Machine Learning

Ben Herbst and Johan du Preez

Applied Mathematics, Electrical and Electronic Engineering
University of Stellenbosch
Matieland 7602
South Africa
herbst@sun.ac.za

Contents

NOTE. These class notes are based on Christopher M. Bishop. Pattern Recognition and Machine Learning. Springer (2006). This should be read together with the original.

CHAPTER 1

Introduction.

1.1. Curve fitting

Suppose we are given the data points, (x_i, t_i), $i = 1, \ldots, N$. We think of $\mathbf{x} = (x_1, \ldots, x_N)^T$ as N observations of a variable x, and similarly for the corresponding observations $\mathbf{t}$. The idea is to be able to predict a value for $\widehat{t}$, given a corresponding value for $\widehat{x}$. One way of approaching this problem is to view it as an interpolation problem, i.e. we fit a polynomial $t = P_N(x)$ (here N refers to the order of the polynomial) of degree $N-1$ through the N data points. The predicted value then becomes $\widehat{t} = P_N(\widehat{x})$.

This is very familiar but not very satisfactory. If N is large one has to worry about the Runge phenomenon. Then there is also the important issue of noise. What is the point of fitting a polynomial exactly through the data if the data itself contains noise, i.e. are not in the 'correct' positions? In that case one can always revert to a least squares approximation where the approximation polynomial is no longer required to pass through the data points. In particular one might want to fit a polynomial of the form

$$y(x, \mathbf{w}) = \sum_{j=0}^{M} w_j x^j, \tag{1.1}$$

where M is the degree of the polynomial (one less than the order of the polynomial). The question is how to determine the coefficients $\mathbf{w}$ in order to get a good approximation. First note that if $M \geq N-1$ then we can fit $y(x, \mathbf{w})$ exactly through the data points, and we are back at the interpolation problem. Let us therefore assume that $M < N-1$ in which case we minimize the sum of the squares of the errors between the predictions $y(x_i, \mathbf{w})$ for each data point x_i and the corresponding target

values t_i, i.e. we minimize

$$E(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\left(y(x_i,\mathbf{w}) - t_i\right)^2. \tag{1.2}$$

EXAMPLE 1. Let us generate data from the function $\sin(2\pi x)$ by first choosing random values of $x \in [0,1]$. The t values are obtained by evaluating $\sin 2\pi x$ at these random values, corrupted by Gaussian noise. The results are shown in Figure 1.1.1.

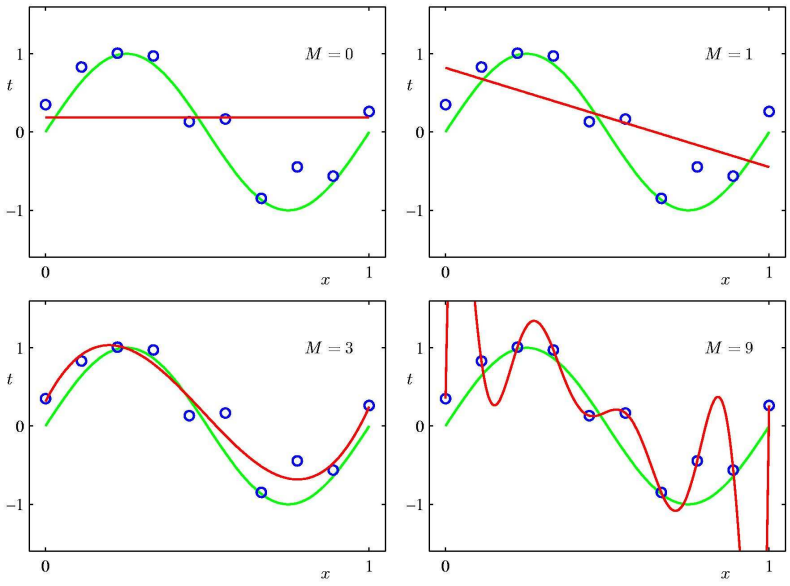


FIGURE 1.1.1. Approximating polynomial (red) of $\sin(2\pi x)$ (green) for various values of M.

The most important observation from this Figure is how bad the approximation becomes for $M = 9$. If one investigates the values of $\mathbf{w}$ for $M = 9$ one finds that the values become very large and oscillatory, see Bishop for more detail. One idea therefore is to introduce a penalty term to the error function in order to prevent large oscillatory values of $\mathbf{w}$,

$$\widetilde{E}(\mathbf{w}) = \frac{1}{2}\sum_{i=1}^{N}\left\{y(x_i,\mathbf{w}) - t_i\right\}^2 + \frac{\lambda}{2}\|\mathbf{w}\|^2 \tag{1.3}$$

where λ governs the relative importance of the regularization term.

It is clear that the situation has improved dramatically.

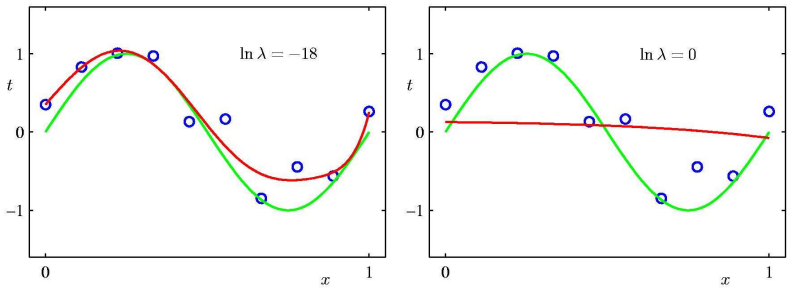


FIGURE 1.1.2. Using $M = 9$ and different values of the regularization constant.

A number of question remain:

- How does one determine the value of λ?
- How does one find the optimal value for M ?
- Are polynomials really the best approximations functions?

1.2. Basic Probability

1.2.1. Discrete probability. Just about everything we do derives from two basic rules of probability, the sum rule and the product rule. Bishop discusses a concrete example, read the book! We go straight to a more general discussion. Suppose we have two random variables X and Y that can take on the values $x_i,\ i = 1, \ldots, M$ and $y_j,\ j = 1, \ldots, L$ respectively. We now conduct a large number of trials N; for each trial the values of both X and Y are recorded. The number of times that X takes on the value of x_i is c_i and the number of times that Y takes on the value y_j is r_j. The number of times that X takes on the value x_i and, at the same trial, Y takes on the the value r_j is denoted by n_{ij}, see Figure 1.2.1.

.

The probability that X takes on the value x_i and Y the value y_j is called the *joint* probability and denoted by $p(X = x_i, Y = y_j)$. It is given by the fraction of the points falling in cell (i, j), i.e.

$$p(X = x_i, Y = y_j) = \frac{n_{ij}}{N},$$

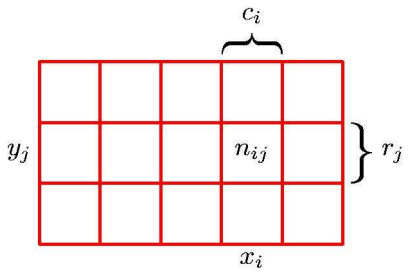


FIGURE 1.2.1. Sum and product rules.

keeping in mind that $N \to \infty$. Similarly

$$p(X = x_i) = \frac{c_i}{N},\ p(Y = y_j) = \frac{r_j}{N}.$$

Note that $c_i = \sum_j n_{ij}$ so that

$$p(X = x_i) = \sum_{j=1}^{L} p(X = x_i, Y = y_j).$$

This is the *sum rule* of probability, also known as the *marginal* probability.

If we consider only those instances for which $X = x_i$, then the fraction of such instances for which $Y = y_j$ is written as $p(Y = y_j|X = x_i)$, and is called the *conditional* probability of $Y = y_j$ given that $X = x_i$. It is the fraction of those points in column i that fall in cell (i, j) and is given by

$$p(Y = y_j|X = x_i) = \frac{n_{ij}}{c_i}$$

From this one can derive the following

$$\begin{aligned} p(X = x_i, Y = y_j) &= \frac{n_{ij}}{N} \\ &= \frac{n_{ij}}{c_i}\frac{c_i}{N} \\ &= p(Y = y_j|X = x_i)p(x_i). \end{aligned}$$

This is the *product* rule of probability.

Simplify notation by blurring the difference between a random variable and its value, and rewrite the sum– and product rules as

$$p(X) = \sum_Y p(X,Y) \tag{1.1}$$

$$p(X,Y) = p(Y|X)p(X). \tag{1.2}$$

The very important *Bayes' Theorem* follows directly from this relation by noting that $p(X,Y) = p(Y,X)$,

$$p(Y|X) = \frac{p(X|Y)p(Y)}{p(X)}. \tag{1.3}$$

Using the sum rule one can also write

$$p(X) = \sum p(X|Y)p(Y). \tag{1.4}$$

One can think of this denominator as a normalization constant required to ensure that the sum of the conditional probability on the left hand side of (1.3) over all the values of Y equal one.

$p(Y)$ is the *prior* probability, i.e. the probability in the absence of any further information, before any observation. $p(Y|X)$ is the *posterior* probability, i.e. the probability after we have observed X. $p(X|Y)$ is also known as the *likelihood* of Y . Note: For fixed Y , $p(X|Y)$ defines a probability over X. For fixed X, $p(X|Y)$ defined the likelihood of Y, which is not a probability.

Note: If $p(X|Y) = p(X)$, i.e. the conditional probability does not depend on Y, the two variables are *independent*. In particular this implies that for independent variables

$$p(X,Y) = p(X)p(Y).$$

EXAMPLE 2. Suppose you have developed a signature verification system with error curves shown in Figure 1.2.2. For more background you may want to read Chapter 15 of Modeling in Applied Mathematics, available at http://dip.sun.ac.za/~herbst/research/manusc

It is clear from the error curves that your system is not perfect—there does not exist a perfect system. It will on occasion accepts a forgery, or reject a genuine signature. Suppose you have installed your system in a bank and your system rejects a signature. What is the probability that it is actually a forgery? This is an important

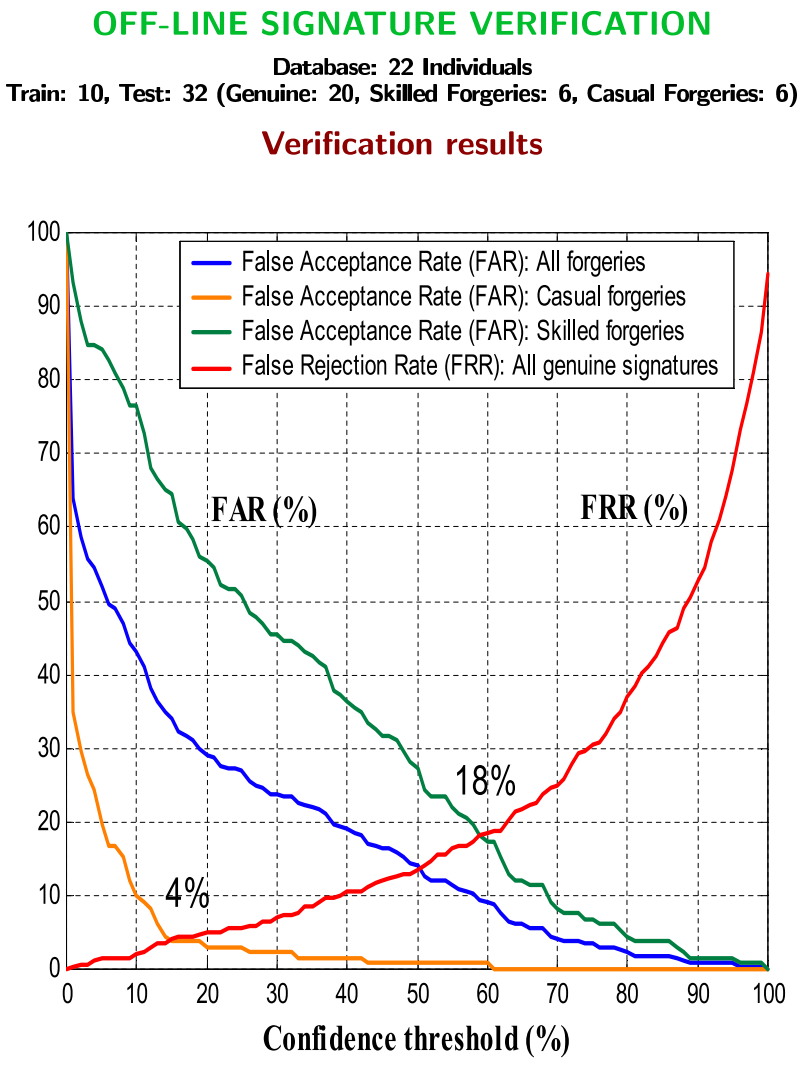


FIGURE 1.2.2. Error rates of a static signature verification system.

question. If your system rejects a large percentage of genuine signatures, it is basically useless. From the Figure is is clear that you cannot expect any better than a 4%

equal error rate (when the false acceptance rate equals the false rejection rate). Crime statistics are normally given as a number per one hundred thousand of the population. We don't have exact statistics of the number of forgers in a society but in countries with high crime rates, a number of forty per hundred thousand is typical. Use these numbers to calculate the probability that a signature is a forgery if rejected by the system. What should the error rate be before your system becomes useful, if by 'useful' you mean that every second signature that is rejected is actually a forgery? It is worth spending some time on this example.

First of all, let us introduce our random variables. Let S denote the signatures, i.e. S can take on two values, $S = t$ indicating a genuine signature, and $S = f$ a forgery. Similarly, let M indicates the measurement (the output of the system) so that $M = t$ and $M = f$ indicate a genuine signature and a forgery respectively, according to the system.

Let us start by considering what we can say about a signature without the benefit of our system, i.e. if we are presented with a signature how much confidence do we have that it is genuine? The important thing to realize is that we actually have at least, some faith, that it is genuine, based on prior knowledge about the behavior of people in your community. Using the crime statistics quoted above, let us suppose that it is a forgery is $\frac{40}{100\,000}$, i.e. $p(S = f) = \frac{40}{100\,000} = 0.0004$. Without the benefit of the system we conclude that it is extremely unlikely to encounter a forgery. Let us now consider how an actual measurement of the system affects this prior belief. Suppose the system indicates a forgery, we are interested in the posterior $p(S = f|M = f)$. Using Bayes' rule we write

$$p(S = f|M = f) = \frac{p(M = f|S = f)p(S = f)}{p(M = f)}.$$

The likelihood, $p(M = f|S = f)$ is an indication of how well the system describes the signatures, and is used as a correction of our prior assumption. According to the error curves we have $p(M = f|S = f) = 0.96$, and the normalization constant $p(M = f) = p(M = f|S = f)p(S = f)+p(M = f|S = t)p(S = t) = 0.96\times0.0004+0.04\times0.9996 = 0.0404$. The posterior is therefore given by $p(S = f|M = f) = \frac{0.96\times0.0004}{0.0404} = 0.0095$, or about 0.95%. Although the likelihood has changed our prior belief, we note that our system performs very poorly indeed—in the vast majority of cases we simply

cannot trust the system even if it does indicate a forgery. What is the reason for this?

If one really does not have any prior idea of the number of forgeries, one can use an uninformative prior $p(S = f) = 0.5$. In this case the posterior becomes

$$p(S = f|M = f) = \frac{0.96 \times 0.5}{0.96 \times 0.5 + 0.04 \times 0.5} = 0.96$$

which is just the output from the system. This really means that in the absence of any information from the system one believes that half the population are criminals. Hopefully this is not the case.

A better approach is to use a plausible prior but conduct multiple independent tests, i.e. we ask the person to sign again. Suppose we verify the signature twice and it comes out to be negative both times, we get

$$p(S = f|M_1 = f, M_2 = f) = \frac{p(M_1 = f, M_2 = f|S = f)p(S = f)}{p(M_1 = f, M_2 = f)}.$$

Now we make an important assumption, namely that M_1 and M_2 are *conditionally* independent, i.e. we assume that $p(M_1 = f, M_2 = f|S = f) = p(M_1 = f|S = f)p(M_2 = f|S = f)$. Also keeping in mind that

$$\begin{aligned} p(M_1, M_2) &= \sum_S p(M_1,M_2, S) \\ &= \sum_S p(M_1, M_2|S)p(s) \\ &= \sum_s P(M_1|S)p(M_2|S)p(S) \end{aligned}$$

we find that

$$\begin{aligned} p(S = f|M_1 = f, M_2 = f) &= \frac{0.96 \times 0.96 \times 0.0004}{0.96 \times 0.96 \times 0.0004 + 0.04 \times 0.04 \times 0.9996} \\ &= 0.1873 \end{aligned}$$

which is about 19%. Although we can't be too sure, our confidence that it is a forgery has increased after the signature has been rejected twice.

1.2.2. Probability densities. If the probability of a real-valued variable x falling in the interval $(x, x + \delta x)$ is given by $p(x)\delta x$ for $\delta x \to 0$,then $p(x)$ is called

the probability density over x. The probability that x will lie in the interval (a, b) is given by

$$p(x \in (a, b)) = \int_a^b p(x)dx.$$

Note: $p(x)$ satisfies

- $p(x) \geq 0$
- $\int_{-\infty}^{\infty} p(x)dx = 1.$

Change of variables: If $x = g(y)$ we want to figure out how the density functions transform, i.e. how $p_x(x)$ is transformed to $p_y(y)$. The probability that x falls in the range $(x, x + \delta x)$ has to be the same as the probability that y falls in the range $(y, y + \delta y)$, i.e. $p_x(x)\delta x = p_y(y)\delta y$, or

$$\begin{aligned} p_y(y) &= p_x(x) \left| \frac{dx}{dy} \right| \\ &= p_x(g(y)) \left| g'(y) \right|. \end{aligned}$$

The sum– and product rules become

$$\begin{aligned} p(x) &= \int p(x, y)dy \\ p(x, y) &= p(x|y)p(y). \end{aligned}$$

Example 3. A transformation function of particular importance in image processing is given by

$$x = g(y) = \int_0^y p_y(r)dr. \tag{1.5}$$

It now follows that

$$\begin{aligned} \frac{dx}{dy} &= \frac{dg}{dy} \\ &= p_y(y). \end{aligned}$$

The importance of this transformation function becomes clear if we now calculate the transformed density function,

$$\begin{aligned} p_x(x) &= p_y(y)\left|\frac{dy}{dx}\right| \\ &= p_y(y)\left|\frac{1}{p_y(y)}\right| \\ &= 1. \end{aligned}$$

We are given a gray scale image and need to enhance the contrast. One good way of doing it is through histogram equalization. For a given histogram $p_y(y)$ (normalized so that $\sum_y p_y(y) = 1$) the idea is to transform it so that $p_x(x) = 1$. This just the transformation calculated above, and we only need to decide how to calculate it in practice. If the image has L gray levels, i.e. y is the discrete variable with values $y_k,\ k = 0, \ldots, L-1$, and there are n pixels in the image, then $p_y(y_k) = \frac{n_k}{n}$ where n_k is the number of pixels that have gray level y_k. The discrete version of the transform (1.5) is then given by

$$\begin{aligned} x_k &= g(y_k) \\ &= \sum_{r=0}^{k} p_y(y_k) \\ &= \sum_{r=0}^{k} \frac{n_k}{n},\ k = 0, \ldots, L-1. \end{aligned}$$

The transformed image is obtained by mapping each pixel with gray level y_k to a pixel with gray level x_k. The result is shown in Figure 1.2.3. Notice how the contrast is enhanced by the histogram equalization.

The histograms are given by the following Figure. Do you have any explanation why the histogram after equalization does not show a flat curve?

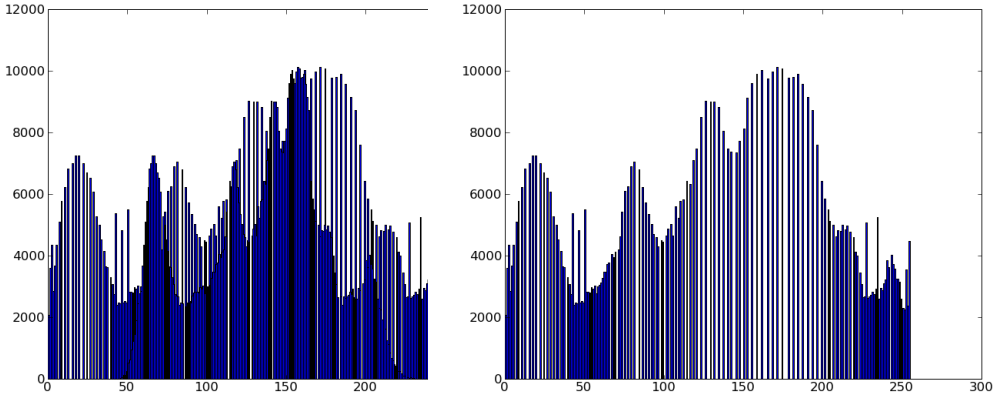
1.2.3. Expectation and covariances.

Discrete expectation:

$$\mathbb{E}[f] = \sum_x p(x) f(x).$$

(a) (b)

FIGURE 1.2.3. (a) Original image. (b) After histogram equalization.



(a) (b)

FIGURE 1.2.4. (a) Histogram of original image. (b) Histogram of equalized image.

Continuous expectation:

$$\mathbb{E}[f] = \int p(x)f(x)dx.$$

Note: If we have N points $x_n,\ n = 1, \ldots, N$ drawn from the probability distribution or density $p(x)$ the expectation is approximated by

$$E[f] \approx \frac{1}{N} \sum_{n=1}^{N} f(x_n).$$

This approximation becomes exact for $N \to \infty$. Note that formally what we are doing is to approximate $p(x)$ with samples drawn from it. There are different strategies for doing this, see for example Chapter 11 of Bishop's book.

