability estimates necessitate the use of reweighted likelihoods, which avoids using Bayes' rule altogether. Secondly, in training a naive Bayes model we use maximum-likelihood parameter estimation, whereas a fully fledged Bayesian approach would not commit to a particular parameter value, but rather employ a full posterior distribution. Personally, I think the essence of naive Bayes is the decomposition of joint likelihoods into marginal likelihoods. This decomposition is evocatively visualised by the Scottish tartan pattern in Figure 1.3 on p.31, which is why I like to call naive Bayes the 'Scottish classifier'.

https://doi.org/10.1017/CBO9780511973000.011 Published online by Cambridge University Press

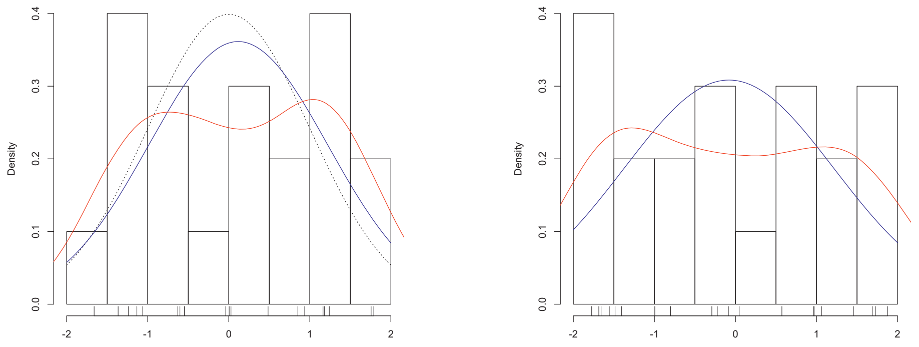

Figure 9.5. **(left)** Examples of three density estimators on 20 points sampled from a normal distribution with zero mean and unit variance (dotted line)). A histogram is a simple non-parametric method which employs a fixed number of equal-width intervals. A kernel density estimator (in red) applies interpolation to obtain a smooth density function. The solid bell curve (in blue) is obtained by estimating the sample mean and variance, assuming the true distribution is normal. **(right)** Here, the 20 points are sampled uniformly from $[-2,2]$, and the non-parametric methods generally do better.

9.3 Discriminative learning by optimising conditional likelihood

In the introduction to this chapter we distinguished between generative and discriminative probabilistic models. Naive Bayes models are generative: after training they can be used to generate data. In this section we look at one of the most commonly used discriminative models: *logistic regression*.[5] The easiest way to understand logistic regression is as a linear classifier whose probability estimates have been logistically calibrated using the method described in Section 7.4, but with one crucial difference: calibration is an integral part of the training algorithm, rather than a post-processing step. While in generative models the decision boundary is a by-product of modelling the distributions of each class, logistic regression models the decision boundary directly. For example, if the classes are overlapping then logistic regression will tend to locate the decision boundary in an area where classes are maximally overlapping, regardless of the 'shapes' of the samples of each class. This results in decision boundaries that are noticeably different from those learned by generative classifiers (Figure 9.6).

Equation 7.13 on p.222 expresses the likelihood ratio as $\exp\big(\gamma(d(\mathbf{x})-d_0)\big)$ with $d(\mathbf{x}) = \mathbf{w}\cdot\mathbf{x}-t$. Since we are learning the parameters all at once in discriminative learning, we can absorb γ and d_0 into $\mathbf{w}$ and t. So the logistic regression model is

[5]Notice that the term 'regression' is a bit of a misnomer here, since, even though a probability estimator approximates an unknown function, the training labels are classes rather than true function values.

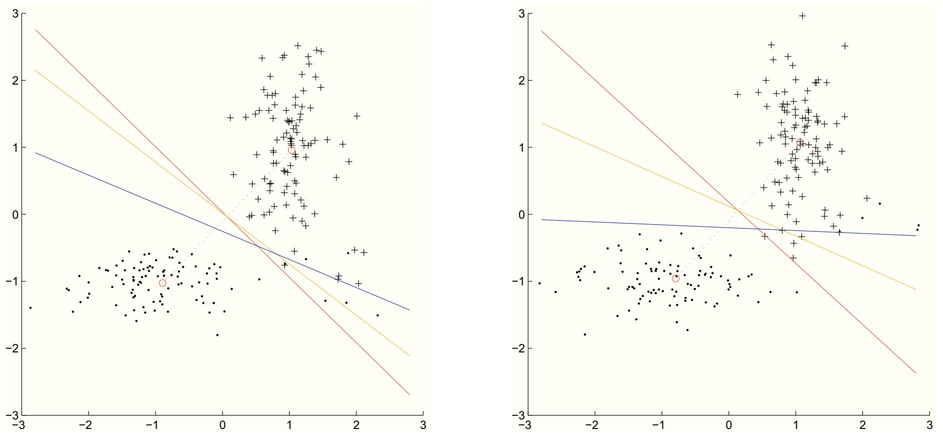

Figure 9.6. (left) On this data set, logistic regression (in blue) outperforms the basic linear classifier (in red) and the least squares classifier (in orange) because the latter two are more sensitive to the shape of the classes, while logistic regression concentrates on where the classes overlap. **(right)** On this slightly different set of points, logistic regression is outperformed by the other two methods because it concentrates too much on tracking the transition from mostly positive to mostly negative.

simply given by

$$\hat{p}(\mathbf{x}) = \frac{\exp(\mathbf{w}\cdot\mathbf{x} - t)}{\exp(\mathbf{w}\cdot\mathbf{x} - t) + 1} = \frac{1}{1 + \exp(-(\mathbf{w}\cdot\mathbf{x} - t))}$$

Assuming the class labels are $y = 1$ for positives and $y = 0$ for negatives, this defines a Bernoulli distribution for each training example:

$$P(y_i|\mathbf{x}_i) = \hat{p}(\mathbf{x}_i)^{y_i}(1 - \hat{p}(\mathbf{x}_i))^{(1-y_i)}$$

It is important to note that the parameters of these Bernoulli distributions are linked through $\mathbf{w}$ and t, and consequently there is one parameter for every feature dimension, rather than for every training instance.