Conditional expectation:

$$\mathbb{E}[f|y] = \sum_{x} p(y|x) f(x).$$

Variance:

$$\text{var}[f] = \mathbb{E}[(f(x) - \mathbb{E}[f(x)])^2].$$

Covariance:

$$\begin{aligned} cov[x, y] &= \mathbb{E}_{x,y}\left[\{x - \mathbb{E}[x]\}\{y - \mathbb{E}[y]\}\right] \\ &= \mathbb{E}_{x,y}[xy] - \mathbb{E}[x]\mathbb{E}[y]. \end{aligned}$$

For vector variables this becomes

$$\text{cov}[\mathbf{x}, \mathbf{y}] = \mathbb{E}_{x,y}\left[\{\mathbf{x} - \mathbb{E}[\mathbf{x}]\}\{\mathbf{y}^T - \mathbb{E}[\mathbf{y}^T]\}\right] = \mathbb{E}_{x,y}[\mathbf{x}\mathbf{y}^T] - \mathbb{E}[\mathbf{x}]\mathbb{E}[\mathbf{y}^T].$$

1.2.4. Gaussian Distribution. One of the most important distributions is the Gaussian– or normal distribution

$$N(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x - \mu)^2\right\}.$$

The Gaussian distribution is governed by exactly two parameters, the mean $\mu$and the variance σ^2. The square root of the variance σ is called the standard deviation and the reciprocal of the variance $1/\sigma^2$ is called the precision—the smaller the variance, the greater the precision, see Figure 1.2.5.

Note that

$$\mathbb{E}[x] = \int_{-\infty}^{\infty} N(x|\mu, \sigma^2) x dx = \mu$$

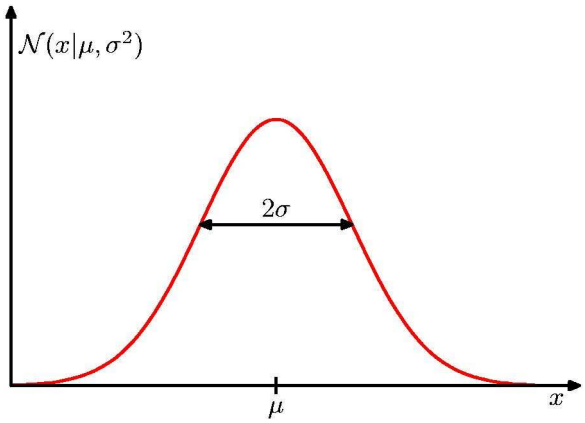


FIGURE 1.2.5. Gaussian density function.

and

$$\mathbb{E}[x^2] = \int_{-\infty}^{\infty} N(x|\mu,\sigma^2)x^2 dx = \mu^2 + \sigma^2.$$

Thus the variance is given by

$$\text{var}[x] = \mathbb{E}[x^2] - \mathbb{E}[x]^2 = \sigma^2.$$

For D-dimensional multivariate variables, we have

$$N(\mathbf{x}|\boldsymbol{\mu},\Sigma) = \frac{1}{|2\pi\Sigma|^{1/2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}.$$

Σ is called the *covariance* matrix.

PROBLEM 4. Jointly Gaussian. Assume that two Gaussian variables x and y, with means μ_x and μ_y and variances σ_x^2 and σ_y^2 respectively, are jointly Gaussian in the sense that

$$p(x,y) = N(\mathbf{x}|\boldsymbol{\mu},\Sigma),$$

where $\mathbf{x} = [x\ y]^T$ and $\Sigma = \begin{bmatrix} \sigma_x^2 & \sigma_{xy} \\ \sigma_{xy} & \sigma_y^2 \end{bmatrix}$.

- Show that x and y are independent if and only if $\sigma_{xy} = 0$.
- Show that $p(x|y)$ is Gaussian and calculate its mean and covariance.

Suppose we have N independent observations $\mathbf{x} = [x_1 \cdots x_N]^T$ all drawn from the same Gaussian distribution $N(x|\mu, \sigma^2)$. Such data points are said to be *independent* and i*dentically* distributed, or i.i.d. Furthermore, suppose that we do not know the values of μ and σ^2 and want to infer them from the observations. Because the data set $\mathbf{x}$ is i.i.d. we can write it in the form

$$p(\mathbf{x}|\mu, \sigma^2) = \prod_{n=1}^{N} N(x_n|\mu, \sigma^2).$$

Viewed as a function of μ and σ^2 this is the likelihood function for the Gaussian, see Figure 1.2.6.

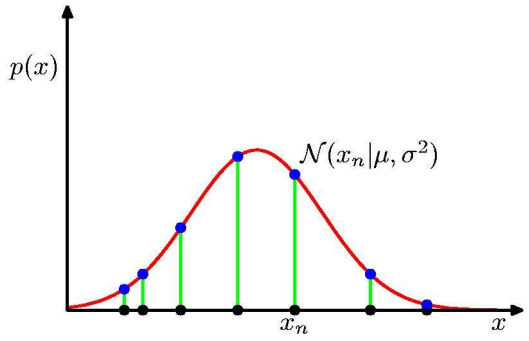


FIGURE 1.2.6. Likelihood function.

In order to infer the values of μ and σ^2 from the data, it is probably the most natural to consider the posterior density, $p(\mu, \sigma^2|\mathbf{x})$. However, it is the likelihood $p(\mathbf{x}|\mu, \sigma^2)$ that is usually more readily available. This is indeed the preferred approach and we'll return to it. In the mean time we choose μ and σ^2 such that the likelihood function is maximized, or rather, we maximize the log likelihood,

$$\ln p(x|\mu, \sigma^2) = -\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n - \mu)^2 - \frac{N}{2}\ln\sigma^2 - \frac{N}{2}\ln(2\pi).$$

Maximizing with respect to μ gives the *maximum likelihood* solution,

$$\mu_{ML} = \frac{1}{N}\sum_{n=1}^{N} x_n,$$

which is the *sample* mean. The maximum likelihood solution of the variance is similarly obtained as

$$\sigma^2_{ML} = \frac{1}{N}\sum_{n=1}^{N}(x_n - \mu_{ML})^2.$$

EXERCISE 5. The sample mean and sample variance are also random variables and one can calculate their mean. Show that

- $E[x_{ML}] = \mu$
- $E[(x_{ML} - \mu)^2] = \frac{\sigma^2}{N}$
- $E[\sigma^2_{ML}] = \frac{N-1}{N}\sigma^2$. Hint: Write

$$\begin{aligned}\sum_{n=1}^{N}(x_n - x_{ML})^2 &= \sum_{n=1}^{N}((x_n - \mu) - (x_{ML} - \mu))^2 \\ &= \sum_{n=1}^{N}(x_n - \mu)^2 - 2(x_{ML} - \mu)\sum_{n=1}^{N}(x_n - \mu) + N(x_{ML} - \mu) \\ &= \sum_{n=1}^{N}(x_n - \mu)^2 - N(x_{ML} - \mu)^2.\end{aligned}$$

From Exercise 5 it is clear that a bias is introduced into the sample variance. This is due to the fact that the variance is calculated with respect to the sample mean and not the real mean. An unbiased estimate is easily obtained (show it yourself!) by

$$\sigma^2_{UB} = \frac{1}{N-1}\sum_{n=1}^{N}(x_n - \mu_{ML})^2.$$

This is illustrated in Figure

1.2.5. More Curve Fitting. Given N input values $\mathbf{x} = [x_1 \ \cdots \ x_N]^T$ and corresponding target values $\mathbf{t} = [t_1 \ \cdots \ t_N]^T$, the problem is to estimate the target value t for a given x. The uncertainty over the target value is expressed using a probability distribution. Accordingly we assume that, given the value of x, the corresponding value of t has a Gaussian distribution with a mean equal to the value $y(x, \mathbf{w})$ of the polynomial (1.1), i.e.

$$p(t|x, \mathbf{w}, \beta) = N(t|y(x, \mathbf{w}), \beta^{-1})$$

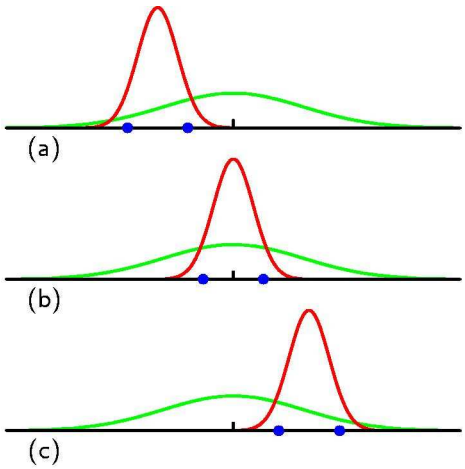


FIGURE 1.2.7. Illustration of how a bias arises. Averaged over three data sets the mean is correct but the variance is under-estimated.

where β is the precision, see Figure 1.2.8.

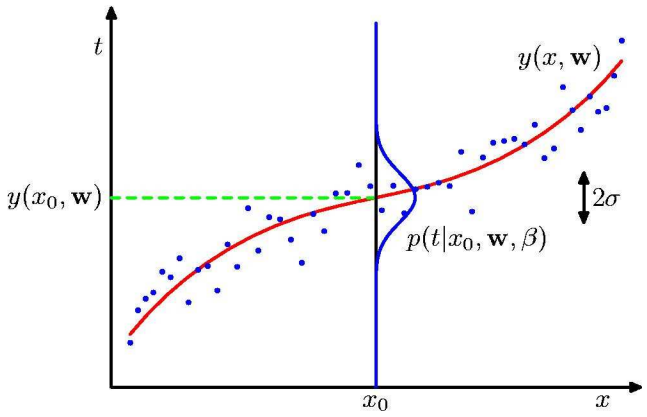


FIGURE 1.2.8. Gaussian distribution for t given x.

We use the training data $\{\mathbf{x}, \mathbf{t}\}$to determine the values of of the unknown parameters $\mathbf{w}$ and β using maximum likelihood. Assuming the data to be i.i.d.

$$p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = \prod_{n=1}^{N} N(t_n|y(x_n, \mathbf{w}), \beta^{-1}).$$

Taking the log-likelihood, we get

$$\ln p(\mathbf{t}|\mathbf{x}, \mathbf{w}, \beta) = -\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{N}{2}\ln\beta - \frac{N}{2}\ln(2\pi).$$

Note: Maximizing the negative log-likelihood with respect to $\mathbf{w}$ is the same as minimizing the sum-of-squares error function (1.2).

Maximizing log-likelihood with respect to β,

$$\frac{1}{\beta_{ML}} = \frac{1}{N}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}_{ML}) - t_n\}^2.$$

Having determined $\mathbf{w}$ and β we can now make predictions for new values of x. Using the probabilistic model, we have a probabilistic distribution over t,

$$p(t|x, \mathbf{w}_{ML}, \beta_{ML}) = N(t|y(x, \mathbf{w}_{ML}), \beta_{ML}^{-1}).$$

This is not entirely satisfactory, basically because we maximize the likelihood and not the posterior $p(\mathbf{w}|x, t, \beta)$ (we have already noted the problems with over fitting). According to Bayes' theorem we can write the posterior in terms of the likelihood

$$p(\mathbf{w}|x, t, \beta) \propto p(t|x, \mathbf{w}, \beta)p(\mathbf{w})$$

by introducing a prior over $\mathbf{w}$. Assume a simplified Gaussian distribution

$$p(\mathbf{w}|\alpha) = N(\mathbf{w}|0, \alpha^{-1}I) = \left(\frac{\alpha}{2\pi}\right)^{(M+1)/2}\exp\left\{-\frac{\alpha}{2}\mathbf{w}^T\mathbf{w}\right\},$$

where α is the precision and $M+1$ the order of the polynomial. Determine $\mathbf{w}$ by maximizing the posterior distribution. Taking negative log, the maximum posterior (MAP) is the minimum of

$$\frac{\beta}{2}\sum_{n=1}^{N}\{y(x_n, \mathbf{w}) - t_n\}^2 + \frac{\alpha}{2}\mathbf{w}^T\mathbf{w}.$$

Note:

(1) Minimizing the posterior is the same as minimizing the regularized sum-of-squares error (1.3).

(2) We are still using a point estimate of $\mathbf{w}$. This is not necessary—it may be better to use a softer approach and average over the distribution of $\mathbf{w}$

1.2.6. Bayesian curve fitting. In a Bayesian approach we consider all values of $\mathbf{w}$ by marginalizing over it. Given the data $\mathbf{x}$, $\mathbf{t}$ and the test point x we want to predict t, i.e. we need $p(t|x, \mathbf{x}, \mathbf{t})$. Using the product and sum rules, we get (omitting the α and β dependence for simplicity),

$$\begin{aligned} p(t|x, \mathbf{x}, \mathbf{t}) &= \int p(t, \mathbf{w}|x, \mathbf{x}, \mathbf{t}) d\mathbf{w} \\ &= \int p(t|x, \mathbf{w}, \mathbf{x}, \mathbf{t}) p(\mathbf{w}|x, \mathbf{x}, \mathbf{t}) d\mathbf{w} \\ &= \int p(t|x, \mathbf{w}) p(\mathbf{w}|\mathbf{x}, \mathbf{t}) d\mathbf{w}. \end{aligned}$$

It turns out (and it is not hard to show) that this is again a Gaussian of the form,

$$p(t|x, \mathbf{x}, \mathbf{t}) = N(t|m(x), s^2(x))$$

with the mean and covariance given by

$$m(x) = \beta \boldsymbol{\phi}(x)^T S \sum_{n=1}^{N} \boldsymbol{\phi}(x_n) t_n$$

$$s^2(x) = \beta^{-1} + \boldsymbol{\phi}(x)^T S \boldsymbol{\phi}(x). \tag{1.6}$$

Here the matrix S is given by

$$S^{-1} = \alpha I + \beta \sum_{n=1}^{N} \boldsymbol{\phi}(x_n) \boldsymbol{\phi}(x_n)^T$$

where I is the identity matrix, and the vector $\boldsymbol{\phi}$ is defined to have the components $\phi_i(x) = x^i,\ i = 1, \ldots, M$.

Note: The first term in (1.6) is the uncertainty in the predicted value of t due to the noise on the target values. The second term is because of the uncertainty in the parameters $\mathbf{w}$ and is a consequence of the Bayesian treatment, see Figure 1.2.9.

1.3. Model selection

The key phrases are, *over-fitting*, *validation* set, and *cross-validation*.

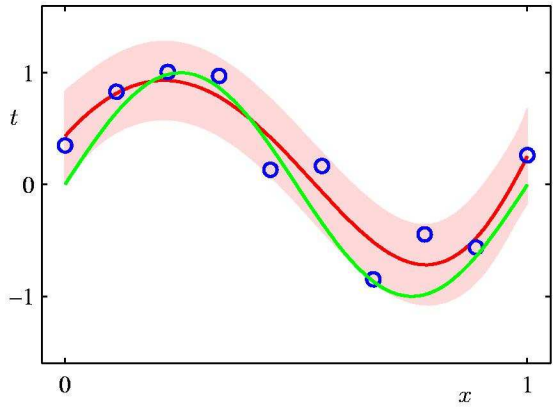


FIGURE 1.2.9. Predictive distribution resulting from a Bayesian treatment.

1.4. The curse of dimensionality

The basic problem is that in high dimensions one needs an awful lot of data in order to sample the space adequately. For example the volume of a sphere in D dimensions scale as r^D and can therefore be written as

$$V_D = K_D r^D.$$

If we now ask what is the fraction of the volume that lies between $r = 1 - \epsilon$ and $r = 1$, one finds that

$$\frac{V_D(1) - V_D(1-\epsilon)}{V_D(1)} = 1 - (1-\epsilon)^D \approx D\epsilon.$$

Thus for $\epsilon \approx \frac{1}{D}$ almost all of the volume is confined in the thin region at the edge of the sphere. This becomes more pronounced for larger D .

1.5. Decision Theory

In the signature verification example we referred to a decision boundary that allows one to accept a signature, or reject is as a forgery. The question is how does one find the optimal decision boundary. We classify a given signature x into one of two classes $C_k,\ k = 1, 2$ and we are interested in the probability of class C_k given a signature x, i.e. we are interested in $P(C_k|x)$.Using Bayes theorem we can write this

as

$$P(C_k|x) = \frac{p(x|C_k)P(C_k)}{p(x)}.$$

(Note that here we write probability with a capital P, and the densities with lower case p; elsewhere we are quite sloppy.) The prior probability $P(C_k)$ tells us the probability of class C_k in the absence of any further information. It simply tells us how likely it is that any given signature is a forgery. This represents our belief in the honesty of the population and may for example be based on crime statistics. $p(C_k|x)$ is the revised, posterior estimate, based on the actual measurement x. The class-based distribution $p(x|C_k)$requires knowledge about our system.

We want to minimize the chances of assigning x to the wrong class. Intuitively we assign x to the class with the highest posterior probability. Let us make this precise.

1.5.1. Minimizing misclassification. Two types of errors are possible, accepting a forgery, or rejecting a genuine signature. In general an input vector can be wrongly assigned to class C_1or vice versa. The probability of a mistake is therefore given by

$$P(\text{mistake}|x) = \begin{cases} P(C_1|x) & \text{if we choose } C_2 \\ P(C_2|x) & \text{if we choose } C_1 \end{cases}. \tag{1.1}$$

Since we may not always observe the same value for x, it may be a better idea to work with the marginal probability,

$$\begin{aligned} P(\text{mistake}) &= \int p(\text{mistake}, x)dx \\ &= \int_{R_1} p(\text{mistake}, x)dx + \int_{R_2} p(\text{mistake}, x)dx \\ &= \int_{R_1} p(\text{mistake}|x)p(x)dx + \int_{R_2} p(\text{mistake}|x)p(x)dx \end{aligned}$$

where R_k is the region where all input vectors x are assigned to C_k, see Figure 1.5.1. Making use of (1.1) this can be written as

$$\begin{aligned} P(\text{mistake}) &= \int_{R_1} P(C_2|x)p(x)dx + \int_{R_2} P(C_1|x)p(x)dx \\ &= \int_{R_1} p(x|C_2)P(C_2)dx + \int_{R_2} p(x|C_1)P(C_1)dx \end{aligned}$$

where the last line follows from Bayes' theorem. This is minimized if x is assigned to the region for which the integrand is the smallest, as illustrated in Figure 1.5.1.

Note that it is not possible to make an informed decision in the absence of knowledge of our system. In fact, following this approach we need to know quite a lot: essentially we need to know the joint densities $p(x, C_k)$. The 'learning' part of machine learning is about inferring joint densities or class conditional densities from the data. One of the reasons why signature verification is hard is because we do not have a model for forgeries. A forgery is everything that is not a genuine signature. The reason why it is at all possible to design efficient systems is because we are able to design a good model for each genuine signature.

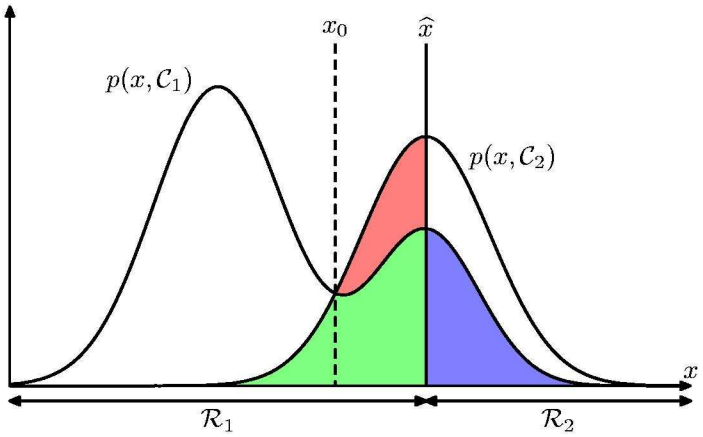


FIGURE 1.5.1. Optimal decision boundary.

EXERCISE 6. Find the optimal decision boundary for the signature verification system of Figure 1.2.2.

1.5.2. Minimizing expected loss. If there is a penalty involved in a particular misclassification it can be modeled by a loss function. Commercial banks for instance would rather accept more forgeries than alienating their customers. For them there is a heavy penalty on rejecting a genuine signature.

Let us generalize a little and assume that there are N different classes (or states of nature), and a different actions $\alpha_1, \ldots, \alpha_N$ every time we observe x. Action α_j for example, may indicate: put x in class C_j. The posterior is given as usual by

$$P(C_j|x) \propto p(x|C_j)P(C_j).$$

Let λ_{ij} expresses the penalty for taking action α_i when the true state is C_j. The loss or penalty for taking action α_i is the average penalty

$$R(\alpha_i|x) = \sum_{j=1}^{N} P(C_j|x)\lambda_{ij}.$$

Our decision rule is a function $\alpha(x)$ that maps each x to an action α_i. Our ideal decision rule then is one that minimizes the overall risk, defined as the expected loss for a given decision rule,

$$R = E\left[R(\alpha(x)|x)\right] = \int R(\alpha(x)|x)p(x)dx.$$

In order to minimize the expected loss, one chooses $\alpha(x)$ such that $R(\alpha(x)|x)$ is as small as possible. In practice we evaluate $R(\alpha_i|x)$ for all possible actions α_i. More precisely, for each x choose

$$\alpha(x) = \arg\min_{\alpha_i} R(\alpha_i|x).$$

1.5.3. Reject option. If it is essential that a correct classification be made, one might consider a reject option for samples falling the ambiguous region close to the decision boundary.

1.5.4. Inference and decision. Instead of two different stages, inference and decision, one might opt for a *discriminant* function. Three approaches

(1) Calculate the posterior density $p(C_K|x)$ from the class conditional densities $p(x|C_k)$. Then use decision theory. This is the so-called *generative* approach. The name stems from the fact that, since one builds a model of the system,

it is possible to use the model to generate data. The main drawback of this approach is the danger of overkill. We may not be interested in a detailed model of the system and in that case it may be better to use the available data to directly achieve ones aim.

(2) Find $p(C_k|x)$ directly, then use decision theory. This is the so-called discriminative approach. Instead of using the data to model the system, the available data is used to estimate parameters that directly leads to confidence measure of the classification.

(3) Find a function $f(x)$ that maps x directly onto a class label. This is perhaps the simples approach. The main drawback is that one does not get a confidence value with the estimate and therefore has no indication to what extent the estimate can be trusted.

Combining models. Suppose we have two independent test of a quantity, for example using both a signature and fingerprint for personal identification. If the two tests are (conditionally) independent we write

$$p(x_1, x_2|C_k) = p(x_1|C_k)p(x_2|C_k).$$

the posterior density is then given by

$$\begin{aligned} p(C_k|x_1, x_2) &\propto p(x_1, x_2|C_k)p(C_k) \\ &= p(x_1|C_k)p(x_2|C_k)p(C_k) \\ &\propto \frac{p(C_k|x_1)p(C_k|x_2)}{p(C_k)}. \end{aligned}$$

It is always a good idea to use different, independent models. Note that this also works if one obtain a second conditionally independent measurement using the same model.

1.5.5. Loss function for regression. Note that one also define a loss function for regression, not only the classification problem.

1.6. Information Theory

We are looking for a function $h(x)$ that gives us an idea of the information carried by a random variable x. If we know exactly what we are going to get, i.e. if $p(x) = 1$,

no information is conveyed; we need $h(p(x) = 1) = 0$. Also if we have two independent variables $p(x, y) = p(x)p(y)$ and we expect that $h(x, y) = h(x)+h(y)$. Thus we define

$$h(x) = -\log_2 p(x). \tag{1.1}$$

If a random variable is transmitted, the average amount of information that is transmitted is given by the *entropy*

$$H[x] = -\sum_x p(x) \log_2 p(x). \tag{1.2}$$

Distributions that are sharply peaked have smaller entropy than those spread more evenly, see Figure

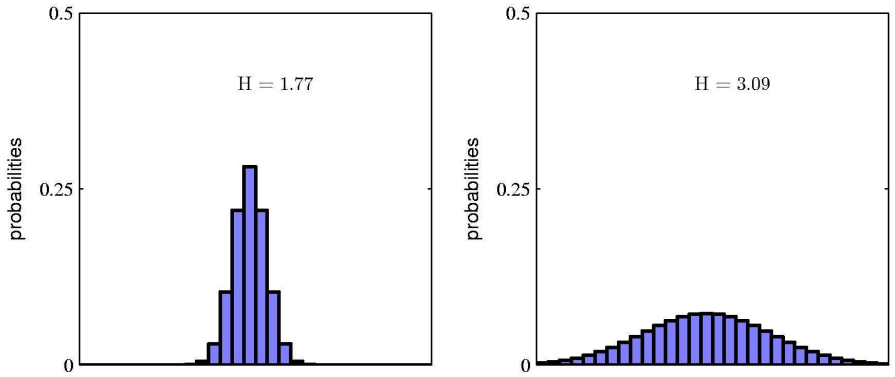


FIGURE 1.6.1. Entropy.

The maximum entropy configuration is obtained by maximizing

$$\widetilde{H} = -\sum_i p(x_i) \ln p(x_i) + \lambda \left(\sum_i p(x_i) - 1 \right)$$

where λ is the Lagrange multiplier. It follows easily that the maximum value of $\widetilde{H}$ is obtained by

$$p(x_i) = \frac{1}{M}.$$

For continuous variables we define the differential entropy is

$$H[x] = -\int p(x) \ln p(x) dx. \tag{1.3}$$

Maximize the differential entropy with constraints,

$$\begin{aligned}\int_{-\infty}^{\infty} p(x)dx &= 1\\ \int_{-\infty}^{\infty} xp(x)dx &= \mu\\ \int_{-\infty}^{\infty} (x-\mu)^2 p(x)dx &= \sigma^2.\end{aligned}$$

Using Lagrange multipliers and calculus of variations we find

$$p(x) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{(x-\mu)^2}{2\sigma^2}\right\}.$$

The maximum differential entropy is given by

$$H[x] = \frac{1}{2}\left\{1 + \ln(2\pi\sigma^2)\right\}.$$

Note: The entropy increases as the distribution becomes wider, i.e. as the variance increases.

The conditional entropy is given by

$$H[y|x] = -\int\int p(x,y) \ln p(y|x) dy dx$$

and it follows that

$$H[x,y] = H[x|y] + H[x].$$

1.6.1. Relative entropy and mutual information. Suppose that $q(x)$ is an approximation of $p(x)$. We want to compare the information carried by them. Accordingly define the relative entropy, or Kullback-Leibler entropy

$$\begin{aligned}KL(p||q) &= -\int p(x) \ln q(x) dx - \left(-\int p(x) \ln p(x) dx\right)\\ &= -\int p(x) \ln\left\{\frac{q(x)}{p(x)}\right\} dx.\end{aligned}$$

Note that $KL(p||q) \geq 0$ with equality if and only if $p(x) = q(x)$. Also note that $KL(p||q) \neq KL(q||p)$.

CHAPTER 2

PROBABILITY DISTRIBUTIONS

2.1. Binary Variables

If $x \in \{0, 1\}$ is a discrete variable, then the Bernoulli distribution is given by

$$\text{Bern}(x|\mu) = \mu^x (1-\mu)^{1-x}, \tag{2.1}$$

where μ is a parameter that defines the frequency of the two variables.

EXERCISE 7. Show that $\mathbb{E}[x] = \mu$, and $\text{var}[x] = \mu(1-\mu)$.

The goal is to estimate μ from data, $\mathcal{D} = \{x_1, \ldots, x_N\}$. If we draw the samples independently, the likelihood function is given by

$$p(\mathcal{D}|\mu) = \prod_{n=1}^{N} p(x_n|\mu) = \prod_{n=1}^{N} \mu^{x_n}(1-\mu)^{1-x_n}. \tag{2.2}$$

Maximize the log-likelihood, given by,

$$\ln p(\mathcal{D}|\mu) = \sum_{n=1}^{N} \ln p(x_n|\mu) = \sum_{n=1}^{N} \{x_n \ln \mu + (1-x_n)\ln(1-\mu)\},$$

to get the sample mean

$$\mu_{ML} = \frac{1}{N}\sum_{n=1}^{N} x_n.$$

If the number of times we observe $x = 1$ is n then

$$\mu_{ML} = \frac{n}{N}.$$

Note: If $N = 3$ and we observe $x = 1$ for all three samples, then the maximum likelihood estimate gives $\mu_{ML} = 1$, which is nonsense.

If we observe $x = 1$, m times out of N observations, it follows from (2.2) that the distribution of m is proportional to $\mu^m(1-\mu)^{N-m}$. The total number of ways in

which one can get m ones out of N samples is $\begin{pmatrix} N \\ m \end{pmatrix} = \frac{N!}{(M-m)!m!}$. The (normalized) binomial distribution is therefore

$$\text{Bin}(m|N,\mu) = \begin{pmatrix} N \\ m \end{pmatrix} \mu^m (1-\mu)^{N-m}. \tag{2.3}$$

EXERCISE 8. Show that $\sum_{m=0}^{N} \text{Bin}(m|N,\mu) = 1$.

It can be shown that

$$\begin{aligned} \mathbb{E}[m] &= N\mu \\ \text{var}[m] &= N\mu(1-\mu). \end{aligned}$$

2.1.1. The beta distribution. Given the data $\mathcal{D}$ we can calculate the maximum likelihood estimate of μ and therefore the likelihood $p(\mathcal{D}|\mu)$. As before, what we are really after is the posterior probability $p(\mu|\mathcal{D})$. According to Bayes' theorem this is proportional to $p(\mathcal{D}|\mu)p(\mu)$, where $p(\mu)$is the prior. The question is how to choose a suitable prior. In the absence of any other considerations, it is a good idea to choose it in such a way so that the posterior distribution has the same functional form as the likelihood. Accordingly, we introduce the beta distribution

$$\text{Beta}(\mu|a,b) = \frac{\Gamma(a+b)}{\Gamma(a)\Gamma(b)} \mu^{a-1}(1-\mu)^{b-1} \tag{2.4}$$

where

$$\Gamma(x) = \int_0^\infty u^{x-1} e^{-u} du,$$

and

$$\begin{aligned} \mathbb{E}[\mu] &= \frac{a}{a+b} \\ \text{var}[\mu] &= \frac{ab}{(a+b)^2(a+b+1)}. \end{aligned}$$

We use the beta distribution as a prior in a Bayesian setting; together with the binomial likelihood, the posterior is proportional to

$$p(\mu|m,l,a,b) \propto \mu^{m+a-1}(1-\mu)^{l+b-1},$$

where $l = N - m$, i.e. the number of times we observe $x = 0$. This is another beta distribution and properly normalized the posterior distribution becomes,

$$p(\mu|m, l, a, b) = \frac{\Gamma(m + a + l + b)}{\Gamma(m + a)\Gamma(l + b)}\mu^{m+a-1}(1 - \mu)^{l+b-1}.$$

Starting with the prior distribution (2.4) the effect if we observe $x = 1$, m times, and $x = 0$, l times is to increase the values of a and b by m and l respectively.

Note (important): The posterior can be used as a prior if subsequent observations are made. This allows for sequential estimates.

Given the data $\mathcal{D}$ the probability of observing $x = 1$,is given by

$$p(x = 1|\mathcal{D}) = \int_0^1 p(x = 1|\mu)p(\mu|\mathcal{D})d\mu = \int_0^1 \mu p(\mu|\mathcal{D})d\mu = \mathbb{E}[\mu|\mathcal{D}].$$

Using the mean of the posterior beta distribution, we get

$$p(x = 1|\mathcal{D}) = \frac{m + a}{m + a + l + b}.$$

Thinking of a and b as fictitious observations prior to any real observations (of $x = 1$, and $x = 0$ respectively), this is just the fraction of the total number (fictitious or not) of observations of $x = 1$. It also agrees with the maximum likelihood estimate as $m,\ l \to \infty$.

2.2. Multinomial Variables

Variables that can take on one of K mutually exclusive states can be represented by a K-dimensional vector of the form,

$$\mathbf{x} = [0\ \cdots\ 0\ 1\ 0\ \cdots\ 0]^T$$

where the 1 indicates the specific state. If $p(x_k = 1) = \mu_k$, then

$$p(\mathbf{x}|\boldsymbol{\mu}) = \prod_{k=1}^{K} \mu_k^{x_k}$$

where $\boldsymbol{\mu} = [\mu_1, \ldots, \mu_K]^T,\ \mu_k \geq 0$ and $\sum_k \mu_k = 1$.

Note:

$$\sum_{\mathbf{x}} p(\mathbf{x}|\boldsymbol{\mu}) = \sum_{k=1}^{K} \mu_k = 1$$

and

$$\mathbb{E}[\mathbf{x}|\boldsymbol{\mu}] = \sum_{\mathbf{x}} p(\mathbf{x}|\boldsymbol{\mu})\mathbf{x} = [\mu_1, \ldots, \mu_K]^T.$$

For a data set $\mathcal{D}$ consisting of N independent observations, the likelihood is,

$$p(\mathcal{D}|\boldsymbol{\mu}) = \prod_{n=1}^{N}\prod_{k=1}^{K}\mu_k^{x_{nk}} = \prod_{k=1}^{K}\mu_k^{(\sum_n x_{nk})} = \prod_{k=1}^{K}\mu_k^{m_k}.$$

Sufficient statistics for the likelihood is the sum

$$m_k = \sum_n x_{nk},$$

the number of observations of $x_k = 1$. Maximizing the log-likelihood $\ln p(\mathcal{D}|\boldsymbol{\mu})$ subject to the constraint $\sum_k \mu_k = 1$,is achieved through a Lagrange multiplier, i.e. by maximizing

$$\sum_{k=1}^{K} m_k \ln \mu_k + \lambda\left(\sum_{k=1}^{K}\mu_k - 1\right).$$

Setting the partial derivative with respect to μ_k equal to zero, gives

$$\mu_k = -\frac{m_k}{\lambda},$$

and the constraint gives

$$\lambda = -N.$$

Thus

$$\mu_k^{ML} = \frac{m_k}{N}.$$

Multinomial distribution

$$\text{Mult}(m_1, \ldots, m_K|\boldsymbol{\mu}, N) = \begin{pmatrix} N \\ m_1 \cdots m_K \end{pmatrix}\prod_{k=1}^{K}\mu_k^{m_k},$$

with

$$\sum_{k=1}^{K} m_k = N.$$

2.2.1. Dirichlet distribution. The Dirichlet distribution is used as a prior for the multinomial distribution

$$\mathrm{Dir}(\boldsymbol{\mu}|\boldsymbol{\alpha}) = \frac{\Gamma(\alpha_0)}{\Gamma(\alpha_1)\cdots\Gamma(\alpha_K)} \prod_{k=1}^{K} \mu_k^{\alpha_k - 1}$$

where

$$\alpha_0 = \sum_{k=1}^{K} \alpha_k.$$

2.3. The Gaussian Distribution

One dimension,

$$N(x|\mu, \sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left\{-\frac{1}{2\sigma^2}(x-\mu)^2\right\}$$

and in D dimensions,

$$N(\mathbf{x}|\boldsymbol{\mu}, \Sigma) = \frac{1}{|2\pi\Sigma|^{\frac{1}{2}}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu})\right\}, \tag{2.1}$$

where $|\cdot|$ denotes the determinant, and Σ is a $D \times D$ matrix.

The Mahalanobis distance is defined as,

$$\Delta^2 = (\mathbf{x}-\boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}-\boldsymbol{\mu}). \tag{2.2}$$

Note:

- Σ is symmetric.
- $\Delta = \text{const}$ defines the curves of constant probability.

If we write $U = [\mathbf{u}_1 \ \cdots \ \mathbf{u}_D]^T$ and $\Lambda = \mathrm{diag}(\lambda_1, \ldots, \lambda_D)$ where

$$\Sigma\mathbf{u}_j = \lambda_j\mathbf{u}_j, \ j = 1, \ldots, D,$$

we can write

$$\begin{bmatrix} \Sigma\mathbf{u}_1 & \ldots & \Sigma\mathbf{u}_D \end{bmatrix} = \begin{bmatrix} \lambda_1\mathbf{u}_1 & \ldots & \lambda_D\mathbf{u}_D \end{bmatrix}$$

or

$$\Sigma U = U\Lambda$$

where $\Lambda = \text{diag}(\lambda_1 \dots \lambda_D)$. This means that

$$\Sigma = U\Lambda U^T$$

or

$$\Sigma^{-1} = U\Lambda^{-1}U^T$$

assuming, of course, that the eigenvectors are normalized. Multiplying out this becomes,

$$\Sigma = \sum_{i=1}^{D} \lambda_i \mathbf{u}_i \mathbf{u}_i^T.$$

Since the eigenvalues of $\Sigma^{-1}$are given by λ^{-1}, we also have,

$$\Sigma^{-1} = \sum_{i=1}^{D} \frac{1}{\lambda_i} \mathbf{u}_i \mathbf{u}_i^T.$$

If we now change variables

$$\mathbf{y} = U^T(\mathbf{x} - \boldsymbol{\mu}) \tag{2.3}$$

in (2.2) we get

$$\begin{aligned} \Delta^2 &= \mathbf{y}^T U^T \Sigma^{-1} U \mathbf{y} \\ &= \mathbf{y}^T \Sigma^{-1} \mathbf{y} \\ &= \sum_{i=1}^{D} \frac{y_i^2}{\lambda_i} \end{aligned}$$

The transformation (2.3) is simply a rotation so that the principle axes coincide with the coordinate axes, see Figure 2.3.1

In these coordinates the Gaussian becomes

$$p(\mathbf{y}) = N(\mathbf{y}|\mathbf{0}, \Lambda).$$

Note:

- $\mathbb{E}[\mathbf{x}] = \boldsymbol{\mu}$.
- $\text{cov}[\mathbf{x}] = \mathbb{E}[(\mathbf{x} - \boldsymbol{\mu})(\mathbf{x} - \boldsymbol{\mu})^T] = \Sigma$.

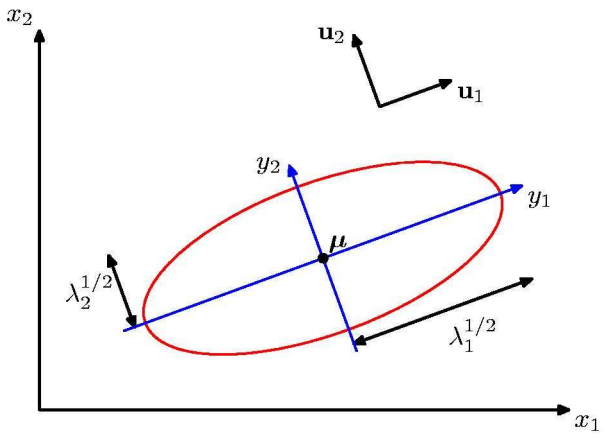


FIGURE 2.3.1. Rotating the principle axes.

Conditional Gaussian distribution. The following identity will be used extensively,

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix}^{-1} = \begin{bmatrix} M & -MBD^{-1} \\ -D^{-1}CM & D^{-1} + D^{-1}CMBD^{-1} \end{bmatrix} \tag{2.4}$$

where

$$M = \left(A - BD^{-1}C\right)^{-1}.$$

EXERCISE 9. Verify the identity by showing that

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \times \begin{bmatrix} M & -MBD^{-1} \\ -D^{-1}CM & D^{-1} + D^{-1}CMBD^{-1} \end{bmatrix} = I.$$

Suppose that $\mathbf{x} \in \mathbb{R}^D$ is a Gaussian random variable $\mathbf{x} \sim N(\mathbf{x}|\boldsymbol{\mu}, \Sigma)$. Partition

$$\mathbf{x} = \begin{bmatrix} \mathbf{x}_a \\ \mathbf{x}_b \end{bmatrix}$$

where $\mathbf{x}_a$ is the first M components of $\mathbf{x}$. We also partition the mean and covariance matrix,

$$\boldsymbol{\mu} = \begin{bmatrix} \boldsymbol{\mu}_a \\ \boldsymbol{\mu}_b \end{bmatrix},$$

and

$$\Sigma = \begin{bmatrix} \Sigma_{aa} & \Sigma_{ab} \\ \Sigma_{ba} & \Sigma_{bb} \end{bmatrix}$$

where $\Sigma_{aa} = \Sigma_{aa}^T,\ \Sigma_{bb} = \Sigma_{bb}^T,\ \Sigma_{ab} = \Sigma_{ba}^T$. Also partition the *precision* matrix,

$$\Lambda = \Sigma^{-1},$$

as

$$\Lambda = \begin{bmatrix} \Lambda_{aa} & \Lambda_{ab} \\ \Lambda_{ba} & \Lambda_{bb} \end{bmatrix}.$$

We are looking for $p(\mathbf{x}_a|\mathbf{x}_b)$. In principle this is very easy to calculate. All we need to do is to fix $\mathbf{x}_b$ in the joint distribution, and to extract the quadratic form of the remaining $\mathbf{x}_a$, see the textbook for details to find,

$$\Sigma_{a|b} = \Lambda_{aa}^{-1},$$

and

$$\boldsymbol{\mu}_{a|b} = \boldsymbol{\mu}_a - \Lambda_{aa}^{-1}\Lambda_{ab}(\mathbf{x}_b - \boldsymbol{\mu}_b).$$

Making use of the matrix identity this can also be expressed in terms of the partitioned covariance matrix

$$\Sigma_{a|b} = \Sigma_{aa} - \Sigma_{ab}\Sigma_{bb}^{-1}\Sigma_{ba} \tag{2.5}$$

and

$$\boldsymbol{\mu}_{a|b} = \boldsymbol{\mu}_a + \Sigma_{ab}\Sigma_{bb}^{-1}(\mathbf{x}_b - \boldsymbol{\mu}_b). \tag{2.6}$$

Note that the conditional mean takes the form

$$\boldsymbol{\mu}_{a|b} = A\mathbf{x}_b + \mathbf{b}$$

and that the conditional covariance is independent of $\mathbf{x}_a$. This is an example of a linear Gaussian model.

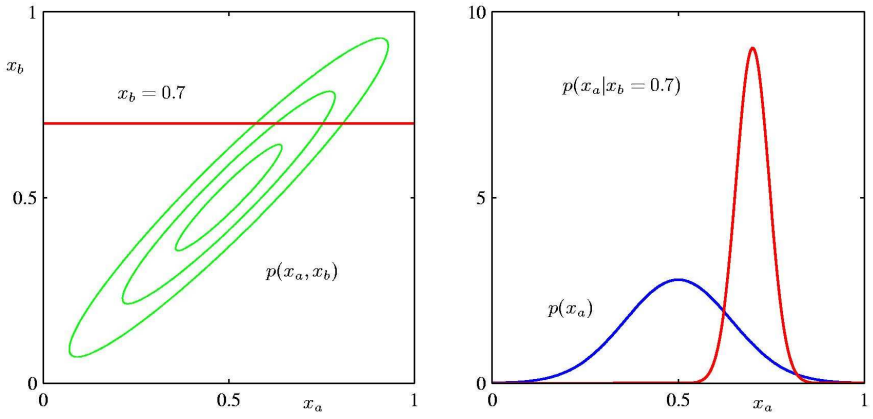
2.3.1. Marginal Gaussian distribution. We want to compute the marginal distribution

$$p(\mathbf{x}_a) = \int p(\mathbf{x}_a, \mathbf{x}_b) d\mathbf{x}_b.$$

Straightforward but tedious calculations show that

$$p(\mathbf{x}_a) = N(\mathbf{x}_a|\boldsymbol{\mu}_a, \Sigma_{aa}).$$

This means that the marginal distribution can be simply read off the partitioned mean and covariance. This illustrated in Figure 2.3.2



(a) (b)

FIGURE 2.3.2. (a) $2D$ Gaussian distribution. (b) Marginal and conditional distributions.