The likelihood function is

$$\mathrm{CL}(\mathbf{w}, t) = \prod_i P(y_i|\mathbf{x}_i) = \prod_i \hat{p}(\mathbf{x}_i)^{y_i}(1 - \hat{p}(\mathbf{x}_i))^{(1-y_i)}$$

This is called *conditional likelihood* to stress that it gives us the *conditional* probability $P(y_i|\mathbf{x}_i)$ rather than $P(\mathbf{x}_i)$ as in a generative model. Notice that our use of the product requires the assumption that the y-values are independent given $\mathbf{x}$; but this is an entirely reasonable assumption and not nearly as strong as the naive Bayes assumption of $\mathbf{x}$ being independent within each class. As usual, the logarithm of the likelihood function is easier to work with:

$$\mathrm{LCL}(\mathbf{w},t) = \sum_i y_i \ln \hat{p}(\mathbf{x}_i) + (1-y_i)\ln(1-\hat{p}(\mathbf{x}_i)) = \sum_{\mathbf{x}^{\oplus}\in Tr^{\oplus}} \ln \hat{p}(\mathbf{x}^{\oplus}) + \sum_{\mathbf{x}^{\ominus}\in Tr^{\ominus}} \ln(1-\hat{p}(\mathbf{x}^{\ominus}))$$

We want to maximise the log-conditional likelihood with respect to these parameters, which means that all partial derivatives must be zero:

$$\nabla_{\mathbf{w}}\mathrm{LCL}(\mathbf{w},t) = \mathbf{0}$$
$$\frac{\partial}{\partial t}\mathrm{LCL}(\mathbf{w},t) = 0$$

Although these equations do not yield an analytic solution, they can be used to obtain further insight into the nature of logistic regression. Concentrating on t, we first need to do some algebraic groundwork.

$$\begin{aligned}
\ln \hat{p}(\mathbf{x}) &= \ln \frac{\exp(\mathbf{w}\cdot\mathbf{x}-t)}{\exp(\mathbf{w}\cdot\mathbf{x}-t)+1}\\
&= \mathbf{w}\cdot\mathbf{x}-t-\ln(\exp(\mathbf{w}\cdot\mathbf{x}-t)+1)\\
\frac{\partial}{\partial t}\ln \hat{p}(\mathbf{x}) &= -1-\frac{\partial}{\partial t}\ln(\exp(\mathbf{w}\cdot\mathbf{x}-t)+1)\\
&= -1-\frac{1}{\exp(\mathbf{w}\cdot\mathbf{x}-t)+1}\exp(\mathbf{w}\cdot\mathbf{x}-t)\cdot(-1)\\
&= \hat{p}(\mathbf{x})-1
\end{aligned}$$

Similarly for the negatives:

$$\begin{aligned}
\ln(1-\hat{p}(\mathbf{x})) &= \ln \frac{1}{\exp(\mathbf{w}\cdot\mathbf{x}-t)+1}\\
&= -\ln(\exp(\mathbf{w}\cdot\mathbf{x}-t)+1)\\
\frac{\partial}{\partial t}\ln(1-\hat{p}(\mathbf{x})) &= \frac{\partial}{\partial t}-\ln(\exp(\mathbf{w}\cdot\mathbf{x}-t)+1)\\
&= \frac{-1}{\exp(\mathbf{w}\cdot\mathbf{x}-t)+1}\exp(\mathbf{w}\cdot\mathbf{x}-t)\cdot(-1)\\
&= \hat{p}(\mathbf{x})
\end{aligned}$$

It follows that the partial derivative of LCL with respect to t has a simple form:

$$\begin{aligned}
\frac{\partial}{\partial t}\mathrm{LCL}(\mathbf{w},t) &= \sum_{\mathbf{x}^{\oplus}\in Tr^{\oplus}}(\hat{p}(\mathbf{x})-1) + \sum_{\mathbf{x}^{\ominus}\in Tr^{\ominus}}\hat{p}(\mathbf{x}^{\ominus})\\
&= \sum_{\mathbf{x}_i\in Tr}(\hat{p}(\mathbf{x}_i)-y_i)
\end{aligned}$$

For the optimal solution this partial derivative is zero. What this means is that, on average, the predicted probability should be equal to the proportion of positives *pos*. This is a satisfying result, as it is clearly a desirable global property of a calibrated classifier.

Notice that grouping models such as probability estimating trees have this property by construction, as they set the predicted probability equal to the empirical probability in a segment.

A very similar derivation leads to the partial derivative of the log-conditional likelihood with respect to the j-th weight w_j. The point to note here is that, whereas $\frac{\partial}{\partial t}(\mathbf{w}\cdot\mathbf{x}-t) = -1$, we have $\frac{\partial}{\partial w_j}(\mathbf{w}\cdot\mathbf{x}-t) = \frac{\partial}{\partial w_j}\left(\sum_j w_j x_j - t\right) = x_j$, the instance's j-th feature value. This then leads to

$$\frac{\partial}{\partial w_j}\text{LCL}(\mathbf{w},t) = \sum_{\mathbf{x}_i\in Tr}(y_i - \hat{p}(\mathbf{x}_i))x_{ij} \tag{9.5}$$

Setting this partial derivative to zero expresses another, feature-wise calibration property. For example, if the j-th feature is a sparse Boolean feature that is mostly zero, then this calibration property only involves the instances $\mathbf{x}_i$ for which $x_{ij} = 1$: on average, those instances should have their predicted probability equal the proportion of positives among them.