2.3.2. Bayes' theorem for Gaussian variables. Assume we are given a Gaussian marginal distribution $p(\mathbf{x})$ and a Gaussian conditional distribution $p(\mathbf{y}|\mathbf{x})$ which has a mean that is linear in $\mathbf{x}$ and a covariance independent of $\mathbf{x}$. We wish to find the marginal distribution $p(\mathbf{y})$ and the conditional distribution $p(\mathbf{x}|\mathbf{y})$. Let

$$\begin{aligned} p(\mathbf{x}) &= N(\mathbf{x}|\boldsymbol{\mu}, \Lambda^{-1}) \\ p(\mathbf{y}|\mathbf{x}) &= N(\mathbf{y}|A\mathbf{x} + \mathbf{b}, L^{-1}). \end{aligned}$$

If $\mathbf{x} \in \mathbb{R}^M$ and $\mathbf{y} \in \mathbb{R}^{\mathbb{D}}$ then A is an $D \times M$ matrix. Again an essentially straightforward but rather tedious calculation gives

$$\begin{aligned} \mathbb{E}[\mathbf{y}] &= A\boldsymbol{\mu} + \mathbf{b} \\ \text{cov}[\mathbf{y}] &= L^{-1} + A\Lambda^{-1}A^T, \end{aligned}$$

and

$$\begin{aligned}\mathbb{E}[\mathbf{x}|\mathbf{y}] &= \left(\Lambda + A^T L A\right)^{-1}\left(A^T L(\mathbf{y}-\mathbf{b}) + \Lambda\boldsymbol{\mu}\right)\\ \operatorname{cov}[\mathbf{x}|\mathbf{y}] &= \left(\Lambda + A^T L A\right)^{-1}.\end{aligned}$$

2.3.3. Maximum likelihood. Given an i.i.d. data set, $X = [\mathbf{x}_1 \dots \mathbf{x}_N]$ drawn from a multivariate Gaussian distribution, the log likelihood is given by

$$\ln p(X|\boldsymbol{\mu}, \Sigma) = -\frac{ND}{2}\ln(2\pi) - \frac{N}{2}\ln|\Sigma| - \frac{1}{2}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu})^T\Sigma^{-1}(\mathbf{x}_n - \boldsymbol{\mu}).$$

The sufficient statistics are given by $\sum_{n=1}^{N}\mathbf{x}_n$ and $\sum_{n=1}^{N}\mathbf{x}_n\mathbf{x}_n^T$. Maximizing give,

$$\boldsymbol{\mu}_{ML} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

and

$$\Sigma_{ML} = \frac{1}{N}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu}_{ML})(\mathbf{x}_n - \boldsymbol{\mu}_{ML})^T.$$

The unbiased estimate is given by

$$\Sigma_{ML} = \frac{1}{N-1}\sum_{n=1}^{N}(\mathbf{x}_n - \boldsymbol{\mu}_{ML})(\mathbf{x}_n - \boldsymbol{\mu}_{ML})^T.$$

2.3.4. Sequential estimation. Let us say we have estimated the mean $\boldsymbol{\mu}_{ML}^{N-1}$using $N-1$ observations, and how we get another observation, $\mathbf{x}_N$. It follows that

$$\boldsymbol{\mu}_{ML}^{N} = \boldsymbol{\mu}_{ML}^{N-1} + \frac{1}{N}(\mathbf{x}_N - \boldsymbol{\mu}_{ML}^{N-1}).$$

A more general formula is the Robbins-Monro formula. Given a pair of random variables, θ and z governed by a joint distribution $p(z,\theta)$. The conditional expectation of z given θ defines a deterministic function $f(\theta)$ given by

$$f(\theta) = \mathbb{E}[z|\theta] = \int zp(z|\theta)dz,$$

illustrated by Figure 2.3.3.

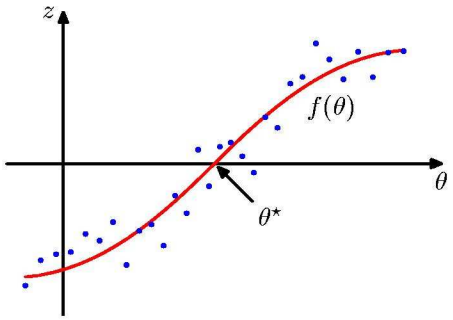


FIGURE 2.3.3. The Robbins-Monro algorithm.

The idea is to find the root $f(\theta^*) = 0$ when we observe z one at a time. Assuming

$$\mathbb{E}[(z - f)^2|\theta] < \infty,$$

$f(\theta) > 0$ if $\theta > \theta^*$ and $f(\theta) < 0$ if $\theta < \theta^*$, the algorithm is given by

$$\theta^N = \theta^{N-1} - a_{N-1} z(\theta^{N-1}) \tag{2.7}$$

where $z(\theta^{N-1})$is an observed value for z when θ takes the value θ^{N-1}. The coefficients $\{a_N\}$ represent a sequence of positive numbers that satisfy

$$\begin{aligned} \lim_{N\to\infty} a_N &= 0 \\ \sum_{N=1}^{\infty} a_N &= \infty \\ \sum_{N=1}^{\infty} a_N^2 &< \infty. \end{aligned}$$

The Robbins-Monro algorithm converges to the correct root with probability one.

Use the Robbins-Monro algorithm to solve general maximum likelihood. By definition

$$-\frac{\partial}{\partial\theta}\left\{\frac{1}{N}\sum_{n=1}^{N}\ln p(\mathbf{x}_n|\theta)\right\}\Bigg|_{\theta_{ML}} = 0.$$

Exchanging the derivative and summation, and taking the limit $N \to \infty$, we have

$$-\lim_{N\to\infty}\frac{1}{N}\sum_{n=1}^{N}\frac{\partial}{\partial\theta}\ln p(\mathbf{x}_n|\theta) = \mathbb{E}\left[-\frac{\partial}{\partial\theta}\ln p(\mathbf{x}_n|\theta)\right].$$

Thus we need to solve

$$\mathbb{E}\left[\frac{\partial}{\partial\theta}\ln p(\mathbf{x}_n|\theta)\right] = 0,$$

which implies finding the root of a regression function. Robbins-Monro now takes the form

$$\theta^N = \theta^{N-1} - a_{N-1}\frac{\partial}{\partial\theta^{N-1}}\left[-\ln p(\mathbf{x}_N|\theta^{N-1})\right].$$

Estimating the mean of a Gaussian distribution means that $\theta^N := \mu_{ML}^N$ and the random value z is given by

$$z = -\frac{\partial}{\partial\mu_{ML}}\ln p(x|\mu_{ML},\sigma^2) = -\frac{1}{\sigma^2}(x-\mu_{ML}).$$

The distribution of z is Gaussian with mean $-(\mu-\mu_{ML,})/\sigma^2$ see Figure

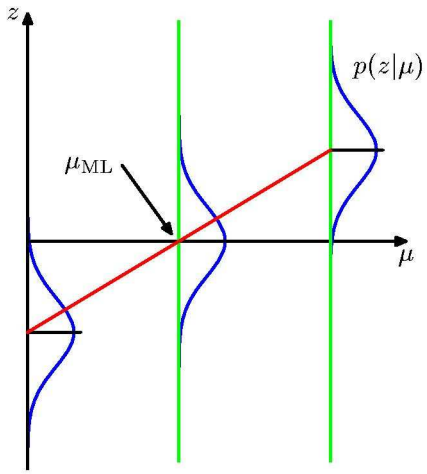


FIGURE 2.3.4. Robbins-Monro for Gaussians.

Need to choose, $a_N = \sigma^2/N$.

2.3.5. Bayesian inference for the Gaussian. Suppose that the variance σ^2 is known. We need to infer the mean μ given data $\mathbf{x} = \{x_1,\ldots,x_N\}$. The likelihood is given by

$$p(\mathbf{x}|\mu) = \prod_{n=1}^{N} p(x_n|\mu) = \frac{1}{(2\pi\sigma^2)^{N/2}}\exp\left\{-\frac{1}{2\sigma^2}\sum_{n=1}^{N}(x_n-\mu)^2\right\}.$$

A conjugate prior $p(\mu)$ takes the form of a Gaussian,

$$p(\mu) = \mathcal{N}(\mu|\mu_0, \sigma_0^2)$$

so that the posterior distribution becomes

$$p(\mu|\mathbf{x}) \propto p(\mathbf{x}|\mu)p(\mu).$$

As before

$$p(\mu|\mathbf{x}) = \mathcal{N}(\mu|\mu_N, \sigma_N^2)$$

where

$$\begin{aligned} \mu_N &= \frac{\sigma^2}{N\sigma_0^2 + \sigma^2}\mu_0 + \frac{N\sigma_0^2}{N\sigma_0^2 + \sigma^2}\mu_{ML} \\ \frac{1}{\sigma_N^2} &= \frac{1}{\sigma_0^2} + \frac{N}{\sigma^2} \end{aligned}$$

where μ_{ML} is the sample mean.

Note that the prior plays the role of a fictitious observation followed by N 'real' observations. The observations serve to 'correct' or modify the prior assumption, see Figure 2.3.5

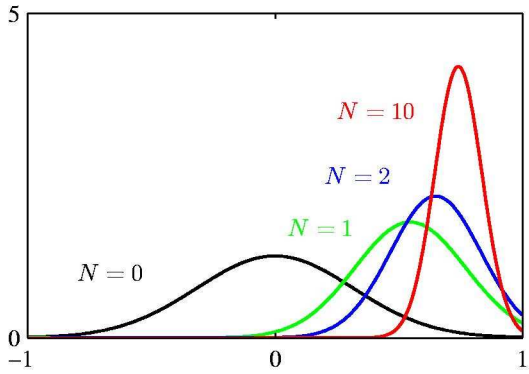


FIGURE 2.3.5. Bayesian inference for the mean μ.

Note that this leads naturally to a sequential view.

Now suppose that the mean is known and we want to infer the variance, or more conveniently the precision $\lambda = \frac{1}{\sigma^2}$. The likelihood is given by

$$p(\mathbf{x}|\lambda) = \prod_{n=1}^{N} \mathcal{N}(x_n|\mu, \lambda^{-1}) \propto \lambda^{N/2} \exp\left\{-\frac{\lambda}{2}\sum_{n=1}^{N}(x_n - \mu)^2\right\}.$$

The conjugate prior is a Gamma distribution

$$\mathrm{Gam}(\lambda|a, b) = \frac{1}{\Gamma(a)} b^a \lambda^{a-1} \exp(-b\lambda).$$

The Gamma distribution is illustrated in Figure

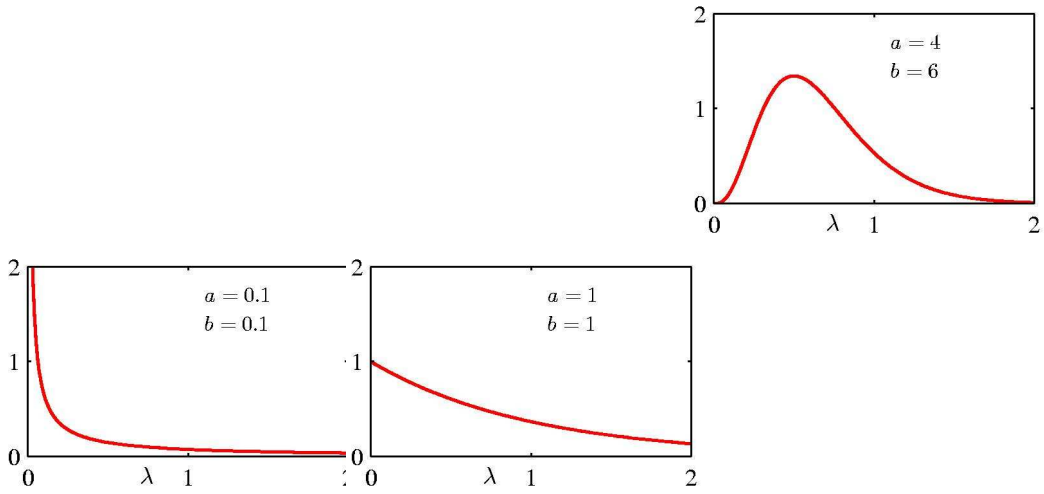


FIGURE 2.3.6. The Gamma distribution.

The mean and variance are given by

$$\begin{aligned} \mathbb{E}[\lambda] &= \frac{a}{b} \\ \mathrm{var}[\lambda] &= \frac{a}{b^2}. \end{aligned}$$

For prior $\mathrm{Gam}(\lambda|a_0, b_0)$ the posterior becomes

$$p(\lambda|\mathbf{x}) \propto \lambda^{a_0-1}\lambda^{N/2} \exp\left\{-b_0\lambda - \frac{\lambda}{2}\sum_{n=1}^{N}(x_n - \mu)^2\right\}$$

which is a Gamma distribution of the form $\text{Gam}(\lambda|a_N, b_N)$ where

$$\begin{aligned} a_N &= a_0 + \frac{N}{2} \\ b_N &= b_0 + \frac{1}{2}\sum_{n=1}^{N}(x_n - \mu)^2 = b_0 + \frac{N}{2}\sigma^2_{ML}. \end{aligned}$$

Note that N observations increase a_0 by $\frac{N}{2}$. One therefore interpret a_0 as being the equivalent of $2a_0$ prior observations. The N observations also increase the value of b_0 by $N\sigma^2_{ML}/2$. If we want to interpret b_0 as being the result of the $2a_0$ prior observations, we have to write $b_0 = a_0\frac{b_0}{a_0}$ so that we conclude that the prior is the equivalent of $2a_0$observations with variance $\frac{b_0}{a_0}$.

If both the mean and variance are unknown, the conjugate prior is a normal-Gamma distribution.

Note: The Student's t-distribution is much more robust to outliers than the Gaussian.

2.4. Linear transformations of gaussians

There is another important and useful property of Gaussian pdf's namely that a linear transformation results in another Gaussian. Let

$$\mathbf{x} = A\mathbf{y}$$

where A is a matrix. Substituting into (2.1) gives,

$$\begin{aligned} \mathcal{N}(\mathbf{y}|\boldsymbol{\mu}_y, \Sigma_y) &\propto \exp(-\frac{1}{2}(A\mathbf{y} - \boldsymbol{\mu})^T\Sigma^{-1}(A\mathbf{y} - \boldsymbol{\mu})) \\ &= \exp(-\frac{1}{2}(\mathbf{y} - A^{-1}\boldsymbol{\mu})^T A^T\Sigma^{-1}A(\mathbf{y} - A^{-1}\boldsymbol{\mu})). \end{aligned}$$

It now easily follows that $\boldsymbol{\mu}_y = A^{-1}\boldsymbol{\mu}$ and $\Sigma_y = A^{-1}\Sigma A^{-T}$. There is one catch however, and that is we clear assume that A invertible. Although it is a linear transformation, it is a very special kind of linear transformation. Consider for instance the following transformation

$$z = x_1 + x_2$$

where both x_1 and x_2 are Gaussian random variables, let us say with means μ_1 and μ_2 and variances σ_1^2 and σ_2^2. This is also a linear transformation but in this case the

transformation matrix in not invertible. The previous derivation does not cover this result, and the question remains, is a linear transformation of a Gaussian again a Gaussian? The next exercise guides you through a simple construction.

EXERCISE 10. Let $z_1 = x_1 + x_2$ and define $z_2 = x_2$. Write down the linear transformation $\mathbf{z} = A\mathbf{x}$ where $\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \end{bmatrix}$ and $\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$. Show that A is invertible. Assume that x_1 and x_2 are jointly Gaussian and write down the joint distribution for $\mathbf{x}$. From this you should be able to calculate the Gaussian for $\mathbf{z}$ from which you can obtain the distribution of $z_1$by marginalization. Consider the cases when x_1 and x_2 are independent, and when they are dependent, separately.

What about random variable that are not Gaussian? The next exercise guides you through an example.

EXERCISE 11. Let $z = x_1 + x_2$ and assume that $x_1$and x_2 are independent random variables with, in this case, the pdf's given by the general functions $p_1(x_1)$ and $p_2(x_2)$ respectively. Try the same trick as in the previous exercise, i.e. write $z_1 = x_1 + x_2$ and $z_2 = x_2$. If $p(x_1, x_2)$ is the joint pdf we can now transform to the z-coordinates, $p(z_1, z_2) = p(x_1(z_1, z_2), x_2(z_1, z_2)) = p_1(x_1(z_1, z_2))p_2(x_2(z_1, z_2))$. If we do the transformation, and marginalize, $p(z_1) = \int p(z_1, z_2)dz_2$, we can write the pdf of z in the form of a convolution.

2.5. Nonparametric methods

Nonparametric methods make few assumptions, if any, about the form of the distribution. The simples is the histogram where N observations are distributed over a number of bins, each of width Δ_i. If n_i observations fall in bin i, then the probability for each bin is given by

$$p_i = \frac{n_i}{N\Delta_i}.$$

This leads to a piecewise constant density function $p(x)$. There are two major problems

(1) $p(x)$ is not smooth

(2) The number of observations required is prohibitive in higher dimensions, scales like M^D where M is the number of bins and D the dimension.

2.5.1. Kernel density estimators. Suppose the observations are drawn from an unknown distribution $p(x)$. The probability of an observation falling inside a region $\mathcal{R}$ is given by

$$P = \int_{\mathcal{R}} p(x)dx.$$

If we have N samples drawn from $p(x)$,each having a probability P of falling inside $\mathcal{R}$ the number of points K falling inside $\mathcal{R}$ is given by the Binomial distribution

$$\text{Bin}(K|N,P) = \frac{N!}{K!(N-K)!}P^K(1-P)^{N-K}.$$

The mean fraction of points falling inside the region $\mathbb{E}[K/N] = P$ with variance $\text{var}(K/N) = P(1-P)/N$. For large N the variance is small so that

$$K \approx NP.$$

If $\mathcal{R}$ is small enough so that $p(x)$ does not vary much, we get

$$P = p(x)V$$

where V is the volume of $\mathcal{R}$. Thus

$$p(x) = \frac{K}{NV}.$$

This can be used in one of two ways. Fix K and determine the value of V from the data—the $K-$nearest neighbor technique or, fix V and determine K from the data—the kernel approach.

Define

$$k(\mathbf{u}) = \begin{cases} 1, & |u_i| \leq \frac{1}{2} \\ 0, & \text{otherwise} \end{cases}$$

which is a unit cube centered at the origin. Note that a cube with sides h, centered at $\mathbf{x}_n$ is given by $k((\mathbf{x}-\mathbf{x}_n)/h)$. The total number of points lying in the cube with side h centered at $\mathbf{x}$ is given by

$$K = \sum_{n=1}^{N} k\left(\frac{\mathbf{x}-\mathbf{x}_n}{h}\right)$$

so that

$$p(x) = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{h^D}k\left(\frac{\mathbf{x}-\mathbf{x}_n}{h}\right).$$

This can be reinterpreted as the sum of N cubes each centered at x_n. A smoother density estimate is obtained if a smoother kernel is used, e.g. a Gaussian.

2.5.2. Nearest-neighbor. We now fix K and find the value of V from the data. If we want to estimate the density at $\mathbf{x}$ we consider a small sphere centered at $\mathbf{x}$ and grows the radius until K points fall inside the sphere. The density is then given by

$$p(x) = \frac{K}{NV}$$

where V is the volume of the sphere.

One can extend the technique to classification. To do this the $K-$nearest-neighbor density estimation is applied to each class separately before Bayes' theorem is used. Suppose there are N_k points in class $\mathcal{C}_k$ with N points in total. Given a new point $\mathbf{x}$ we draw a sphere round it containing exactly K points, irrespective of class. Suppose the sphere contains K_k points from class $\mathcal{C}_k$, then the estimate of the density of each class is given by

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{K_k}{N_k V},$$

the unconditional density is given by

$$\begin{aligned} p(\mathbf{x}) &= \sum_k p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k) \\ &= \sum_k \frac{K_k}{N_k V}\frac{N_k}{N} \\ &= \frac{K}{NV}, \end{aligned}$$

where the class priors are given by

$$p(\mathcal{C}_k) = \frac{N_k}{N}.$$

Bayes' theorem now gives

$$p(\mathcal{C}_k|\mathbf{x}) = \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{p(\mathbf{x})} = \frac{K_k}{K}.$$

Assigning $\mathbf{x}$ to the class with the largest posterior probability , corresponding to the largest value of $\frac{K_k}{K}$, minimizes the probability of misclassification.

CHAPTER 3

Linear models for regression

3.1. Linear basis function models

Consider linear combinations of fixed, nonlinear functions

$$y(\mathbf{x}, \mathbf{w}) = w_0 + \sum_{j=1}^{M-1} w_j \phi_j(\mathbf{x})$$

where the ϕ_j are knows as the basis functions. Can also be written as

$$y(\mathbf{x}, \mathbf{w}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}).$$

3.1.1. Maximum likelihood and least squares. Assume that the target value is given by a deterministic function plus noise,

$$t = y(\mathbf{x}, \mathbf{w}) + \epsilon$$

where ϵ is a Gaussian random variable with zero mean and precision β, i.e.

$$p(t|\mathbf{x}, \mathbf{w}, \beta) = \mathcal{N}(t|y(\mathbf{x}, \mathbf{w}), \beta^{-1}).$$

Given data $X = \{\mathbf{x}_1, \ldots, \mathbf{x}_N\}$ with corresponding target values $t_1, \ldots, t_N$ (drawn independently from a Gaussian distribution), the likelihood is given by,

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

and

$$\begin{aligned} \ln(p(\mathbf{t}|X, \mathbf{w}, \beta) &= \sum_{n=1}^{N} \ln \mathcal{N}(t_n|\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}_n), \beta^{-1}) \\ &= \frac{N}{2} \ln \beta - \frac{N}{2} \ln(2\pi) - \beta E_D(\mathbf{w}) \end{aligned} \tag{3.1}$$

where

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

This can be rewritten as

$$\begin{aligned} E_D(\mathbf{w}) &= \frac{1}{2}(\mathbf{t}-\boldsymbol{\Phi}\mathbf{w})^T(\mathbf{t}-\boldsymbol{\Phi}\mathbf{w}) \\ &= \frac{1}{2}(\mathbf{t}^T\mathbf{t} - \mathbf{w}^T\boldsymbol{\Phi}^T\mathbf{t} - \mathbf{t}^T\boldsymbol{\Phi}\mathbf{w} + \mathbf{w}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi}\mathbf{w}) \end{aligned}$$

where $\boldsymbol{\Phi}$ is the matrix

$$\boldsymbol{\Phi} = \begin{bmatrix} \boldsymbol{\phi_0(\mathbf{x}_1)} & \cdots & \boldsymbol{\phi_{M-1}(\mathbf{x}_1)} \\ \vdots & \ddots & \vdots \\ \boldsymbol{\phi_0(\mathbf{x}_N)} & \cdots & \boldsymbol{\phi_{M-1}(\mathbf{x}_N)} \end{bmatrix}.$$

Minimizing E_D with respect to $\mathbf{w}$ gives the maximum log-likelihood

$$\nabla_w E_D = -\mathbf{t}^T\boldsymbol{\Phi} + \mathbf{w}^T\boldsymbol{\Phi}^T\boldsymbol{\Phi} = 0.$$

This is easily solved to get

$$\mathbf{w}_{ML} = \boldsymbol{(\Phi^T\Phi)^{-1}\Phi}^T\mathbf{t}.$$

We also find that

$$\frac{1}{\beta_{ML}} = \frac{1}{N}\sum_{n=1}^{N}\left\{t_n - \mathbf{w}_{ML}^T\boldsymbol{\phi}(\mathbf{x}_n)\right\}^2.$$

3.1.2. Regularized least squares. In order to control over-fitting one can add a regularization term so that the total error term becomes,

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

Minimizing the error with respect to $\mathbf{w}$ gives

$$\mathbf{w} = \left(\lambda I + \boldsymbol{\Phi}^T\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^T\mathbf{t}.$$

3.2. Bayesian Linear Regression

Assume the noise precision parameter β to be known. Since the likelihood function is Gaussian, the conjugate prior is also Gaussian

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_0, S_0).$$

The posterior distribution is proportional to the product of the likelihood function and the prior. Since

$$p(\mathbf{w}|\mathbf{t}) \propto p(\mathbf{t}|\mathbf{w})p(\mathbf{w})$$

with

$$p(\mathbf{t}|\mathbf{w}) = \mathcal{N}(\mathbf{t}|\mathbf{\Phi}\mathbf{w}, \beta^{-1}),$$

we can write

$$p(\mathbf{w}|\mathbf{t}) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, S_N)$$

where

$$\begin{aligned} \mathbf{m}_N &= S_N(S_0^{-1}\mathbf{m}_0 + \beta\mathbf{\Phi}^T\mathbf{t}) \\ S_N^{-1} &= S_0^{-1} + \beta\mathbf{\Phi}^T\mathbf{\Phi}. \end{aligned}$$

For simplicity consider the prior

$$p(\mathbf{w}|\alpha) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}I)$$

so that

$$\begin{aligned} \mathbf{m}_N &= \beta S_N\mathbf{\Phi}^T\mathbf{t} \\ S_N^{-1} &= \alpha I + \beta\mathbf{\Phi}^T\mathbf{\Phi}. \end{aligned}$$

This form of the prior simply adds a term to the posterior so that we have from (3.1)

$$\ln p(\mathbf{w}|\mathbf{t}) = -\frac{\beta}{2}\sum_{n=1}^{N}\left\{t_n - \mathbf{w}^T\boldsymbol{\phi}(x_n)\right\}^2 - \frac{\alpha}{2}\mathbf{w}^T\mathbf{w} + \text{const.}$$

Maximizing the posterior distribution with respect to $\mathbf{w}$ is therefore equivalent to the minimization of the sum-of-squares error with a quadratic regularization. Note that we have already obtained the solution that minimizes the regularized error term, namely $\mathbf{m}_N$ above.

EXAMPLE. Let $y(x, \mathbf{w}) = w_0 + w_1 x$. Generate data from $y = -0.3 + 0.5x$ by first choosing x from a uniform distribution and then adding Gaussian noise with $\frac{1}{\sqrt{\beta}} = 0.2$. The prior (over $\mathbf{w}$) has precision matrix $\frac{1}{2.0} I$. The data points are obtained sequentially with the posterior of the previous estimate acting as the prior for the next estimate, illustrated in Figure 3.2.1.

3.2.1. Predictive distribution. In practice we are not interested in the value of $\mathbf{w}$ itself but in predicting t for unseen values of $\mathbf{x}$. Thus we marginalize over $\mathbf{w}$, to get the predictive distribution,

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

where $\alpha$is the precision of the prior over $\mathbf{w}$ and β is the precision of the noise in the data. Note that

$$p(t|\mathbf{w}, \beta) = \mathcal{N}(\mathbf{t}|\mathbf{\Phi}^T \mathbf{w}, \beta^{-1})$$

and

$$p(\mathbf{w}|\mathbf{t}, \alpha, \beta) = \mathcal{N}(\mathbf{w}|\mathbf{m}_N, S_N).$$

Since $p(t|\mathbf{t}, \alpha, \beta)$ is the marginalization of two Gaussians we get

$$p(t|\mathbf{t}, \alpha, \beta) = \mathcal{N}(t|\mathbf{m}_N^T \boldsymbol{\phi}(\mathbf{x}), \sigma_N^2(\mathbf{x}))$$

where the variance is given by

$$\sigma_N^2(\mathbf{x}) = \frac{1}{\beta} + \boldsymbol{\phi}(\mathbf{x})^T S_N \boldsymbol{\phi}(\mathbf{x}).$$

This illustrated in Figure 3.2.2

The green curve shows the synthetic (sinusoidal) data set, using a linear combination of Gaussian basis functions (details not specified). This Figure shows the mean of the predictive distribution in red for $N = 1$, $N = 2$, $N = 4$ and $N = 24$ data points. The mean is of course the best estimate that we have for the given number of data points, at any given value of x. We also see (shaded) the standard deviation around the mean as a function of the input value x. We can also look at the variations in the predictive function. In order to do this we draw samples of $\mathbf{w}$ from its posterior distribution (for the given data points). For each sample we then draw the predictive curve $y(x, \mathbf{w})$. This is shown in Figure 3.2.3.

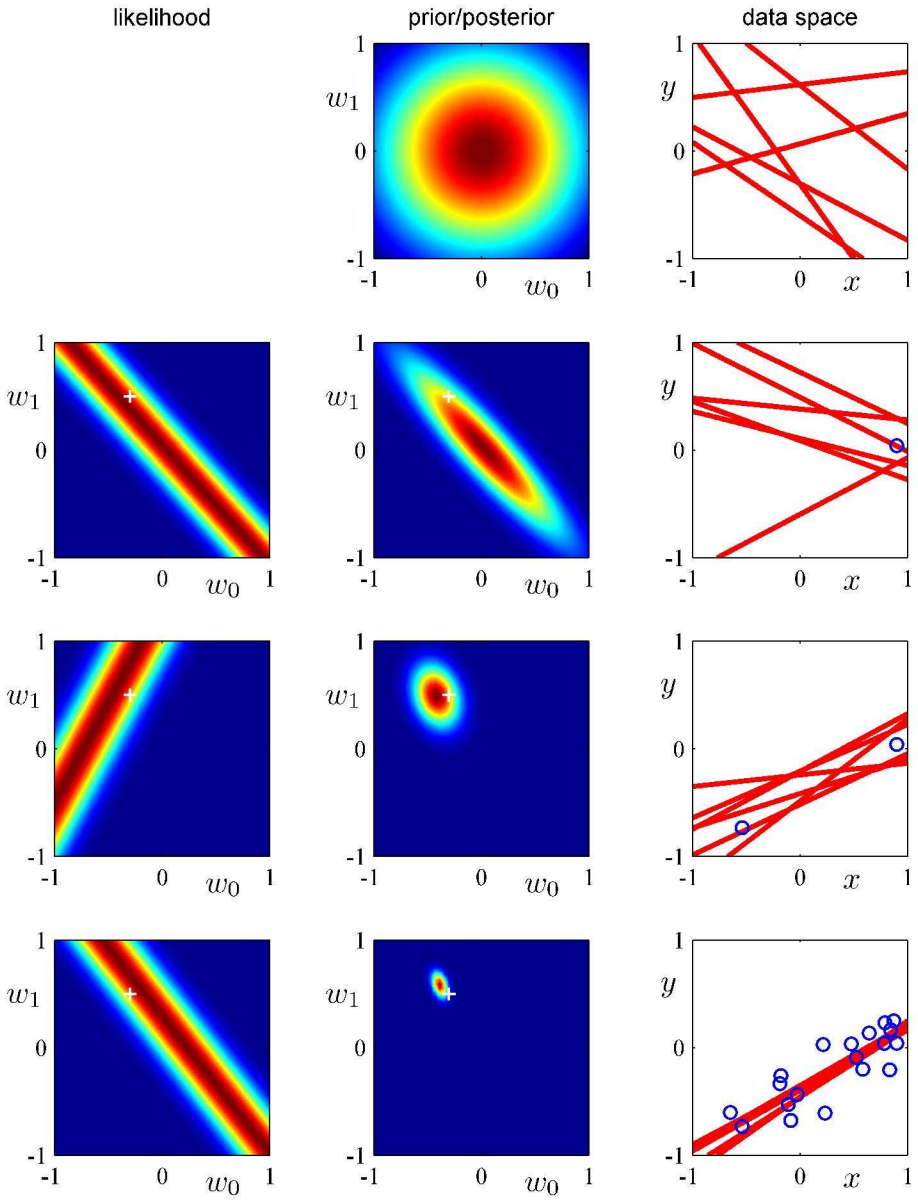


FIGURE 3.2.1. Sequential Bayesian learning.

3.2.2. Equivalent kernel. There is a powerful alternative interpretation of the posterior mean. Writing the posterior mean as $\mathbf{m}_N$ we get

$$\begin{aligned} y(\mathbf{x}, \mathbf{m}_N) &= \mathbf{m}_N^T \boldsymbol{\phi}(\mathbf{x}) \\ &= \beta \boldsymbol{\phi}(\mathbf{x})^T S_N \boldsymbol{\Phi}^T \mathbf{t} \\ &= \sum_{n=1}^{N} \beta \boldsymbol{\phi}(\mathbf{x})^T S_N \boldsymbol{\phi}(\mathbf{x}_n) t_n. \end{aligned}$$

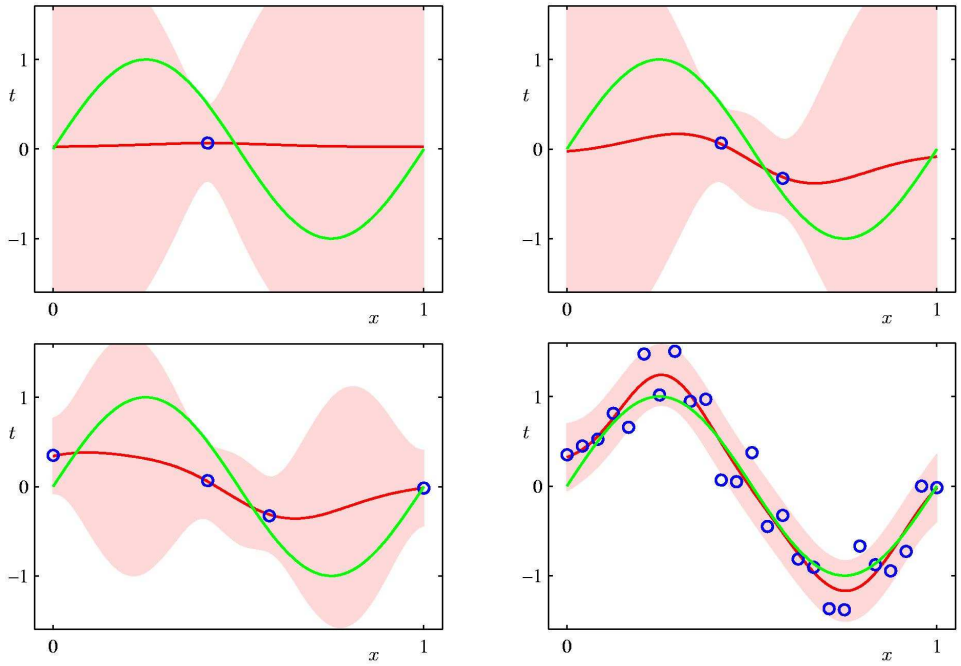


FIGURE 3.2.2. Predictive distribution.

Defining the kernel

$$k(\mathbf{x}, \mathbf{x}_n) = \beta \boldsymbol{\phi}(\mathbf{x})^T S_N \boldsymbol{\phi}(\mathbf{x}_n)$$

we get

$$y(\mathbf{x}, \mathbf{m}_N) = \sum_{n=1}^{N} k(\mathbf{x}, \mathbf{x}_n) t_n.$$

We can think of the kernel as a smoother of the data t_n. Plotting the kernel as in Figure 3.2.4 we see that it gives more weight to the data closest to $\mathbf{x}$ where we want to estimate the value of t.

This provides a general procedure: In general one has to find an appropriate kernel.

Note that

$$\sum_{n=1}^{N} k(\mathbf{x}, \mathbf{x}_n) = 1.$$

This can be seen intuitively as follows. Let us say that all out input values $t_n = 1$, then we need the prediction to equal 1 as well for all input values $\mathbf{x}$.

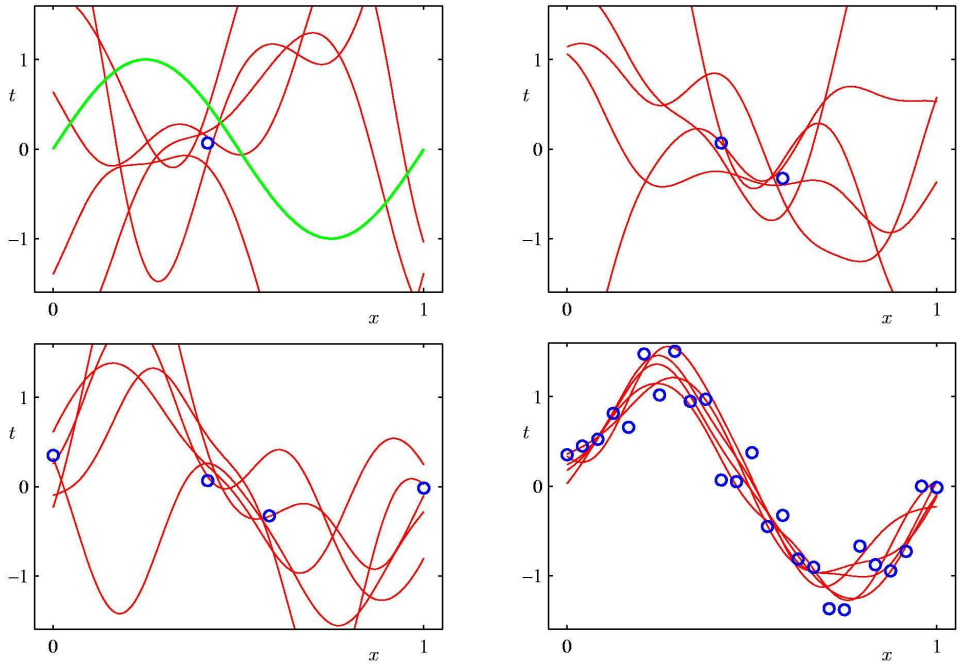


FIGURE 3.2.3. Samples from the posterior distribution of $\mathbf{w}$.

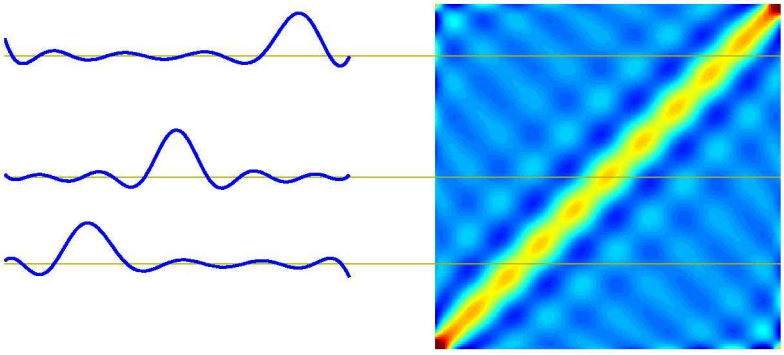


FIGURE 3.2.4. Linear smoother.

3.3. Bayesian Model Comparison

Given observed data $\mathcal{D}$ we consider L different models $\mathcal{M}_i$, $i = 1, \ldots, L$, where each model is a probability observation over the data $\mathcal{D}$. The idea is to select the best model. Our uncertainty in the exact nature of the model is expressed in the prior

probability, $p(\mathcal{M}_i)$. Given a training set we are looking for the posterior distribution

$$p(\mathcal{M}_i|\mathcal{D}) \propto p(\mathcal{D}|\mathcal{M}_i)p(\mathcal{M}_i).$$

If we assume equal prior probabilities the interesting term is the model evidence $p(\mathcal{D}|\mathcal{M}_i)$ which describes the preference shown by the data for different models. Once we know the posterior distribution of the model, the predictive distribution is obtained from marginalizing over the different models,

$$\begin{aligned} p(t|\mathbf{x},\mathcal{D}) &= \sum_{i=1}^{L} p(t,\mathcal{M}_i|\mathbf{x},\mathcal{D}) \\ &= \sum_{i=1}^{L} p(t|\mathbf{x},\mathcal{M}_i,\mathcal{D})p(\mathcal{M}_i|\mathbf{x},\mathcal{D}) \\ &= \sum_{i=1}^{L} p(t|\mathbf{x},\mathcal{M}_i,\mathcal{D})p(\mathcal{M}_i|\mathcal{D}), \end{aligned}$$

since the model only depends on the data.

Note that this is an example of how one can combine different models into a single predictive model, hopefully more accurate than any of the individual models. An approximation to this model averaging is to choose the single most probable model alone to make predictions. This is known as model selection.

If the model is governed by a set of parameters $\mathbf{w}$, as we have used up to now, the model evidence is obtained by marginalizing,

$$\begin{aligned} p(\mathcal{D}|\mathcal{M}_i) &= \int p(\mathcal{D},\mathbf{w}|\mathcal{M}_i)d\mathbf{w} \\ &= \int p(\mathcal{D}|\mathbf{w},\mathcal{M}_i)p(\mathbf{w}|\mathcal{M}_i)d\mathbf{w}. \end{aligned}$$

We can try and understand the meaning by making a simple approximation. Assuming a single parameter w, then the posterior distribution for a given model (omitted in the notation) is given by

$$p(w|\mathcal{D}) \propto p(\mathcal{D}|w)p(w).$$

Assuming that the posterior distribution is sharply peaked around w_{MAP} with width $\Delta w_{\mathrm{posterior}}$we approximate the integral by the maximum value of the integrand

times the width of the integrand. Also assume that the prior is flat with width Δw_{prior} so that $p(w) = \frac{1}{\Delta w_{\text{prior}}}$, we find that

$$\begin{aligned} p(\mathcal{D}) &= \int p(\mathcal{D}|w)p(w)dw \\ &\approx p(\mathcal{D}|w_{\text{MAP}})\frac{\Delta w_{\text{posterior}}}{\Delta w_{\text{prior}}}. \end{aligned}$$

Taking logs

$$\ln p(\mathcal{D}) = \ln p(\mathcal{D}|w_{\text{MAP}}) + \ln\left(\frac{\Delta w_{\text{posterior}}}{\Delta w_{\text{prior}}}\right).$$

This approximation is illustrated in Figure 3.3.1.

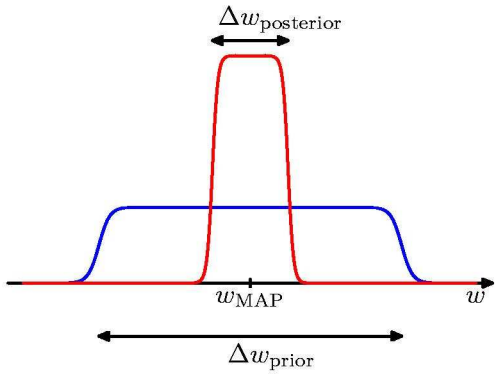


FIGURE 3.3.1. Approximation of the model evidence.

The first term represents the fit to the data given by the most probable parameter values. The second term penalizes the model according to its complexity. Since $\Delta w_{\text{posterior}} < \Delta w_{\text{prior}}$, it is negative and increases in magnitude as the ratio $\frac{\Delta w_{\text{posterior}}}{\Delta w_{\text{prior}}}$ gets smaller. Thus if the parameters are finely tuned to the data in the posterior distribution, the penalty term is large.

If we have M parameters, and assuming that all parameters have the same ratio $\frac{\Delta w_{\text{posterior}}}{\Delta w_{\text{prior}}}$, we get

$$\ln p(\mathcal{D}) = \ln p(\mathcal{D}|w_{\text{MAP}}) + M \ln\left(\frac{\Delta w_{\text{posterior}}}{\Delta w_{\text{prior}}}\right).$$

Thus the size of the penalty increases linearly with the number of adaptive parameters in the model. If we increase the number of parameters in the model the first term should decrease since we are better able to fit the data. The penalty term on the other hand increases. We need to find the trade-off between these two competing terms.

3.4. The Evidence Approximation

In a full Bayesian treatment the predictive distribution is obtained from a full marginalization,

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

where we can think of β as the precision of the noise in the data, and α as the precision of the prior over the model parameters $\mathbf{w}$. These integrals are analytically intractable (even if we assume Gaussian distributions) and one has to make various approximations. If the posterior $p(\alpha,\beta|\mathbf{t})$ is sharply peaked around $\widehat{\alpha}$ and $\widehat{\beta}$,then the integral becomes

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

where we need to find the values of $\widehat{\alpha}$ and $\widehat{\beta}$. From Bayes' theorem follows

$$p(\alpha,\beta|\mathbf{t}) \propto p(\mathbf{t}|\alpha,\beta)p(\alpha,\beta).$$

If the value of the prior is relatively flat, the values of $\widehat{\alpha}$ and $\widehat{\beta}$ are obtained by maximizing the marginal likelihood function $p(\mathbf{t}|\alpha,\beta)$.