Example 9.6 (Univariate logistic regression). Consider the data in Figure 9.7 with 20 points in each class. Although both classes were generated from normal distributions, class overlap in this particular sample is less than what could be expected on the basis of the class means. Logistic regression is able to take advantage of this and gives a much steeper sigmoid than the basic linear classifier with logistic calibration (explained in Example 7.7 on p.222), which is entirely formulated in terms of class means and variance. Also shown are the probability estimates obtained from the convex hull of the ROC curve (see Figure 7.13 on p.224); this calibration procedure is non-parametric and hence better able to detect the limited class overlap.

In terms of statistics, logistic regression has better mean squared error (0.040) than the logistically calibrated classifier (0.057). Isotonic calibration leads to the lowest error (0.021), but note that no probability smoothing has been applied to mitigate the risk of overfitting. The sum of predicted probabilities is 18.7 for the logistically calibrated classifier and 20 for the other two – i.e., equal to the number of examples, which is a necessary condition for full calibration. Finally, $\sum_{\mathbf{x}_i\in Tr}(y_i - \hat{p}(\mathbf{x}_i))x_i$ is 2.6 for the logistically calibrated classifier, 4.7 for the ROC-calibrated classifier, and 0 for logistic regression as expected from Equation 9.5.

In order to train a logistic regression model we need to find

$$\mathbf{w}^*, t^* = \arg\max_{\mathbf{w},t}\text{CL}(\mathbf{w},t) = \arg\max_{\mathbf{w},t}\text{LCL}(\mathbf{w},t)$$

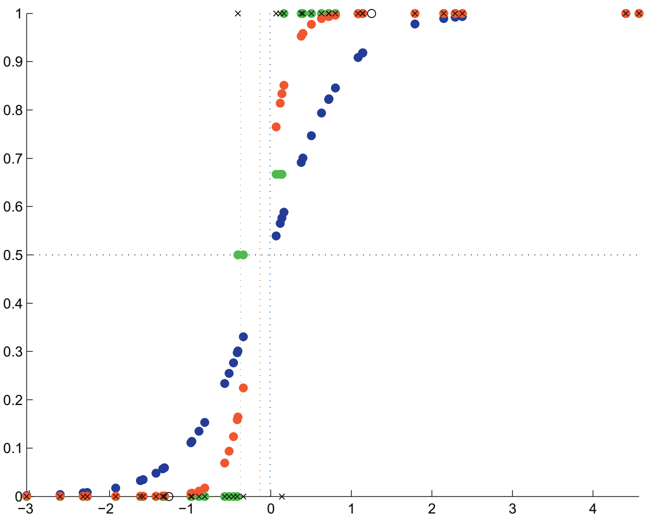

Figure 9.7. Logistic regression (in red) compared with probability estimates obtained by logistic calibration (in blue) and isotonic calibration (in green); the latter two are applied to the basic linear classifier (estimated class means are indicated by circles). The corresponding three decision boundaries are shown as vertical dotted lines.

This can be shown to be a convex optimisation problem, which means that there is only one maximum. A range of optimisation techniques can be applied. One simple approach is inspired by the perceptron algorithm and iterates over examples, using the following update rule:

$$\mathbf{w} = \mathbf{w} + \eta(y_i - \hat{p}_i)\mathbf{x}_i$$

where η is the learning rate. Notice the relationship with the partial derivative in Equation 9.5. Essentially, we are using single examples to approximate the direction of steepest ascent.

9.4 Probabilistic models with hidden variables

Suppose you are dealing with a four-class classification problem with classes A, B, C and D. If you have a sufficiently large and representative training sample of size n, you can use the relative frequencies in the sample $n_A, \dots, n_D$ to estimate the class prior $\hat{p}_A = n_A/n, \dots, \hat{p}_D = n_D/n$, as we have done many times before.[6] Conversely, if you know the prior and want to know the most likely class distribution in a random

[6]Of course, if you're not sure whether the sample is large enough it is better to smooth these relative frequency estimates by, e.g., the ☞*Laplace correction* (Section 2.3).

sample of n instances, you would use the prior to calculate expected values $\mathbb{E}[n_A] = p_A \cdot n, \ldots, \mathbb{E}[n_D] = p_D \cdot n$. So, complete knowledge of one allows us to estimate or infer the other. However, sometimes we have a bit of knowledge about both. For example, we may know that $p_A = 1/2$ and that C is twice as likely as B, without knowing the complete prior. And we may know that the sample we saw last week was evenly split between $A \cup B$ and $C \cup D$, and that C and D were equally large, but we can't remember the size of A and B separately. What should we do?

Formalising what we know about the prior, we have $p_A = 1/2$; $p_B = \beta$, as yet unknown; $p_C = 2\beta$, since it is twice p_B; and $p_D = 1/2 - 3\beta$, since the four cases need to add up to 1. Furthermore: $n_A + n_B = a + b = s$, $n_C = c$ and $n_D = d$, with s, c and d known. We want to infer a, b and β: however, it seems we are stuck in a chicken-and-egg problem. If we knew β we would have full knowledge about the prior and we could use that to infer expected values for a and b:

$$\frac{\mathbb{E}[a]}{\mathbb{E}[b]} = \frac{1/2}{\beta} \qquad\qquad \mathbb{E}[a] + \mathbb{E}[b] = s$$

from which we could derive

$$\mathbb{E}[a] = \frac{1}{1+2\beta}s \qquad\qquad \mathbb{E}[b] = \frac{2\beta}{1+2\beta}s \tag{9.6}$$

So, for example, if $s = 20$ and $\beta = 1/10$, then $\mathbb{E}[a] = 16\frac{2}{3}$ and $\mathbb{E}[b] = 3\frac{1}{3}$.