3.4.1. Evaluation of the evidence function. The marginal likelihood function is obtained from

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

This is evaluated by completing the square

$$p(\mathbf{t}|\alpha,\beta) = \left(\frac{\beta}{2\pi}\right)^{N/2}\left(\frac{\alpha}{2\pi}\right)^{M/2}\int \exp\left\{-E(\mathbf{w})\right\}d\mathbf{w}$$

where

$$\begin{aligned} E(\mathbf{w}) &= \beta E_D(\mathbf{w}) + \alpha E_W(\mathbf{w}) \\ &= \frac{\beta}{2}\left\|\mathbf{t} - \mathbf{\Phi}\mathbf{w}\right\|^2 + \frac{\alpha}{2}\mathbf{w}^T\mathbf{w}. \end{aligned}$$

This is the regularized sum-of-squares error. Completing the square over $\mathbf{w}$ gives

$$E(\mathbf{w}) = E(\mathbf{m}_N) + \frac{1}{2}(\mathbf{w} - \mathbf{m}_N)^T A(\mathbf{w} - \mathbf{m}_N)$$

where

$$A = \alpha I + \beta \mathbf{\Phi}^{\boldsymbol{T}}\mathbf{\Phi}$$

and

$$E(\mathbf{m}_N) = \frac{\beta}{2}\left\|\mathbf{t} - \mathbf{\Phi}\mathbf{m}_N\right\|^2 + \frac{\alpha}{2}\mathbf{m}_N^T\mathbf{m}_N$$

with

$$\mathbf{m}_N = \beta A^{-1}\mathbf{\Phi}^T\mathbf{t}.$$

Since $A = S_N^{-1}$, $\mathbf{m}_N$ represents the mean of the posterior distribution. It now follows easily that

$$\begin{aligned} \int \exp\left\{-E(\mathbf{w})\right\} d\mathbf{w} &= \exp\left\{-E(\mathbf{m}_N)\right\} \int \exp\left\{-\frac{1}{2}(\mathbf{w} - \mathbf{m}_N)^T A(\mathbf{w} - \mathbf{m}_N)\right\} d\mathbf{w} \\ &= \exp\left\{-E(\mathbf{m}_N)\right\} (2\pi)^{M/2} \left|A\right|^{-1/2}. \end{aligned}$$

The log of the marginal likelihood there becomes

$$\begin{aligned} \ln p(\mathbf{t}|\alpha,\beta) &= \ln\left[\left(\frac{\beta}{2\pi}\right)^{N/2}\left(\frac{\alpha}{2\pi}\right)^{M/2}\int \exp\left\{-E(\mathbf{w})\right\} d\mathbf{w}\right] \\ &= \frac{M}{2}\ln\alpha + \frac{N}{2}\ln\beta - E(\mathbf{m}_N) - \frac{1}{2}\ln\left|A\right| - \frac{N}{2}\ln(2\pi). \end{aligned}$$

The model evidence function for polynomial regression is shown in Figure 3.4.1.

The best value is obtained for $M = 3$.

3.4.2. Maximizing the evidence function. Maximize $p(\mathbf{t}|\alpha,\beta)$ with respect to α. First diagonalize A by defining the eigenfunctions

$$\left(\beta\mathbf{\Phi}^T\mathbf{\Phi}\right)\mathbf{u}_i = \lambda_i\mathbf{u}_i.$$

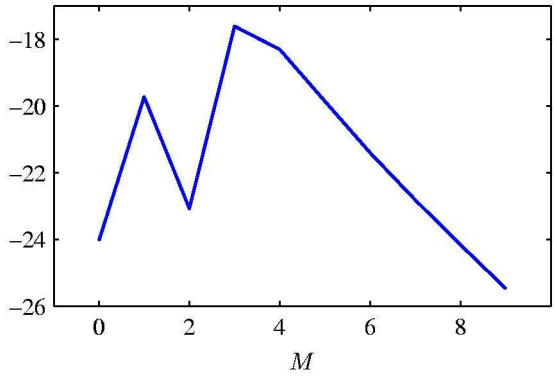


FIGURE 3.4.1. Model evidence for polynomial regression.

Then A has the eigenvalues $\alpha + \lambda_i$. We have

$$\begin{aligned} \frac{d}{d\alpha} \ln |A| &= \frac{d}{d\alpha} \ln \prod_i (\lambda_i + \alpha) \\ &= \sum_i \frac{1}{\lambda_i + \alpha}. \end{aligned}$$

Thus $\frac{d}{d\alpha} \ln p(\mathbf{t}|\alpha, \beta) = 0$ becomes

$$0 = \frac{M}{2\alpha} - \frac{1}{2}\mathbf{m}_N^T \mathbf{m}_N - \frac{1}{2} \sum_i \frac{1}{\lambda_i + \alpha}$$

or

$$\alpha \mathbf{m}_N^T \mathbf{m}_N = M - \alpha \sum_{i=1}^{M} \frac{1}{\lambda_i + \alpha} = \gamma.$$

Rewriting we get

$$\begin{aligned} \gamma &= \sum_{i=1}^{M} \left[1 - \frac{\alpha}{\lambda_i + \alpha} \right] \\ &= \sum_{i=1}^{M} \left[\frac{\lambda_i + \alpha}{\lambda_i + \alpha} - \frac{\alpha}{\lambda_i + \alpha} \right] \\ &= \sum_{i=1}^{M} \frac{\lambda_i}{\lambda_i + \alpha}, \end{aligned}$$

and the marginal likelihood is maximized by

$$\alpha = \frac{\gamma}{\mathbf{m}_N^T \mathbf{m}_N}.$$

Note:

- This is an implicit function that needs to be solved by iteration, e.g. fixed point iteration.
- α is determined purely by using the training data, no independent data set is required for optimizing the model complexity.

Do the same for β. Since the eigenvalues λ_i are proportional to β it follows that

$$\frac{d\lambda_i}{d\beta} = \frac{\lambda_i}{\beta}$$

so that

$$\begin{aligned} \frac{d}{d\beta} \ln |A| &= \frac{d}{d\beta} \sum_i \ln (\lambda_i + \alpha) \\ &= \frac{1}{\beta} \sum_i \frac{\lambda_i}{\lambda_i + \alpha} \\ &= \frac{\gamma}{\beta}. \end{aligned}$$

Thus $\frac{d}{d\alpha} \ln p(\mathbf{t}|\alpha, \beta) = 0$ becomes

$$0 = \frac{N}{2\beta} - \frac{1}{2} \sum_{n=1}^{N} \left\{ t_n - \mathbf{m}_N^T \boldsymbol{\phi}(\mathbf{x}_n) \right\}^2 - \frac{\gamma}{2\beta}$$

so that

$$\frac{1}{\beta} = \frac{1}{N - \gamma} \sum_{n=1}^{N} \left\{ t_n - \mathbf{m}_N^T \boldsymbol{\phi}(\mathbf{x}_n) \right\}^2.$$

Again β has to be calculated using an iterative procedure. Note that one can solve for both α and β.

3.5. Summary

(1) Want to build a probabilistic model

$$p(t|\mathbf{w}, \beta) = \mathcal{N}(t|\mathbf{w}^T \boldsymbol{\phi}(\mathbf{x}), \beta^{-1})$$

by estimating the parameters $\mathbf{w}$ and β from data $\left\{ \begin{array}{cc} \mathbf{x}_n & t_n \end{array} \right\}$, $n = 1, \ldots, N$.

(2) Calculating the parameters using least squares is the same as maximizing the log-likelihood

$$\ln p(\mathbf{t}|\mathbf{w}, \beta).$$

(3) Maximizing the posterior (assuming β to be known), $p(\mathbf{w}|\mathbf{t}) \propto p(\mathbf{t}|\mathbf{w})p(\mathbf{w})$ requires the prior distribution $p(\mathbf{w})$. Assuming $p(\mathbf{w}) = \mathcal{N}(\mathbf{w}|\mathbf{0}, \alpha^{-1}I)$, introduces a hyper parameter α. Maximizing the posterior is the same as least squares with a regularization term.

(4) Since $\mathbf{w}$ is now a full-fledged random variable, one can adopt a full Bayesian treatment and marginalizes over it.

(5) Similarly the hyper-parameters α and β can also be considered to be random variables in which case one can get rid of them by marginalizing over them. Alternatively one can go for a point-wise estimate of the hyper-parameters using the data. A point-wise estimate is obtained from maximizing

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

This is knows as the evidence approximation.

CHAPTER 4

Linear Models for Classification.

4.1. Discriminant Functions

Need to assign an input vector $\mathbf{x}$ to one of K classes, $\mathcal{C}_k$.

4.1.1. Two Classes. A simple example of a linear discriminant function is a linear function of the input vector

$$y(\mathbf{x}) = \mathbf{w}^T\mathbf{x} + w_0$$

where $\mathbf{w}$ is a weight vector, and w_0 a bias. $\mathbf{x}$ is assigned to class $\mathcal{C}_1$ if $y(\mathbf{x}) \geq 0$ and to $\mathcal{C}_2$ otherwise. The decision boundary is therefore give by $y(\mathbf{x}) = 0$, which is a $D-1$ dimensional hyperplane in the D dimensional input space. If $\mathbf{x}_A$ and $\mathbf{x}_B$ both lie on the decision boundary then $\mathbf{w}^T(\mathbf{x}_A - \mathbf{x}_B) = 0$ so that $\mathbf{w}$ is orthogonal to the decision boundary.

Let $\mathbf{x}_0 = \alpha\frac{\mathbf{w}}{\|\mathbf{w}\|}$ be the point on the decision surface closest to the origin, then α is the distance of the decision boundary from the origin. Since $y(\mathbf{x}_0) = 0$, it follows that $\alpha = -\frac{w_0}{\|\mathbf{w}\|}$. The bias determines the location of the decision surface.

From Figure 4.1.1 we note that $y(\mathbf{x})$ gives a signed measure of the perpendicular distance r of $\mathbf{x}$ from the decision surface. First write

$$\mathbf{x} = \mathbf{x}_\perp + r\frac{\mathbf{w}}{\|\mathbf{w}\|}.$$

Since $y(\mathbf{x}_\perp) = 0$, i.e. $\mathbf{w}^T\mathbf{x}_\perp + w_0 = 0$ it follows that

$$\begin{aligned} y(\mathbf{x}) &= y(\mathbf{x}_\perp + r\frac{\mathbf{w}}{\|\mathbf{w}\|}) \\ &= \mathbf{w}^T\mathbf{x}_\perp + r\frac{\mathbf{w}^T\mathbf{w}}{\|\mathbf{w}\|} + w_0 \\ &= r\,\|\mathbf{w}\|\,. \end{aligned}$$

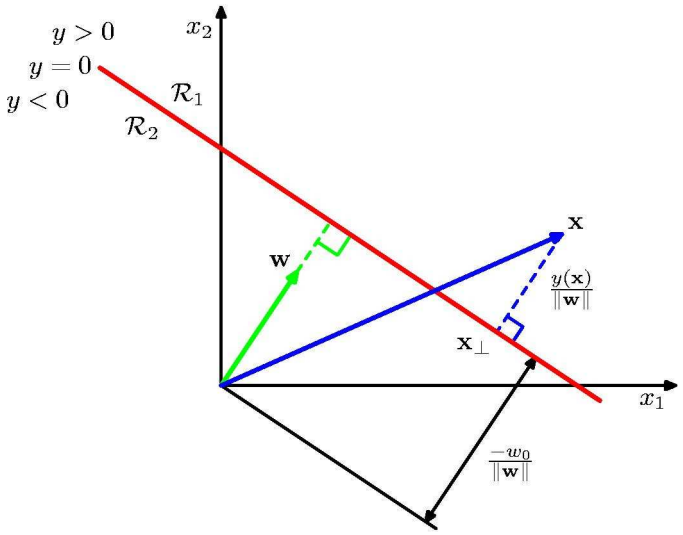


FIGURE 4.1.1. Geometry of the linear discriminant function.

so that

$$r = \frac{y(\mathbf{x})}{\|\mathbf{w}\|}.$$

Note that the linear classifier can also be written as

$$y(\mathbf{x}) = \widetilde{\mathbf{w}}^T \widetilde{\mathbf{x}}.$$

4.1.2. Multiple classes. It is not feasible to build an $K-$class system from a number of two-class systems. Better to use a single K-class system based on K linear functions of the form

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}.$$

A point $\mathbf{x}$ is then assigned to class $\mathcal{C}_k$ if $y_k(\mathbf{x}) > y_j(\mathbf{x})$ for all $j \neq k$. The decision boundary between $\mathcal{C}_k$ and $\mathcal{C}_j$ is therefore given by $y_k(\mathbf{x}) = y_j(\mathbf{x})$ and corresponds to a hyperplane defined by

$$(\mathbf{w}_k - \mathbf{w}_j)^T \mathbf{x} + (w_{k0} - w_{j0}) = 0.$$

Same form as the decision surface for the two-class problem.

It is important to note that the decision regions of such a discriminant are always singly connected and convex. Let $\mathbf{x}_A$ and $\mathbf{x}_B$ both lie inside decision region $\mathcal{R}_k$ as shown in Figure 4.1.2.

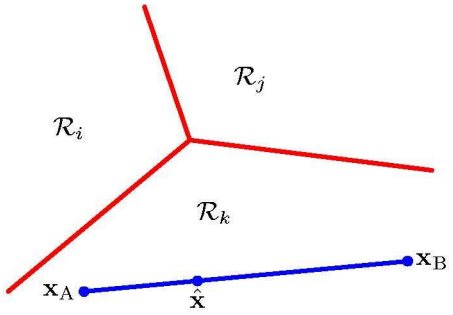


FIGURE 4.1.2. Decision regions for a multi-class linear discriminant function.

Any point $\widehat{\mathbf{x}}$ on the line connecting $\mathbf{x}_A$ and $\mathbf{x}_B$ satisfies

$$\widehat{\mathbf{x}} = \lambda \mathbf{x}_A + (1 - \lambda)\mathbf{x}_B$$

where $0 \leq \lambda \leq 1$. Then

$$y_k(\widehat{\mathbf{x}}) = \lambda y_k(\mathbf{x}_A) + (1 - \lambda) y_k(\mathbf{x}_B).$$

Since $y_k(\mathbf{x}_A) > y_j(\mathbf{x}_A)\ \forall j \neq k$ and $y_k(\mathbf{x}_B) > y_j(\mathbf{x}_B)\ \forall j \neq k$ it follows that $y_k(\widehat{\mathbf{x}}) > y_j(\widehat{\mathbf{x}})\ \forall j \neq k$. This means that $\widehat{\mathbf{x}}$ also lies inside $\mathcal{R}_k$. Thus $\mathcal{R}_k$ is singly connected and convex.

We still need to know how we learn the parameters $\mathbf{w}$.

4.1.3. Least squares for classification. Each class $\mathcal{C}_k$ is described by its own linear model

$$y_k(\mathbf{x}) = \mathbf{w}_k^T \mathbf{x} + w_{k0}$$

written as

$$\mathbf{y}(\mathbf{x}) = \widetilde{W}^T \widetilde{\mathbf{x}}$$

where

$$\widetilde{W} = \begin{bmatrix} w_{00} & \cdots & w_{K0} \\ \vdots & \ddots & \vdots \\ w_{0D} & \cdots & w_{KD} \end{bmatrix}$$

and $\widetilde{\mathbf{x}} = \begin{bmatrix} 1 & \mathbf{x}^T \end{bmatrix}^T$. An unseen $\mathbf{x}$ is assigned to the class for which the output $y_k = \widetilde{\mathbf{w}}_k^T \widetilde{\mathbf{x}}$ is the largest and we determine the coefficients $\widetilde{W}$ using a least squares technique. We use a 1-of-K coding scheme for the target vector which is far from Gaussian, the least-squares approach is therefore inefficient. It is, among other things, sensitive to outliers. Nevertheless, let us see how we can obtain the $\mathbf{w}_k$s through least squares. It mounts to an exercise in linear algebra.

We assume that we have a fully observable system, i.e. we observe a number of features $\mathbf{x}_n$, $n = 1, \ldots, N$ together with its class label, given by $\mathbf{t}_n$ where $\mathbf{t}_n$ is a K-dimensional vector such that its j-th component is 1if $\mathbf{x}_n$ belongs to class j, and the rest of the components equal zero. Accordingly we observe $\{\mathbf{x}_n, \mathbf{t}_n\}$ $n = 1, \ldots, N$. The trick is to write the equations in matrix form. First we write

$$\begin{bmatrix} y_1(\mathbf{x}_n) \\ \vdots \\ y_K(\mathbf{x}_n) \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{x}}_n^T \widetilde{\mathbf{w}}_1 \\ \vdots \\ \widetilde{\mathbf{x}}_n^T \widetilde{\mathbf{w}}_K \end{bmatrix}, \; n = 1, \ldots, N.$$

If we collect all these equations into a single equation, we get

$$\begin{bmatrix} y_1(\mathbf{x}_1) & \cdots & y_K(\mathbf{x}_1) \\ \vdots & \ddots & \vdots \\ y_1(\mathbf{x}_N) & \cdots & y_K(\mathbf{x}_N) \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{x}}_1^T \widetilde{\mathbf{w}}_1 & \cdots & \widetilde{\mathbf{x}}_1^T \widetilde{\mathbf{w}}_K \\ \vdots & \ddots & \vdots \\ \widetilde{\mathbf{x}}_N^T \widetilde{\mathbf{w}}_1 & \cdots & \widetilde{\mathbf{x}}_N^T \widetilde{\mathbf{w}}_K \end{bmatrix} = \begin{bmatrix} \widetilde{\mathbf{x}}_1^T \\ \vdots \\ \widetilde{\mathbf{x}}_N^T \end{bmatrix} \begin{bmatrix} \widetilde{\mathbf{w}}_1 & \cdots & \widetilde{\mathbf{w}}_K \end{bmatrix}.$$

This is rewritten as

$$Y = \widetilde{X}\widetilde{W}$$

where the meanings of $\widetilde{W}$ and $\widetilde{X}$ derive from the previous expression. The n-th column of Y has to be matched to $\mathbf{t}_n$. If we now define a matrix $T = \begin{bmatrix} \mathbf{t}_1 & \cdots & \mathbf{t}_N \end{bmatrix}$ we need to find $\widetilde{W}$ that minimizes the Frobenius norm $\|Y - T\|_F$, i.e. we need to find the least squares solution. Let us now stack the columns of T and $\widetilde{W}$ to obtain a system of the form

$$\mathbf{t} = A\widetilde{\mathbf{w}}$$

where $\mathbf{t}$ and $\mathbf{w}$ consist of the columns of Y and $\widetilde{W}$ respectively, and A is a block-diagonal matrix where the diagonal blocks $\widetilde{X}$. The least squares solution is then

given by $\widetilde{\mathbf{w}} = \left(A^T A\right)^{-1} A^T \mathbf{t}$. If we rearrange we get

$$\widetilde{W} = \left(\widetilde{X}^T \widetilde{X}\right)^{-1} \widetilde{X}^T T^T.$$

Note: As it is written this system is numerically unstable—it is not a good idea to solve the *normal equations* directly. A numerical stable scheme is to first do a QR factorization of A and then solve the normal equations.

4.1.4. Fisher's linear discriminant. View the classification problem as one of dimensionality reduction. First consider only two classes. Given a D-dimensional vector $\mathbf{x}$, project it down to one dimension using

$$y = \mathbf{w}^T \mathbf{x}.$$

Placing a threshold on y and classifying $y \geq -w_0$ as class $\mathcal{C}_1$, and otherwise of class $\mathcal{C}_2$, we recover the simple classifier. The idea is to choose $\mathbf{w}$ in such a way that maximum separation is possible in one dimension. Calculate the means of the two classes,

$$\mathbf{m}_1 = \frac{1}{N_1} \sum_{n \in \mathcal{C}_1} \mathbf{x}_n, \quad \mathbf{m}_2 = \frac{1}{N_2} \sum_{n \in \mathcal{C}_2} \mathbf{x}_n.$$

We want to maximally separate the projected class means, i.e. we want to choose $\mathbf{w}$ so as to maximize

$$m_2 - m_1 = \mathbf{w}^T (\mathbf{m}_2 - \mathbf{m}_1),$$

where $m_k = \mathbf{w}^T \mathbf{m}_k$. Constraining $\mathbf{w}$ to $\mathbf{w}^T \mathbf{w} = 1$, by using a Lagrange multiplier, we minimize

$$L(\mathbf{w}) = \mathbf{w}^T (\mathbf{m}_2 - \mathbf{m}_1) + \lambda (\mathbf{w}^T \mathbf{w} - 1)$$

subject to the constraint. From $\frac{\partial L}{\partial w_n} = 0$ follows

$$(m_{2n} - m_{1n}) + 2\lambda w_n = 0$$

subject to $\mathbf{w}^T \mathbf{w} = 1$. This gives

$$\mathbf{w} \propto \mathbf{m}_2 - \mathbf{m}_1.$$

This means that $\mathbf{w}$ points in the direction of the vector connecting the two class means, and that the decision boundary is orthogonal to this direction. The situation is illustrated in Figure 4.1.3.

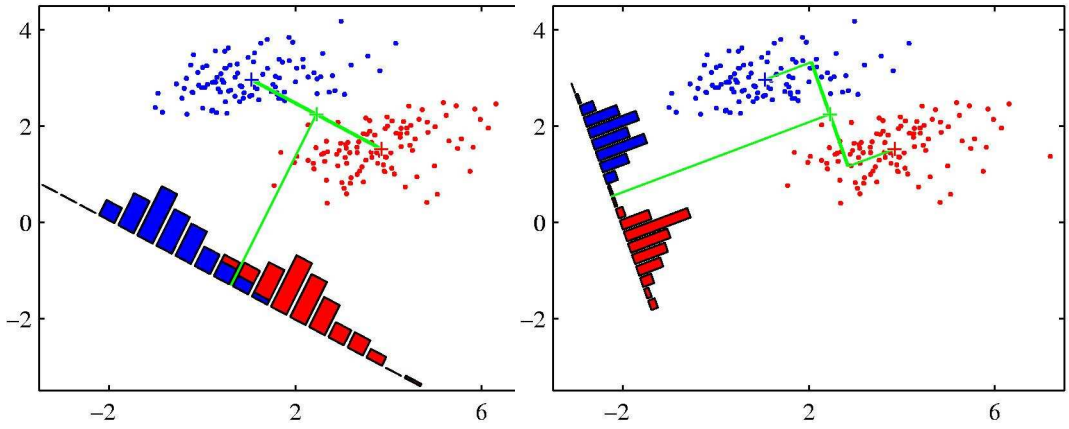


FIGURE 4.1.3. Projection onto the line joining the class means.