Conversely, if we knew a and b, then we could estimate β by maximum-likelihood estimation, using a multinomial distribution for a, b, c and d:

$$\begin{aligned} P(a,b,c,d|\beta) &= K(1/2)^a \beta^b (2\beta)^c (1/2 - 3\beta)^d \\ \ln P(a,b,c,d|\beta) &= \ln K + a\ln(1/2) + b\ln\beta + c\ln(2\beta) + d\ln(1/2 - 3\beta) \end{aligned}$$

Here, K is a combinatorial constant that doesn't affect the value of β which maximises the likelihood. Taking the partial derivative with respect to β gives

$$\frac{\partial}{\partial\beta}\ln P(a,b,c,d|\beta) = \frac{b}{\beta} + \frac{2c}{2\beta} - \frac{3d}{1/2 - 3\beta}$$

Setting to 0 and solving for β finally gives

$$\hat{\beta} = \frac{b+c}{6(b+c+d)} \tag{9.7}$$

So, for example, if $b = 5$ and $c = d = 10$, then $\hat{\beta} = 1/10$.

The way out of this chicken-and-egg problem is to iterate the following two steps: (*i*) calculate an expected value of the missing frequencies a and b from an assumed or previously estimated value of the parameter β; and (*ii*) calculate a maximum-likelihood estimate of the parameter β from assumed or expected values of the missing frequencies a and b. These two steps are iterated until a stationary configuration is reached.

So, if we start with $a = 15$, $b = 5$ and $c = d = 10$, then we have just seen that $\hat{\beta} = 1/10$. Plugging this value of β into Equation 9.6 gives us $\mathbb{E}[a] = 16\frac{2}{3}$ and $\mathbb{E}[b] = 3\frac{1}{3}$. Plugging these values back into Equation 9.7 yields $\hat{\beta} = 2/21$, which in turn gives $\mathbb{E}[a] = 16.8$ and $\mathbb{E}[b] = 3.2$, and so on. A stationary configuration with $\beta = 0.0948$, $a = 16.813$ and $b = 3.187$ is reached in fewer than 10 iterations. In this simple case this is a global optimum that is reached regardless of the starting point, essentially because the relationship between b and β is monotonic ($\mathbb{E}[b]$ increases with β according to Equation 9.6 and $\hat{\beta}$ increases with b according to Equation 9.7). However, this is not normally the case: we will return to this point later.

Expectation-Maximisation

The problem that we have just discussed is an example of a problem with missing data, where the full data Y separates into observed variables X and *hidden variables* Z (also called *latent variables*). In the example, the observed variables are c, d and s, and the hidden variables are a and b. We also have model parameter(s) θ, which is β in the example.[7] Denote the estimate of θ in the t-th iteration as θ^t. We have two relevant quantities:

- the expectation $\mathbb{E}\left[Z|X,\theta^t\right]$ of the hidden variables given the observed variables and the current estimate of the parameters (so in Equation 9.6 the expectations of a and b depend on s and β);
- the likelihood $P(Y|\theta)$, which is used to find the maximising value of θ.

In the likelihood function we need values for $Y = X \cup Z$. We obviously use the observed values for X, but we need to use previously calculated expectations for Z. This means that we really want to maximise $P(X \cup \mathbb{E}\left[Z|X,\theta^t\right]|\theta)$, or equivalently, the logarithm of that function. We now make the assumption that the logarithm of the likelihood function is linear in Y: notice that this assumption is valid in the example above. For any linear function f, $f(\mathbb{E}[Z]) = \mathbb{E}\left[f(Z)\right]$ and thus we can bring the expectation outside in our objective function:

$$\ln P(X \cup \mathbb{E}\left[Z|X,\theta^t\right]|\theta) = \mathbb{E}\left[\ln P(X \cup Z|\theta)|X,\theta^t\right] = \mathbb{E}\left[\ln P(Y|\theta)|X,\theta^t\right] \qquad (9.8)$$

This last expression is usually denoted as $Q(\theta|\theta^t)$, as it essentially tells us how to calculate the next value of θ from the current one:

$$\theta^{t+1} = \arg\max_{\theta} Q(\theta|\theta^t) = \arg\max_{\theta} \mathbb{E}\left[\ln P(Y|\theta)|X,\theta^t\right] \qquad (9.9)$$

[7]Model parameters are also 'hidden' in a sense, but they are different from hidden variables in that you would never expect to observe the value of a parameter (e.g., a class mean), whereas a hidden variable could be observed in principle but happens to be unobserved in the case at hand.

This, then, is the general form of the celebrated *Expectation-Maximisation* (*EM*) algorithm, which is a powerful approach to probabilistic modelling with hidden variables or missing data. Similar to the example above, we iterate over assigning an expected value to the hidden variables given our current estimates of the parameters, and re-estimating the parameters from these updated expectations, until a stationary configuration is reached. We can start the iteration by initialising either the parameters or the hidden variables in some way. The algorithm bears a striking resemblance to the ☞*K-means* algorithm (Algorithm 8.1 on p.248), which also iterates over assigning data points to current cluster means, and re-estimating the cluster means from the new assignments. This resemblance is not accidental, as we shall see in a moment. Like the K-means algorithm, EM can be proved to always converge to a stationary configuration for a wide class of probabilistic models. However, EM can get trapped in a local optimum that is dependent on the initial configuration.

Gaussian mixture models

A common application of Expectation-Maximisation is to estimate the parameters of a *Gaussian mixture model* from data. In such a model the data points are generated by K normal distributions, each with their own mean $\boldsymbol{\mu}_j$ and covariance matrix $\boldsymbol{\Sigma}_j$, and the proportion of points coming from each Gaussian is governed by a prior $\boldsymbol{\tau} = (\tau_1, \ldots, \tau_K)$. If each data point in a sample were labelled with the index of the Gaussian it came from this would be a straightforward classification problem, which could be solved easily by estimating each Gaussian's $\boldsymbol{\mu}_j$ and $\boldsymbol{\Sigma}_j$ separately from the data points belonging to class j. However, we are now considering the much harder predictive clustering problem in which the class labels are hidden and need to be reconstructed from the observed feature values.

A convenient way to model this is to have for each data point $\mathbf{x}_i$ a Boolean vector $\mathbf{z}_i = (z_{i1}, \ldots, z_{iK})$ such that exactly one bit z_{ij} is set to 1 and the rest set to 0, signalling that the i-th data point comes from the j-th Gaussian. Using this notation we can adapt the expression for the ☞*multivariate normal distribution* (Equation 9.2 on p.267) to obtain a general expression for a Gaussian mixture model:

$$P(\mathbf{x}_i, \mathbf{z}_i | \theta) = \sum_{j=1}^{K} z_{ij} \tau_j \frac{1}{(2\pi)^{d/2} \sqrt{|\boldsymbol{\Sigma}_j|}} \exp\left(-\frac{1}{2}(\mathbf{x}_i - \boldsymbol{\mu}_j)^{\mathrm{T}} \boldsymbol{\Sigma}_j^{-1} (\mathbf{x}_i - \boldsymbol{\mu}_j)\right) \tag{9.10}$$

Here, θ collects all the parameters τ, $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_K$ and $\boldsymbol{\Sigma}_1, \ldots, \boldsymbol{\Sigma}_K$. The interpretation as a generative model is as follows: we first randomly select a Gaussian using the prior $\boldsymbol{\tau}$, and then we invoke the corresponding Gaussian using the indicator variables z_{ij}.

In order to apply Expectation-Maximisation we form the Q function:

$$
\begin{aligned}
Q(\theta|\theta^t) &= \mathbb{E}\left[\ln P(\mathbf{X}\cup\mathbf{Z}|\theta)|\mathbf{X},\theta^t\right] \\
&= \mathbb{E}\left[\ln\prod_{i=1}^{n} P(\mathbf{x}_i\cup\mathbf{z}_i|\theta)\middle|\mathbf{X},\theta^t\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{n}\ln P(\mathbf{x}_i\cup\mathbf{z}_i|\theta)\middle|\mathbf{X},\theta^t\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{n}\ln\sum_{j=1}^{K} z_{ij}\tau_j\frac{1}{(2\pi)^{d/2}\sqrt{|\boldsymbol{\Sigma}_j|}}\exp\left(-\frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu}_j)^{\mathrm{T}}\boldsymbol{\Sigma}_j^{-1}(\mathbf{x}_i-\boldsymbol{\mu}_j)\right)\middle|\mathbf{X},\theta^t\right] \\
&= \mathbb{E}\left[\sum_{i=1}^{n}\sum_{j=1}^{K} z_{ij}\ln\left(\tau_j\frac{1}{(2\pi)^{d/2}\sqrt{|\boldsymbol{\Sigma}_j|}}\exp\left(-\frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu}_j)^{\mathrm{T}}\boldsymbol{\Sigma}_j^{-1}(\mathbf{x}_i-\boldsymbol{\mu}_j)\right)\right)\middle|\mathbf{X},\theta^t\right] \qquad (*) \\
&= \mathbb{E}\left[\sum_{i=1}^{n}\sum_{j=1}^{K} z_{ij}\left(\ln\tau_j-\frac{d}{2}\ln(2\pi)-\frac{1}{2}\ln|\boldsymbol{\Sigma}_j|-\frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu}_j)^{\mathrm{T}}\boldsymbol{\Sigma}_j^{-1}(\mathbf{x}_i-\boldsymbol{\mu}_j)\right)\middle|\mathbf{X},\theta^t\right] \\
&= \sum_{i=1}^{n}\sum_{j=1}^{K}\mathbb{E}\left[z_{ij}\middle|\mathbf{X},\theta^t\right]\left(\ln\tau_j-\frac{d}{2}\ln(2\pi)-\frac{1}{2}\ln|\boldsymbol{\Sigma}_j|-\frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu}_j)^{\mathrm{T}}\boldsymbol{\Sigma}_j^{-1}(\mathbf{x}_i-\boldsymbol{\mu}_j)\right)
\end{aligned}
\tag{9.11}
$$

The step marked (*) is possible because for a given i only one z_{ij} is switched on, hence we can bring the indicator variables outside the logarithm. The last line shows the Q function in the desired form, involving on the one hand expectations over the hidden variables conditioned on the observable data $\mathbf{X}$ and the previously estimated parameters θ^t, and on the other hand expressions in θ that allow us to find θ^{t+1} by maximisation.

The Expectation step of the EM algorithm is thus the calculation of the expected values of the indicator variables $\mathbb{E}\left[z_{ij}\middle|\mathbf{X},\theta^t\right]$. Notice that expectations of Boolean variables take values on the entire interval $[0,1]$, under the constraint that $\sum_{j=1}^{K} z_{ij}=1$ for all i. In effect, the hard cluster assignment of K-means is changed into a soft assignment – one of the ways in which Gaussian mixture models generalise K-means. Now, suppose that $K=2$ and we expect both clusters to be of equal size and with equal covariances. If a given data point $\mathbf{x}_i$ is equidistant from the two cluster means (or rather, our current estimates of these), then clearly $\mathbb{E}\left[z_{i1}|\mathbf{X},\theta^t\right]=\mathbb{E}\left[z_{i2}|\mathbf{X},\theta^t\right]=1/2$. In the general case these expectations are apportioned proportionally to the probability mass assigned to the point by each Gaussian:

$$
\mathbb{E}\left[z_{ij}\middle|\mathbf{X},\theta^t\right]=\frac{\tau_j^t f(\mathbf{x}_i|\boldsymbol{\mu}_j^t,\boldsymbol{\Sigma}_j^t)}{\sum_{k=1}^{K}\tau_k^t f(\mathbf{x}_i|\boldsymbol{\mu}_k^t,\boldsymbol{\Sigma}_k^t)}
\tag{9.12}
$$

where $f(\mathbf{x}|\boldsymbol{\mu},\boldsymbol{\Sigma})$ stands for the multivariate Gaussian density function.