The Figure also shows that one can do better. We want to minimize class overlap by also minimizing the projected class scatter, given by

$$s_k^2 = \sum_{n \in \mathcal{C}_k} (y_n - m_k)^2$$

where $y_n = \mathbf{w}^T \mathbf{x}_k$. The total within-class scatter is $s_1^2 + s_2^2$. The Fisher criterion is to maximize $J(\mathbf{w})$ where

$$\begin{aligned} J(\mathbf{w}) &= \frac{(m_2 - m_1)^2}{s_1^2 + s_2^2}. \\ &= \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}, \end{aligned}$$

$$S_B = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T$$

and

$$S_W = \sum_{n \in \mathcal{C}_1} (\mathbf{x}_n - \mathbf{m}_1)(\mathbf{x}_n - \mathbf{m}_1)^T + \sum_{n \in \mathcal{C}_2} (\mathbf{x}_n - \mathbf{m}_2)(\mathbf{x}_n - \mathbf{m}_2)^T.$$

$J(\mathbf{w})$ is maximized when

$$(\mathbf{w}^T S_B \mathbf{w}) S_W \mathbf{w} = (\mathbf{w}^T S_W \mathbf{w}) S_B \mathbf{w}.$$

Note that since we are only interested in the direction of $\mathbf{w}$ not its magnitude, we can drop the scalar factors. Also $S_B \mathbf{w} = (\mathbf{m}_2 - \mathbf{m}_1)(\mathbf{m}_2 - \mathbf{m}_1)^T \mathbf{w}$ points in the

direction of $\mathbf{m}_2 - \mathbf{m}_1$ and we get that

$$\mathbf{w} \propto S_W^{-1}(\mathbf{m}_2 - \mathbf{m}_1).$$

We can use the projected data to construct an optimal classifier as described in the first Chapter.

4.1.5. Fisher's discriminant for multiple classes. Here one really has to think of Fischer's scheme as a way to do dimensionality reduction, but dimensionality reduction with a difference. We want to find the projection onto a lower dimensional space optimized for classification, i.e. maximal separation between the classes. We are going to discuss dimensionality reduction in much more detail a little later and there we will also see how one can reduce the dimensionality of the data space while preserving the geometric structure of the space. For now we concentrate on classification.

Assume that the dimensionality D is greater than the number of classes K. Introduce D' linear features $y_k = \mathbf{w}_k^T \mathbf{x}$ written in matrix form as

$$\mathbf{y} = W^T \mathbf{x}$$

where

$$W = \left[\begin{array}{ccc} \mathbf{w}_1 & \cdots & \mathbf{w}_{D'} \end{array} \right].$$

The D dimensional features $\mathbf{x}$ are therefore projected onto a lower dimensional space, D'. Our task is to find the lower dimensional space with maximal class separation.

The within-class scatter is defined by

$$S_W = \sum_{k=1}^{K} S_k$$

where

$$\begin{aligned} S_k &= \sum_{n \in \mathcal{C}_k} (\mathbf{x}_n - \mathbf{m}_k)(\mathbf{x}_n - \mathbf{m}_k)^T \\ \mathbf{m}_k &= \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{x}_n \end{aligned}$$

where N_k is the number of patterns in class $\mathcal{C}_k$. The generalization of the between-class scatter matrix is derived from the total scatter

$$S_T = \sum_{n=1}^{N} (\mathbf{x}_n - \mathbf{m})(\mathbf{x}_n - \mathbf{m})^T$$

where $\mathbf{m}$ is the mean of the total data set,

$$\mathbf{m} = \frac{1}{N} \sum_{n=1}^{N} \mathbf{x}_n = \frac{1}{N} \sum_{n=1}^{K} N_k \mathbf{m}_k. \tag{4.1}$$

The total scatter is written as the sum of within-class scatter matrix and an additional matrix S_B which is identified as a measure of the between-class scatter,

$$S_t = S_w + S_b$$

where

$$S_b = \sum_{k=1}^{K} N_k (\mathbf{m}_k - \mathbf{m})(\mathbf{m}_k - \mathbf{m})^T.$$

To see this, write

$$\begin{aligned}
S_t &= \sum_{n=1}^{N}(\mathbf{x}_n-\mathbf{m})(\mathbf{x}_n-\mathbf{m})^T \\
&= \sum_{k=1}^{K}\sum_{n\in\mathcal{C}_k}(\mathbf{x}_n-\mathbf{m})(\mathbf{x}_n-\mathbf{m})^T \\
&= \sum_{k=1}^{K}\sum_{n\in\mathcal{C}_k}(\mathbf{x}_n-\mathbf{m}_k+\mathbf{m}_k-\mathbf{m})(\mathbf{x}_n-\mathbf{m}_k+\mathbf{m}_k-\mathbf{m})^T \\
&= \sum_{k=1}^{K}\sum_{n\in\mathcal{C}_k}\left[(\mathbf{x}_n-\mathbf{m}_k)(\mathbf{x}_n-\mathbf{m}_k)^T+(\mathbf{m}_k-\mathbf{m})(\mathbf{m}_k-\mathbf{m})^T\right] \\
&\quad +\sum_{k=1}^{K}\sum_{n\in\mathcal{C}_k}\left[(\mathbf{x}_n-\mathbf{m}_k)(\mathbf{m}_k-\mathbf{m})^T+(\mathbf{m}_k-\mathbf{m})(\mathbf{x}_n-\mathbf{m}_k)^T\right] \\
&= \sum_{k=1}^{K}\sum_{n\in\mathcal{C}_k}(\mathbf{x}_n-\mathbf{m}_k)(\mathbf{x}_n-\mathbf{m}_k)^T+\sum_{k=1}^{K}\sum_{n\in\mathcal{C}_k}(\mathbf{m}_k-\mathbf{m})(\mathbf{m}_k-\mathbf{m})^T \\
&= S_w+\sum_{k=1}^{K}N_k(\mathbf{m}_k-\mathbf{m})(\mathbf{m}_k-\mathbf{m})^T,
\end{aligned}$$

where we noted that

$$\begin{aligned}
\sum_{n\in\mathcal{C}_k}\left[(\mathbf{x}_n-\mathbf{m}_k)(\mathbf{m}_k-\mathbf{m})^T+(\mathbf{m}_k-\mathbf{m})(\mathbf{x}_n-\mathbf{m}_k)^T\right] &= \left[\sum_{n\in\mathcal{C}_k}(\mathbf{x}_n-\mathbf{m}_k)\right](\mathbf{m}_k-\mathbf{m})^T+ \\
&\quad +(\mathbf{m}_k-\mathbf{m})\left[\sum_{n\in\mathcal{C}_k}(\mathbf{x}_n-\mathbf{m}_k)^T\right] \\
&= 0.
\end{aligned}$$

We now define similar matrices on the D'-dimensional linear, projected spaces (using $\mathbf{y}=W^T\mathbf{x}$),

$$s_w=\sum_{k=1}^{K}\sum_{n\in\mathcal{C}_k}(\mathbf{y}_n-\boldsymbol{\mu}_k)(\mathbf{y}_n-\boldsymbol{\mu}_k)^T$$

and

$$s_b = \sum_{nk=1}^{K} N_k(\boldsymbol{\mu}_k - \boldsymbol{\mu})(\boldsymbol{\mu}_k - \boldsymbol{\mu})^T$$

where

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n \in \mathcal{C}_k} \mathbf{y}_n, \quad \boldsymbol{\mu} = \frac{1}{N} \sum_{k=1}^{K} N_k \boldsymbol{\mu}_k.$$

The main problem is to formulate criteria for class separability. One possibility is to maximize the ration between the between-class scatter and within-class scatter, the appropriate quantity to look at is $s_w^{-1} s_b$. This implies that these matrices need to be converted to a number such that the number is larger when the between-class scatter is larger and the within-class scatter is smaller. There are different choices, one example being

$$J(W) = \operatorname{tr}\left\{s_w^{-1} s_b\right\}$$

that can be written as

$$J(W) = \operatorname{tr}\left\{(W S_w W^T)^{-1}(W S_b W^T)\right\}.$$

This one can maximize over W, i.e. we need $W^* = \arg\max_W J(W)$. Recall that W^* defines a projection from our original data space to a new data space with maximal class separability, where the data points $\mathbf{x}_n$ are projected to

$$\mathbf{y}_n = W^{*T} \mathbf{x}_n.$$

Also note that this does not yet allow us to classify any feature $\mathbf{x}$, all it does is to prepare our data space for developing a suitable classifier.

We now proceed to finding the optimal transformation.

4.1.6. Derivation of the optimal transformation. The derivation is a little tricky and before we proceed with the derivation proper, we take a slight detour and derive a result that will prove fundamental, and is interesting in its own right.

4.1.6.1. *Simultaneous diagonalization of two symmetric matrices.* Suppose we are given two symmetric matrices S_w and S_b and we are looking for a transformation that will transform both of them to diagonal matrices. To be more specific, we are looking for a matrix B such that $B^T S_w B = I$, and $B^T S_b B = \Lambda$, where I is the identity matrix and Λ is a diagonal matrix.

Note: If S_w and S_b are the in-class covariance matrices of two different clusters, then B defines a transformation of the data space so that the covariance matrices of the two clusters become I and Λ respectively. (We actually assume that the matrices are covariance matrices so that their eigenvalues are non-negative.)

Algorithm:

(1) Factorize $S_w = LL^T$ by using for example, Cholesky factorization or the Singular Value Decomposition (SVD). If we use the SVD, we write $S_w = U\Sigma U^T = U\Sigma^{1/2}\left(U\Sigma^{1/2}\right)^T$. This means that $L^{-1}S_wL^{-T} = I$.

(2) In the case that S_w is singular we 'remove' the singularity by first writing $S_w = U\Sigma U^T$. More specifically, assume that the rank of S_w is r. By retaining only the first r columns of U, written as $\widehat{U}$, and the r nonzero singular values of Σ, written as Σ_+, the reduced form of the SVD is written as $S_w = \widehat{U}\Sigma_+\widehat{U}^T$, so that $\Sigma_+^{-\frac{1}{2}}\widehat{U}^TS_w\widehat{U}\Sigma_+^{\frac{1}{2}} = I$. The algorithm now proceeds with these values for L. It may be useful to point out that we have reduced the dimension of the space by pruning away the part that belongs to the zero singular values.

(3) Form $K = L^{-1}S_bL^{-T}$.

(4) Since K is symmetric we diagonalize it using an orthonormal matrix Q, i.e. $K = Q\Lambda Q^T$ where $QQ^T = I = Q^TQ$. This means that $Q^TKQ = \Lambda$.

(5) Substituting the expression for K, we find that

$$Q^TL^{-1}S_bL^{-T}Q = \Lambda$$

and

$$Q^TL^{-1}S_wL^{-T}Q = Q^TQ = I.$$

(6) For $B = L^{-T}Q$ we therefore find that $B^TS_wB = I$ and $B^TS_bB = \Lambda$ as required.

Note:

(1) There is a useful interpretation of B,

$$\begin{aligned} S_w^{-1}S_b &= \left(B^{-T}B^{-1}\right)^{-1}B^{-T}\Lambda B^{-1} \\ &= B\Lambda B^{-1}. \end{aligned}$$

Thus B is the eigenvector matrix of $S_w^{-1}S_b$ and Λ is the corresponding eigenvalue matrix.

(2) If calculated this way, the eigenvectors are not normalized. In order to use it as described below, it is important to normalize the columns of B.

4.1.6.2. *Maximization of* $tr\left(s_w^{-1}s_b\right)$. We need to optimize

$$\begin{aligned} J(W) &= \operatorname{tr}\left(s_w^{-1}s_b\right) \\ &= \operatorname{tr}\left(\left(W^T S_w W\right)^{-1}\left(W^T S_b W\right)\right). \end{aligned}$$

The optimum is obtained by setting $\frac{\partial J(W)}{\partial W} = 0$. Taking derivatives of traces of matrices is well-documented and here I'll asked you to accept that the result is given by

$$\left(S_w^{-1}S_b\right)W = W\left(s_w^{-1}s_b\right). \tag{4.2}$$

In order to find an expression for W, we first calculate B, the matrix that simultaneously diagonalize s_w and s_b, as described above. Note:

(1) B is the eigenvector matrix of $s_w^{-1}s_b$.

(2) If we write $\left(s_w^{-1}s_b\right)B = B\Lambda$ where $\Lambda = \operatorname{diag}\begin{pmatrix} \lambda_1 & \cdots & \lambda_{D'} \end{pmatrix}$ then it follows that $\operatorname{tr}\left(s_w^{-1}s_b\right) = \sum_n \lambda_n$.

(3) B transforms the y-space to a z-space through $\mathbf{z} = B^T\mathbf{y}$. This transformation leaves the fundamental quantity $\operatorname{tr}\left(s_w^{-1}s_b\right)$ invariant. This is important because it tells us that we only need to minimize $\sum_n \lambda_n$. In order to show that the trace remains invariant, suppose that $\mathbf{z} = B^T\mathbf{y}$ transforms s_w and s_b to z_w and z_b respectively. This means that $z_w = B^T s_w B$ and $z_b = B^T s_b B$. A quick calculation shows that

$$\begin{aligned} \operatorname{tr}\left(z_w^{-1}z_b\right) &= \operatorname{tr}\left(\left(B^T s_w B\right)^{-1}\left(B^T s_b B\right)\right) \\ &= \operatorname{tr}\left(B^{-1}s_w^{-1}s_b B\right). \end{aligned}$$

This is a similarity transformation which preserves the trace.

We now proceed to show that Λ is also the eigenvalue matrix of the original scatter matrices S_w and S_b. Starting with (4.2), we find that

$$\begin{aligned}\left(S_w^{-1}S_b\right)W &= W\left(s_w^{-1}s_b\right)\\ &= W\left(B\Lambda B^{-1}\right),\end{aligned}$$

or

$$\left(S_w^{-1}S_b\right)WB = WB\Lambda. \tag{4.3}$$

We have finally arrived at our desired result. If we denote the eigenvalues of $S_w^{-1}S_b$ by $\mu_1, \ldots, \mu_D$ it follows that:

(1) $\operatorname{tr}\left(S_w^{-1}S_b\right) = \mu_1 + \cdots + \mu_D$
(2) $\operatorname{tr}\left(s_w^{-1}s_b\right) = \lambda_1 + \cdots + \lambda_{D'}$

Since (4.3) tells that the λ_i's are also the eigenvalues of $S_w^{-1}S_b$, we can maximize $\operatorname{tr}\left(s_w^{-1}s_b\right)$ by selecting the largest eigenvalues of $S_w^{-1}S_b$. The corresponding eigenvectors form the transformation matrix W.

PROBLEM 12. There seems to be something inconsistent here. The eigenvectors of $S_w^{-1}S_b$ are given, according to (4.3), by WB, and not W, which is the transformation matrix we are after. Hint: Recall that B leaves $\operatorname{tr}\left(s_w^{-1}s_b\right)$ invariant. This also shows that the transformation matrix is not unique.

Note:

(1) The question is how to calculate the eigenvectors of $S_w^{-1}S_b$ without having to calculate the inverse of S_w. One possibility is to calculate the matrix that simultaneously diagonalizes both S_w and S_b, as explained above.
(2) The maximum value that we can use for D' is $K-1$, since the maximum rank of S_b is $K-1$. S_b is the sum of K rank one matrices that satisfy a single constraint (4.1)

4.2. Probabilistic Generative Models

Model the class-conditional densities $p(\mathbf{x}|\mathcal{C}_k)$ as well as the priors $p(\mathcal{C}_k)$, Bayes' theorem then gives the posterior probabilities $p(\mathcal{C}_k|\mathbf{x})$.

Consider the case of only two classes,

$$\begin{aligned} p(\mathcal{C}_1|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1) + p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)} \\ &= \frac{1}{1+\exp(-a)} \\ &= \sigma(a) \end{aligned}$$

where

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}.$$

Note that $p(\mathcal{C}_2|\mathbf{x}) = 1 - p(\mathcal{C}_1|\mathbf{x})$.

For more than two classes we have

$$\begin{aligned} p(\mathcal{C}_k|\mathbf{x}) &= \frac{p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k)}{\sum_{j=1}^{K} p(\mathbf{x}|\mathcal{C}_j)p(\mathcal{C}_j)} \\ &= \frac{\exp a_k}{\sum_{j=1}^{K} \exp a_j} \end{aligned}$$

where

$$a_k = \ln \left(p(\mathbf{x}|\mathcal{C}_k)p(\mathcal{C}_k) \right).$$

It is straightforward to verify that $\sum_{j=1}^{K} p(\mathcal{C}_j|\mathbf{x}) = 1$, as it should. The normalized sum of exponentials is also known as the softmax function.

4.2.1. Continuous inputs. Assume that the class-conditional densities are Gaussian. Also assume that they all share the same covariance matrix. The class-conditional density is then given by

$$p(\mathbf{x}|\mathcal{C}_k) = \frac{1}{|2\pi\Sigma|^{1/2}} \exp\left\{ -\frac{1}{2}(\mathbf{x} - \boldsymbol{\mu}_k)\Sigma^{-1}(\mathbf{x} - \boldsymbol{\mu}_k)^T \right\}.$$

For the case of two classes we have

$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + w_0)$$

where

$$\begin{aligned} \mathbf{w} &= \Sigma^{-1}(\boldsymbol{\mu}_1 - \boldsymbol{\mu}_2) \\ w_0 &= -\frac{1}{2}\boldsymbol{\mu}_1^T\Sigma^{-1}\boldsymbol{\mu}_1 + \frac{1}{2}\boldsymbol{\mu}_2^T\Sigma^{-1}\boldsymbol{\mu}_2 + \ln \frac{p(\mathcal{C}_1)}{p(\mathcal{C}_2)}. \end{aligned}$$

Note:

(1) The posterior probabilities only enter through the bias term w_0.

(2) The quadratic terms in the Gaussians cancel because of the shared covariance. This is what makes it a *linear* classifier. To see this, we classify $\mathbf{x}$ as belonging to $\mathcal{C}_1$ if $p(\mathcal{C}_1|\mathbf{x}) > p(\mathcal{C}_2|\mathbf{x})$ other wise to $\mathcal{C}_2$, with the decision boundary given by $p(\mathcal{C}_1|\mathbf{x}) = p(\mathcal{C}_2|\mathbf{x})$. This can be rewritten as $\sigma(\mathbf{w}^T\mathbf{x} + w_0) = 1 - \sigma(\mathbf{w}^T\mathbf{x} + w_0)$, or $\mathbf{w}^T\mathbf{x} + w_0 = \text{const}$.

For K classes we have

$$a_k(\mathbf{x}) = \mathbf{w}_k\mathbf{x} + w_{k0}$$

where

$$\begin{aligned} \mathbf{w}_k &= \Sigma^{-1}\boldsymbol{\mu}_k \\ w_{k0} &= \frac{1}{2}\boldsymbol{\mu}_k^T\Sigma^{-1}\boldsymbol{\mu}_k + \ln p(\mathcal{C}_k). \end{aligned}$$

Again because of the shared covariance, the decision boundary is linear. For non-shared covariance matrices the decision boundaries become quadratic, see Figure 4.2.1.

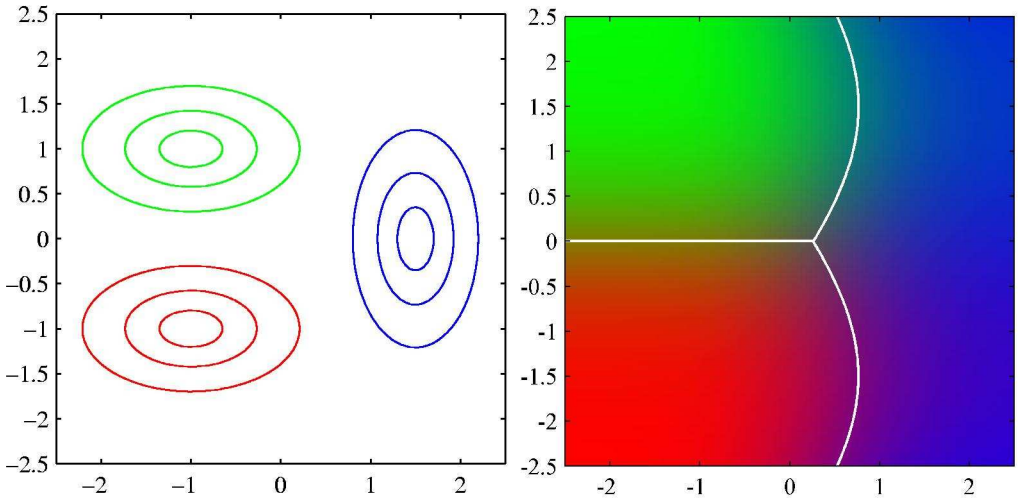


FIGURE 4.2.1. Quadratic discriminant functions.

4.2.2. Maximum likelihood solution. In order to find the parameters of the class conditional densities as well as the prior densities we need a data set of observations together with their class labels.