For the Maximisation step we optimise the parameters in Equation 9.11. Notice there is no interaction between the terms containing τ_j and the terms containing the other parameters, and so the prior distribution $\boldsymbol{\tau}$ can be optimised separately:

$$\begin{aligned}\boldsymbol{\tau}^{t+1} &= \arg\max_{\boldsymbol{\tau}} \sum_{i=1}^{n}\sum_{j=1}^{K} \mathbb{E}\left[z_{ij}\middle|\mathbf{X},\theta^t\right]\ln\tau_j \\ &= \arg\max_{\boldsymbol{\tau}} \sum_{j=1}^{K} E_j \ln\tau_j \qquad \text{under the constraint } \sum_{j=1}^{K}\tau_j = 1\end{aligned}$$

where I have written E_j for $\sum_{i=1}^{n}\mathbb{E}\left[z_{ij}\middle|\mathbf{X},\theta^t\right]$, which is the total (partial) membership of the j-th cluster – notice that $\sum_{j=1}^{K}E_j = n$. For simplicity we assume $K = 2$, so that $\tau_2 = 1-\tau_1$: then

$$\tau_1^{t+1} = \arg\max_{\tau_1} E_1\ln\tau_1 + E_2\ln(1-\tau_1)$$

Setting the derivative with respect to τ_1 to zero and solving for τ_1, it can be easily verified that $\tau_1^{t+1} = E_1/(E_1+E_2) = E_1/n$ and thus $\tau_2^{t+1} = E_2/n$. In the general case of K clusters we have analogously

$$\tau_j^{t+1} = \frac{E_j}{\sum_{k=1}^{K}E_k} = \frac{1}{n}\sum_{i=1}^{n}\mathbb{E}\left[z_{ij}\middle|\mathbf{X},\theta^t\right] \tag{9.13}$$

The means and covariance matrices can be optimised for each cluster separately:

$$\begin{aligned}\boldsymbol{\mu}_j^{t+1},\boldsymbol{\Sigma}_j^{t+1} &= \arg\max_{\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j}\sum_{i=1}^{n}\mathbb{E}\left[z_{ij}\middle|\mathbf{X},\theta^t\right]\left(-\frac{1}{2}\ln|\boldsymbol{\Sigma}_j| - \frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu}_j)^{\mathrm{T}}\boldsymbol{\Sigma}_j^{-1}(\mathbf{x}_i-\boldsymbol{\mu}_j)\right) \\ &= \arg\min_{\boldsymbol{\mu}_j,\boldsymbol{\Sigma}_j}\sum_{i=1}^{n}\mathbb{E}\left[z_{ij}\middle|\mathbf{X},\theta^t\right]\left(\frac{1}{2}\ln|\boldsymbol{\Sigma}_j| + \frac{1}{2}(\mathbf{x}_i-\boldsymbol{\mu}_j)^{\mathrm{T}}\boldsymbol{\Sigma}_j^{-1}(\mathbf{x}_i-\boldsymbol{\mu}_j)\right)\end{aligned}$$

Notice that the term between brackets is a squared-distance term with the expectations functioning as instance weights on each instance. This describes a generalised version of the problem of finding the point that ☞*minimises the sum of squared Euclidean distances* to a set of points (Theorem 8.1 on p.238). While that problem is solved by the arithmetic mean, here we simply take the *weighted* average over all the points:

$$\boldsymbol{\mu}_j^{t+1} = \frac{1}{E_j}\sum_{i=1}^{n}\mathbb{E}\left[z_{ij}\middle|\mathbf{X},\theta^t\right]\mathbf{x}_i = \frac{\sum_{i=1}^{n}\mathbb{E}\left[z_{ij}\middle|\mathbf{X},\theta^t\right]\mathbf{x}_i}{\sum_{i=1}^{n}\mathbb{E}\left[z_{ij}\middle|\mathbf{X},\theta^t\right]} \tag{9.14}$$

Similarly, the covariance matrix is computed as a weighted average of covariance matrices obtained from each data point, taking into account the newly estimated mean:

$$\begin{aligned}\boldsymbol{\Sigma}_j^{t+1} &= \frac{1}{E_j}\sum_{i=1}^{n}\mathbb{E}\left[z_{ij}\middle|\mathbf{X},\theta^t\right](\mathbf{x}_i-\boldsymbol{\mu}_j^{t+1})(\mathbf{x}_i-\boldsymbol{\mu}_j^{t+1})^{\mathrm{T}} \\ &= \frac{\sum_{i=1}^{n}\mathbb{E}\left[z_{ij}\middle|\mathbf{X},\theta^t\right](\mathbf{x}_i-\boldsymbol{\mu}_j^{t+1})(\mathbf{x}_i-\boldsymbol{\mu}_j^{t+1})^{\mathrm{T}}}{\sum_{i=1}^{n}\mathbb{E}\left[z_{ij}\middle|\mathbf{X},\theta^t\right]}\end{aligned} \tag{9.15}$$

Equations 9.12–9.15, then, constitute the EM solution to learning a Gaussian mixture model from an unlabelled sample. I have presented it here in its most general

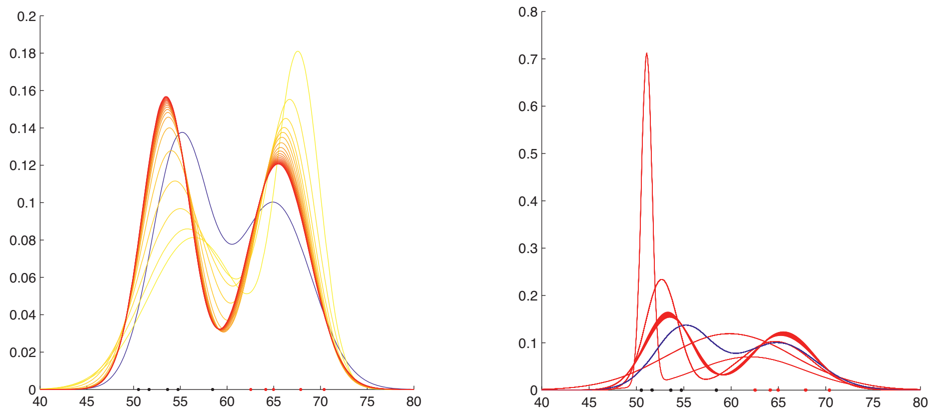

Figure 9.8. **(left)** The blue line shows the true Gaussian mixture model from which the 10 points on the x-axis were sampled; the colour of the points indicates whether they came from the left or the right Gaussian. The other lines show convergence of Expectation-Maximisation to a stationary configuration from a random initialisation. **(right)** This plot shows four stationary configurations for the same set of points. The EM algorithm was run for 20 iterations; the thickness of one of the lines demonstrates that this configuration takes longer to converge.

form, explicitly modelling unequal cluster sizes and covariance matrices. The latter is important as it allows for clusters of different shapes, unlike the K-means algorithm which assumes that all clusters have the same spherical shape. Consequently, the boundaries between clusters will not be linear, as they are in the clusterings learned by K-means. Figure 9.8 demonstrates the convergence of EM on a simple univariate data set, as well as the existence of multiple stationary configurations.

In conclusion, Expectation-Maximisation is a versatile and powerful method to deal with missing variables in a principled way. As we have seen in detail for the Gaussian mixture model, the main ingredient is an expression for the parametric likelihood function $P(X \cup Z|\theta)$, from which the update equations can be derived by means of the Q function. A word of caution is also in order, since – except in the simplest cases – there will be more than one stationary configuration. Like with K-means, the optimisation should therefore be run multiple times with different starting configurations.

9.5 Compression-based models

We end this chapter with a brief discussion of an approach to machine learning that is both closely related to and quite distinct from the probabilistic approach. Consider the maximum a posteriori decision rule again:

$$y_{\text{MAP}} = \arg\max_y P(X = x|Y = y)P(Y = y)$$

Y	$P(\text{Viagra} = 1\|Y)$	$IC(\text{Viagra} = 1\|Y)$	$P(\text{Viagra} = 0\|Y)$	$IC(\text{Viagra} = 0\|Y)$
spam	0.40	**1.32 bits**	0.60	**0.74 bits**
ham	0.12	**3.06 bits**	0.88	**0.18 bits**

Table 9.2. Example marginal likelihoods.

Taking negative logarithms, we can turn this into an equivalent minimisation:

$$y_{\text{MAP}} = \operatorname*{arg\,min}_y -\log P(X = x|Y = y) - \log P(Y = y) \tag{9.16}$$

This follows because for any two probabilities $0 < p < p' < 1$ we have $\infty > -\log p > -\log p' > 0$. If an event has probability p of happening, the negative logarithm of p quantifies the *information content* of the message that the event has indeed happened. This makes intuitive sense, as the less expected an event is, the more information an announcement of the event contains. The unit of information depends on the base of the logarithm: it is customary to take logarithms to the base 2, in which case information is measured in bits. For example, if you toss a fair coin once and tell me it came up heads, this contains $-\log_2 1/2 = 1$ bit of information; if you roll a fair die once and let me know it came up six, the information content of your message is $-\log_2 1/6 = 2.6$ bits. Equation 9.16 tells us that the MAP decision rule chooses the least surprising or the most expected class for an instance x given particular prior distributions and likelihoods. We write $IC(X|Y) = -\log_2 P(X|Y)$ and $IC(Y) = -\log_2 P(Y)$.

Example 9.7 (Information-based classification). Table 9.2 reproduces the left table in Table 1.3 on p.29 together with the relevant information content quantities. If Y is uniformly distributed then $IC(Y = \text{spam}) = 1$ bit and $IC(Y = \text{ham}) = 1$ bit. It follows that

$$\operatorname*{arg\,min}_y \big(IC(\text{Viagra} = 1|Y = y) + IC(Y = y)\big) = \text{spam}$$

$$\operatorname*{arg\,min}_y \big(IC(\text{Viagra} = 0|Y = y) + IC(Y = y)\big) = \text{ham}$$

If ham is four times as likely as spam then $IC(Y = \text{spam}) = 2.32$ bit and $IC(Y = \text{ham}) = 0.32$ bit, and $\operatorname{arg\,min}_y \big(IC(\text{Viagra} = 1|Y = y) + IC(Y = y)\big) = \text{ham}$.

Clearly, for a uniform distribution over k outcomes, each outcome has the same information content $-\log_2 1/k = \log_2 k$. For a non-uniform distribution these information contents differ, and hence it makes sense to compute the average information content or *entropy* $\sum_{i=1}^{k} -p_i \log_2 p_i$. We have encountered entropy before as an ☞*impurity measure* in Section 5.1.

So far I have not really told you anything new, other than that there is a one-to-one relationship between probability and information content. What really kicks things off in compression-based learning is a fundamental result from information theory proved by Claude Shannon in 1948. Shannon's result says – loosely speaking – that we cannot transmit information at a rate that surpasses entropy, but we can get arbitrarily close to the optimal rate by designing clever binary codes. Some well-known codes include the Shannon–Fanon code and the Huffman code, which are worth looking up as they employ a simple tree structure to build the code from empirical probabilities. Even more efficient codes, such as arithmetic coding, combine multiple messages into a single code word.

Assuming the availability of a near-optimal code, we can now turn the tables and use information content – or 'description length' as it is more commonly called – as a proxy for probability. One simplified version of the minimum description length (MDL) principle runs as follows.

Definition 9.1 (Minimum description length principle). *Let $L(m)$ denote the length in bits of a description of model m, and let $L(D|m)$ denote the length in bits of a description of data D given model m. According to the minimum description length principle, the preferred model is the one minimising the description length of model and data given model:*

$$m_{\mathrm{MDL}} = \arg\min_{m \in M} \left(L(m) + L(D|m)\right) \tag{9.17}$$

In a predictive learning context, 'description of data given model' refers to whatever information we need, in addition to the model and the feature values of the data, to infer the target labels. If the model is 100% accurate no further information is needed, so this term essentially quantifies the extent to which the model is incorrect. For example, in a uniform two-class setting we need one bit for every data point incorrectly classified by the model. The term $L(m)$ quantifies the complexity of the model. For instance, if we are fitting a polynomial to the data we need to encode the degree of the polynomial as well as its roots, up to a certain resolution. MDL learning thus trades off accuracy and complexity of a model: the complexity term serves to avoid overfitting in a similar way to the ☞*regularisation* term in ridge regression in Section 7.1 and the ☞*slack variable* term in soft-margin SVMs in Section 7.3.

What encoding to use in order to determine the model complexity $L(m)$ is often not straightforward and to some extent subjective. This is similar to the Bayesian perspective, where we need to define a prior distribution on models. The MDL viewpoint offers a concrete way of defining model priors by means of codes.

9.6 Probabilistic models: Summary and further reading

In this chapter we covered a range of machine learning models that are all based on the idea that features and target variables can be modelled as random variables, giving the opportunity to explicitly represent and manipulate the level of certainty we have about those variables. Such models are usually predictive in that they result in a conditional distribution $P(Y|X)$ with which Y can be predicted from X. Generative models estimate the joint distribution $P(Y,X)$ – often through the likelihood $P(X|Y)$ and the prior $P(Y)$ – from which the posterior $P(Y|X)$ can be obtained, while conditional models learn the posterior $P(Y|X)$ directly without spending resources on learning $P(X)$. The 'Bayesian' approach to machine learning is characterised by concentrating on the full posterior distribution wherever this is feasible, rather than just deriving a maximising value.

☞ In Section 9.1 we saw that the normal or Gaussian distribution supports many useful geometric intuitions, essentially because the negative logarithm of the Gaussian likelihood can be interpreted as a squared distance. Straight decision boundaries result from having the same per-class covariance matrices, which means that models resulting in such linear boundaries, including linear classifiers, linear regression and K-means clustering, can be interpreted from a probabilistic viewpoint that makes their inherent assumptions explicit. Two examples of this are that the basic linear classifier is Bayes-optimal for uncorrelated, unit-variance Gaussian features; and least-squares regression is optimal for linear functions contaminated by Gaussian noise on the target variable.

☞ Section 9.2 was devoted to different versions of the naive Bayes classifier, which makes the simplifying assumption that features are independent within each class. Lewis (1998) gives an overview and history. This model is widely used in information retrieval and text classification as it is often a good ranker if not a good probability estimator. While the model that is usually understood as naive Bayes treats features as categorical or Bernoulli random variables, variants employing a multinomial model tend to better model the number of occurrences of words in a document (McCallum and Nigam, 1998). Real-valued features can be taken into account by either modelling them as normally distributed within each class, or by non-parametric density estimation – John and Langley (1995) suggest that the latter gives better empirical results. Webb, Boughton and Wang (2005) discuss ways of relaxing the strong independence assumptions made by naive Bayes. Probability smoothing by means of the m-estimate was introduced by Cestnik (1990).

☞ Perhaps paradoxically, I don't think there is anything particularly 'Bayesian' about the naive Bayes classifier. While it is a generative probabilistic model estimating the posterior $P(Y|X)$ through the joint $P(Y,X)$, in practice the posterior is very poorly calibrated owing to the unrealistic independence assumptions. The reason naive Bayes is often successful is because of the quality of $\arg\max_Y P(Y|X)$ rather than the quality of the posterior as such, as analysed by Domingos and Pazzani (1997). Furthermore, even the use of Bayes' rule in determining the maximising Y can be avoided, as it only serves to transform uncalibrated likelihoods into uncalibrated posteriors. So my recommendation is to use naive Bayes likelihoods as scores on an unknown scale whose decision threshold needs to be calibrated by means of ROC analysis, as has been discussed several times before.

☞ In Section 9.3 we looked at the widely used logistic regression model. The basic idea is to combine a linear decision boundary with logistic calibration, but to train this in a discriminative fashion by optimising conditional likelihood. So, rather than modelling the classes as clouds of points and deriving a decision boundary from those clouds, logistic regression concentrates on areas of class overlap. It is an instance of the larger class of generalised linear models (Nelder and Wedderburn, 1972). Jebara (2004) discusses the advantages of discriminative learning in comparison with generative models. Discriminative learning can also be applied to sequential data in the form of conditional random fields (Lafferty *et al.*, 2001)

☞ Section 9.4 presented the Expectation-Maximisation algorithm as a general way of learning models involving unobserved variables. This general form of EM was proposed by Dempster, Laird and Rubin (1977) based on a variety of earlier work. We have seen how it can be applied to Gaussian mixture models to obtain a more general version of K-means predictive clustering, which is also able to estimate cluster shapes and sizes. However, this increases the number of parameters of the model and thus the risk of getting stuck in a non-optimal stationary configuration. (Little and Rubin, 1987) is a standard reference for dealing with missing data.

☞ Finally, in Section 9.5 we briefly discussed some ideas related to learning as compression. The link with probabilistic modelling is that both seek to model and exploit the non-random aspects of the data. In a simplified setting, the minimum description length principle can be derived from Bayes' rule by taking the negative logarithm, and states that models minimising the description length of the model and of the data given the model should be preferred. The first term quantifies the complexity of the model, and the second term quantifies its accuracy (as only the model's errors need to be encoded explicitly). The advantage of the MDL principle is that encoding schemes are often more tangible and easier to define than prior distributions. However, not just any encoding will do: as with their probabilistic counterparts, these schemes need to be justified in the domain being modelled. Pioneering work in this area has been done by Solomonoff (1964*a*,*b*); Wallace and Boulton (1968); Rissanen (1978), among others. An excellent introduction and overview is provided by Grünwald (2007).