Again we first consider the case of only two classes and assume Gaussian class-conditional densities with a shared covariance matrix. The data set is denoted by $\{\mathbf{x}_n, t_n\}, \quad n = 1, \ldots, N$. Here $t_n = 1$ denotes class $\mathcal{C}_1$ and $t_n = 0$ denotes class $\mathcal{C}_2$. Also let, $p(\mathcal{C}_1) = \pi$ and $p(\mathcal{C}_2) = 1 - \pi$. For $\mathbf{x}_n \in \mathcal{C}_1$we have that $t_n = 1$ so that

$$p(\mathbf{x}_n, \mathcal{C}_1) = p(\mathcal{C}_1)p(\mathbf{x}_n|\mathcal{C}_1) = \pi\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \Sigma),$$

and similarly for class $\mathcal{C}_2$,

$$p(\mathbf{x}_n, \mathcal{C}_2) = p(\mathcal{C}_2)p(\mathbf{x}_n|\mathcal{C}_2) = (1 - \pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \Sigma).$$

This implies that t has a Bernoulli distribution, $p(t|\pi) = \pi^t(1 - \pi)^{1-t}$. We can now write down the likelihood function as given by the *joint distribution*

$$\begin{aligned}
p(\mathbf{t}, X|\pi, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma) &= p(X|\mathbf{t}, \boldsymbol{\mu}_1, \boldsymbol{\mu}_2, \Sigma)p(\mathbf{t}|\pi) \\
&= \prod_{n=1}^{N} p(\mathbf{x}_n|\boldsymbol{\mu}_1, \Sigma)^{t_n} p(\mathbf{x}_n|\boldsymbol{\mu}_2, \Sigma)^{1-t_n} p(t_n|\pi) \\
&= \prod_{n=1}^{N} p(\mathbf{x}_n|\boldsymbol{\mu}_1, \Sigma)^{t_n} p(\mathbf{x}_n|\boldsymbol{\mu}_2, \Sigma)^{1-t_n} \pi^{t_n}(1 - \pi)^{1-t_n} \\
&= \prod_{n=1}^{N} \left[\pi\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_1, \Sigma)\right]^{t_n} \left[(1 - \pi)\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_2, \Sigma)\right]^{1-t_n}
\end{aligned}$$

where $\mathbf{t} = [t_1, \ldots, t_N]^T$.

Note:

(1) It will be worth your while to study what we are doing here very carefully—we are trying to obtain the likelihood in a consistent manner from a joint distribution.

(2) Although we define the likelihood function in terms of a joint distribution, we are not entirely consistent—the parameters appear not as part of a joint distribution but as conditionals. To be consistent they too should enter the joint distribution as random variables, in which case priors are required.

That this is not a bad idea was already observed when we discussed linear regression. It, among other things, avoids the over fitting problem in case of small amounts of data.

As usual in situations like this it is easier to work with the log-likelihood. Taking the log, the terms that depend on π are given by

$$\sum_{n=1}^{N} \{t_n \ln \pi + (1 - t_n) \ln(1 - \pi)\}.$$

Setting the derivative equal to zero,

$$\pi = \frac{1}{N} \sum_{n=1}^{N} t_n = \frac{N_1}{N} = \frac{N_1}{N_1 + N_2}.$$

This is just the fraction of points in class $\mathcal{C}_1$.

Maximize with respect to $\boldsymbol{\mu}_1$, the terms in the log-likelihood that depend on $\boldsymbol{\mu}_1$ are,

$$\sum_{n=1}^{N} \ln \mathcal{N}(\mathbf{x}_n | \boldsymbol{\mu}_1, \Sigma) = -\frac{1}{2} \sum_{n=1}^{N} t_n (\mathbf{x}_n - \boldsymbol{\mu}_1)^T \Sigma^{-1} (\mathbf{x}_n - \boldsymbol{\mu}_1)$$

Setting the derivative equal to zero,

$$\boldsymbol{\mu}_1 = \frac{1}{N_1} \sum_{n=1}^{N} t_n \mathbf{x}_n$$

which is the mean of the vectors assigned to $\mathcal{C}_1$. Similarly

$$\boldsymbol{\mu}_2 = \frac{1}{N_2} \sum_{n=1}^{N} (1 - t_n) \mathbf{x}_n.$$

Doing the same for the shared covariance, the terms in the log-likelihood are given by

$$-\frac{1}{2}\sum_{n=1}^{N} t_n \ln|\Sigma| - \frac{1}{2}\sum_{n=1}^{N} t_n(\mathbf{x}_n - \boldsymbol{\mu}_1)^T\Sigma^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_1)$$
$$-\frac{1}{2}\sum_{n=1}^{N}(1-t_n)\ln|\Sigma| - \frac{1}{2}\sum_{n=1}^{N}(1-t_n)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T\Sigma^{-1}(\mathbf{x}_n - \boldsymbol{\mu}_2)$$
$$= -\frac{N}{2}\ln|\Sigma| - \frac{N}{2}\mathrm{Tr}\left\{\Sigma^{-1}S\right\}$$

where

$$\begin{aligned} S &= \frac{N_1}{N}S_1 + \frac{N_2}{N}S_2 \\ S_1 &= \frac{1}{N_1}\sum_{n\in\mathcal{C}_1}(\mathbf{x}_n - \boldsymbol{\mu}_1)(\mathbf{x}_n - \boldsymbol{\mu}_1)^T \\ S_2 &= \frac{1}{N_2}\sum_{n\in\mathcal{C}_2}(\mathbf{x}_n - \boldsymbol{\mu}_2)(\mathbf{x}_n - \boldsymbol{\mu}_2)^T . \end{aligned}$$

It now follows that $\Sigma = S$.

The extension to K classes is straightforward.

4.2.3. Naive Bayes classifier. The naive Bayes classifier is a simple yet useful classifier. Let us first point out a a difficulty with the previous approach. The difficulty is entirely computational and has to do with the number of parameters one has to estimate. Assuming a K class problem in D dimensions, and different full covariances for each class (leading to quadratic, not linear decision surfaces), one has to estimate $D + \frac{1}{2}D(D+1)$ parameters for each class, i.e. a total of $\frac{1}{2}KD(D+3)$ parameters for K classes. This can easily become very large in particular when D becomes large. One possibility is to share a full covariance between all the classes as we did above. In this case the number of parameters become $KD + \frac{1}{2}D(D+1)$. Naive Bayes essentially assumes a *diagonal* covariance for each class. This reduces the number of parameters to $2KD$, a substantial saving, especially if D is large. More generally the naive Bayes assumption is that the attributes (the components of the random vector $\mathbf{x}$) are conditionally independent, conditioned on the class, i.e.

if $\mathbf{x} = \begin{bmatrix} x_1 & \cdots & x_D \end{bmatrix}^T$ then

$$p(\mathbf{x}|\mathcal{C}) = \prod_{n=1}^{D} p(x_n|\mathcal{C}).$$

EXERCISE 13. Show that the conditional independence assumption leads to a diagonal covariance matrix if one assumes a Gaussian class conditional pdf.

Using Bayes' theorem we write

$$\begin{aligned} p(\mathcal{C}_k|\mathbf{x}) &= p(\mathcal{C}_k)p(\mathbf{x}|\mathcal{C}_k)/p(\mathbf{x}) \\ &= \frac{p(\mathcal{C}_k)\prod_n p(x_n|\mathcal{C}_k)}{\sum_k p(\mathcal{C}_k)\prod_n p(x_n|\mathcal{C}_k)}. \end{aligned}$$

The naive Bayes classification rule is therefore given by

$$\mathcal{C} = \arg\max_{\mathcal{C}} \frac{p(\mathcal{C})\prod_n p(x_n|\mathcal{C})}{\sum_k p(\mathcal{C}_k)\prod_n p(x_n|\mathcal{C}_k)}$$

which simplifies to (since the denominator does not depend on the class)

$$\mathcal{C} = \arg\max_{\mathcal{C}} p(\mathcal{C}) \prod_n p(x_n|\mathcal{C}).$$

EXERCISE 14. Show, using maximum likelihood, and assuming Gaussian pdf's that

$$\begin{aligned} p(\mathcal{C}_k) &= \frac{N_k}{N} \\ \mu_{nk} &= \frac{1}{N_k}\sum x_{nk} \\ \sigma^2_{nk} &= \frac{1}{N_k}\sum (x_{nk} - \mu_{nk})^2\,,\ n = 1, \ldots, D;\ k = 1, \ldots, K \end{aligned}$$

where N_k is the number of samples in class $\mathcal{C}_k$ and the sums go over all the samples x_{nk} that belong to class $\mathcal{C}_k$.

4.3. Probabilistic Discriminative Models

4.3.1. Logistic Regression. For the two-class problem the posterior probability is given by a logistic sigmoid,

$$p(\mathcal{C}_1|\mathbf{x}) = \frac{1}{1+\exp(-a)} = \sigma(a)$$

where

$$a = \ln \frac{p(\mathbf{x}|\mathcal{C}_1)p(\mathcal{C}_1)}{p(\mathbf{x}|\mathcal{C}_2)p(\mathcal{C}_2)}.$$

Assuming Gaussian densities with a shared covariance we find that

$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^T\mathbf{x} + w_0). \tag{4.1}$$

Thus it should be possible to estimate the coefficients $\mathbf{w}$ directly, instead of estimating the mean and covariance of Gaussians. Furthermore, it is also possible to generalize and expand in terms of nonlinear basis functions,

$$p(\mathcal{C}_1|\mathbf{x}) = \sigma(\mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}))$$

with $p(\mathcal{C}_2|\mathbf{x}) = 1 - p(\mathcal{C}_1|\mathbf{x})$. Now we use maximum likelihood to determine the parameters of this logistic regression problem. Given a data set $\{\boldsymbol{\phi}_n, t_n\}$ where $t_n \in \{0,1\}$ indicates the class that $\mathbf{x}_n$ belongs to, and $\boldsymbol{\phi}_n = \boldsymbol{\phi}(\mathbf{x}_n), \quad n = 1,\ldots,N$ we now want to derive the likelihood function, written as the joint distribution

$$\begin{aligned} p(X,\mathbf{t}|\mathbf{w}) &= p(\mathbf{t}|X,\mathbf{w})p(X) \\ &= p(X)\prod_n p(t_n|X,\mathbf{w}). \end{aligned} \tag{4.2}$$

Recall that X consists of data the belongs to the two classes, $t = 1$ or $t = 0$. Let us now define specific class labels, i.e. let $t = 1$ denote class $\mathcal{C}_1$and let $t = 0$ denote class $\mathcal{C}_2$. This means that X consists of two subsets, X_1 and X_2 belonging to classes $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively. Using this notation the likelihood function can be rewritten as

$$p(X,\mathbf{t}|\mathbf{w}) = p(X)\prod_n p(t_n = 1|X_1,\mathbf{w})^{t_n}(1 - p(t_n = 1|X_1,\mathbf{w}))^{1-t_n}. \tag{4.3}$$

Note that one has a choice in factorizing the joint probability (4.2). The reason for the specific choice in (4.2) is because we have a model for the posterior $p(C|\mathbf{x})$, given by (4.1). Making use of this model, (4.3) is rewritten as

$$\begin{aligned} p(X, \mathbf{t}|\mathbf{w}) &= p(X) \prod_n \sigma(\mathbf{w}^T \boldsymbol{\phi}_n)^{t_n} \left(1 - \sigma(\mathbf{w}^T \boldsymbol{\phi}_n)\right)^{1-t_n} \\ &= p(X) \prod_{n=1}^{N} y_n^{t_n} (1 - y_n)^{1-t_n} \end{aligned}$$

where $y_n = \sigma(\mathbf{w}^T \boldsymbol{\phi}_n)$. Using the negative log-likelihood as an error function we get

$$E(\mathbf{w}) = -\ln p(\mathbf{t}|\mathbf{w}) = -\sum_{n=1}^{N} \{t_n \ln y_n + (1 - t_n) \ln(1 - y_n)\} - \ln p(X).$$

Since $\ln p(X)$ does not depent on $\mathbf{w}$ one can safely ignore this term. The gradient of the error with respect to $\mathbf{w}$ gives

$$\boldsymbol{\nabla} E(\mathbf{w}) = \sum_{n=1}^{N} (y_n - t_n) \boldsymbol{\phi}_n$$

where we use

$$\frac{d\sigma}{da} = a(1 - a).$$

Note: Setting $\boldsymbol{\nabla} E(\mathbf{w}) = 0 = \sum_{n=1}^{N} (y_n - t_n) \boldsymbol{\phi}_n$ results in a nonlinear system of equations that need to be solved numerically using an optimization procedure.

4.3.2. Multiclass logistic regression. For K classes we need K models

$$\begin{aligned} p(\mathcal{C}_k|\mathbf{x}, \mathbf{w}_k) &= \frac{\exp\left(\mathbf{w}_k^T \boldsymbol{\phi}(\mathbf{x})\right)}{\sum_{j=1}^{K} \exp\left(\mathbf{w}_j^T \boldsymbol{\phi}(\mathbf{x})\right)} \\ &= \frac{\exp\left(a_k(\mathbf{x})\right)}{\sum_{j=1}^{K} \exp\left(a_j(\mathbf{x})\right)}, \ k = 1, \ldots, K. \end{aligned}$$

Given data $\mathbf{x}_n$, $\mathbf{t}_n$, we use a 1-of-K coding scheme, i.e. the k-th element of $\mathbf{t}_n$, $t_{kn} = 1$ if $\mathbf{x}_n$ belongs to $\mathcal{C}_k$ with the rest of the elements, $t_{jn} = 0$, $j \neq k$. We derive a suitable

likelihood function from the joint distribution

$$\begin{aligned} p(X,T|\mathbf{w}) &= p(X)\prod_n p(\mathbf{t}_n|X,\mathbf{w}) \\ &= p(X)\prod_n\prod_k p(t_{kn}|\mathbf{x}_n,\mathbf{w}_k)^{t_{kn}}. \end{aligned}$$

The log-likelihood is given by

$$\begin{aligned} \ell(\mathbf{w}) &= \ln p(X) + \sum_n\sum_k t_{kn}\ln p(t_{kn}|\mathbf{x}_n,\mathbf{w}_k) \\ &= \ln p(X) + \sum_n\sum_k t_{kn}\left[a_k(\mathbf{x}_n) - \ln\left(\sum_{j=1}^K \exp a_j(\mathbf{x}_n)\right)\right], \end{aligned}$$

and we need to find $\arg\max_{\mathbf{w}} \ell(\mathbf{w})$, or $\nabla_{\mathbf{w}}\ell(\mathbf{w}) = 0$. Since

$$\nabla_{\mathbf{w}_k}\ell(\mathbf{w}) = \sum_n \left[t_{nk} - \frac{\exp \mathbf{w}_k^T\boldsymbol{\phi}(\mathbf{x}_n)}{\sum_{j=1}^K \exp \mathbf{w}_k^T\boldsymbol{\phi}(\mathbf{x}_n)}\right]\boldsymbol{\phi}(\mathbf{x}_n),\ k = 1,\ldots,K,$$

there are no analytical solutions of $\nabla_w\ell(\mathbf{w}) = 0$, and one has to resort to numerical optimization techniques.

4.4. Laplace approximation

The Laplace approximation of a density function is just the best Gaussian approximation of the density function in the vicinity of a local maximum, see Figure 4.4.1

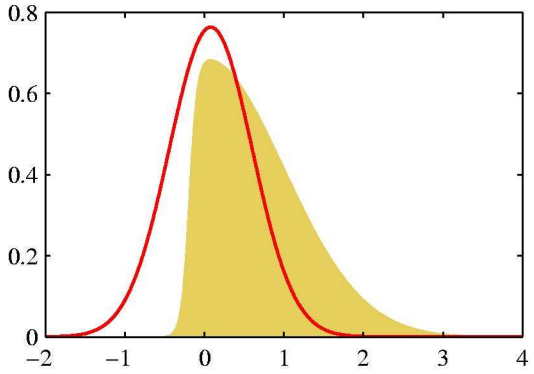


FIGURE 4.4.1. Laplace approximation.

Suppose we have a density function given by

$$p(z) = \frac{1}{Z} f(z),$$

written in this way because we don't care about the normalization coefficient. Suppose further that $p(z)$ has a local maximum at z_0, i.e.

$$\frac{df}{dz}(z_0) = 0.$$

The Taylor expansion of $\ln f(z)$ at z_0 gives

$$\ln f(z) \approx \ln f(z_0) - \frac{1}{2} A (z - z_0)^2$$

where

$$A = -\frac{d^2}{dz^2} \ln f(z_0)$$

so that

$$f(z) \approx f(z_0) \exp\left\{ -\frac{1}{2} A (z - z_0)^2 \right\}.$$

Normalizing we find the Laplace approximation for $p(z)$ in the vicinity of z_0

$$q(z) = \left(\frac{A}{2\pi} \right)^{1/2} \exp\left\{ -\frac{1}{2} A (z - z_0)^2 \right\}.$$

For vector $\mathbf{z}$ the Laplace approximation becomes

$$\begin{aligned} q(\mathbf{z}) &= \left(\left| \frac{A}{2\pi} \right| \right)^{1/2} \exp\left\{ -\frac{1}{2} (\mathbf{z} - \mathbf{z}_0)^T A (\mathbf{z} - \mathbf{z}_0) \right\} \\ &= \mathcal{N}(\mathbf{z} | \mathbf{z}_0, A^{-1}) \end{aligned}$$

where A is the Hessian matrix

$$A = \boldsymbol{\nabla} \boldsymbol{\nabla} \ln f(\mathbf{z}_0).$$

4.5. Bayesian Logistic Regression

In a Bayesian approach we need to put a prior over the coefficients

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \mathbf{m}_0, S_0).$$

The posterior distribution over $\mathbf{w}$ is given by

$$p(\mathbf{w}|\mathbf{t}) \propto p(\mathbf{w})p(\mathbf{t}|\mathbf{w})$$

where we recall that

$$p(\mathbf{t}|\mathbf{w}) = \prod_{n=1}^{N} y_n^{t_n}(1-y_n)^{1-t_n}$$

so that

$$\begin{aligned}\ln p(\mathbf{w}|\mathbf{t}) &= -\frac{1}{2}(\mathbf{w}-\mathbf{m}_0)^T S_0^{-1}(\mathbf{w}-\mathbf{m}_0) \\ &\quad + \sum_{n=1}^{N}\{t_n \ln y_n + (1-t_n)\ln(1-y_n)\} + \text{const}\end{aligned}$$

where $y_n = \sigma(\mathbf{w}^T\boldsymbol{\phi}_n)$. Note that this posterior distribution is not a Gaussian but one can approximate it using Laplace's approximation $q(\mathbf{w})$. In a full Bayesian treatment we now marginalize over the parameters to find the predictive distribution directly

$$p(\mathcal{C}_1|\boldsymbol{\phi},\mathbf{t}) = \int p(\mathcal{C}_1|\boldsymbol{\phi},\mathbf{w})p(\mathbf{w}|\mathbf{t})d\mathbf{w} \approx \int \sigma(\mathbf{w}^T\boldsymbol{\phi})q(\mathbf{w})d\mathbf{w}.$$

The class $\mathcal{C}_2$ is given by $p(\mathcal{C}_2|\boldsymbol{\phi},\mathbf{t}) = 1 - p(\mathcal{C}_1|\boldsymbol{\phi},\mathbf{t})$.

CHAPTER 5

Kernel Methods

One can think of a kernel $k(\mathbf{x}, \mathbf{x}')$ as an inner product between two vectors $\mathbf{x}$ and $\mathbf{x}'$. If one chooses a feature space mapping $\boldsymbol{\phi}(\mathbf{x})$ then one can define a kernel through

$$k(\mathbf{x}, \mathbf{x}') = \boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\phi}(\mathbf{x}').$$

More generally, with any function $k(\mathbf{x}, \mathbf{x}')$ one can identify a Gram matrix K with elements $k(\mathbf{x}_n, \mathbf{x}_m)$. If the Gram matrix associated with a kernel is symmetric positive semi-definite for all possible choices $\mathbf{x}_n$ and $\mathbf{x}_m$, then that function is valid kernel.

EXAMPLE. $k(\mathbf{x}, \mathbf{z}) = (\mathbf{x}^T \mathbf{z})^2$. For a two dimensional input space this becomes

$$\begin{aligned} k(\mathbf{x}, \mathbf{z}) &= (x_1 z_1 + x_2 z_2)^2 \\ &= x_1^2 z_1^2 + 2 x_1 x_2 z_1 z_2 + x_2^2 z_2^2 \\ &= \begin{bmatrix} x_1^2 & \sqrt{2} x_1 x_2 & x_2^2 \end{bmatrix} \begin{bmatrix} z_1^2 & \sqrt{2} z_1 z_2 & z_2^2 \end{bmatrix}^T \\ &= \boldsymbol{\phi}(\mathbf{x})^T \boldsymbol{\phi}(\mathbf{z}). \end{aligned}$$

Note how the two-dimensional input space is first mapped to a three-dimensional input space before the inner product is formed.

5.1. Gaussian processes

The essential idea behind Gaussian processes is to define a prior probability distribution directly over functions. Let us first return to familiar territory.

5.1.1. Linear regression revisited. Consider the linear model

$$y(\mathbf{x}) = \mathbf{w}^T \boldsymbol{\phi}(\mathbf{x})$$

and the prior distribution

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w} | \mathbf{0}, \alpha^{-1} I).$$

The important thing to notice is that the prior probability over $\mathbf{w}$ defines a probability distribution over the function $y(\mathbf{x})$. It is not entirely straightforward to figure out what is probability distribution is. The fact that $\mathbf{w}$ is Gaussian provides a strong hint of course. It should be clear that the mean of $y(\mathbf{x})$ is zero, but how about its covariance? let us go discrete and evaluate $y(\mathbf{x})$ at $\mathbf{x}_m,\ m = 1, \ldots, N$. Then we find that

$$\mathbf{y} = \Phi\mathbf{w}$$

where $\Phi_{nk} = \phi_k(\mathbf{x}_n)$. This is a linear transformation of a Gaussian random variable and therefore $\mathbf{y}$ is also a Gaussian random variable, specified by its mean and covariance.

Note:

- It is only if the transformation matrix is square, and invertible, when it is easy to show that a linear transformation of a Gaussian variable is again Gaussian. In this case Φ is rectangular and it is not so straightforward to show that $\mathbf{y}$ is again a Gaussian variable. It is true however.

EXERCISE 15. Let $x_1$and x_2 be two Gaussian random variables and consider the linear transformation $z = \begin{bmatrix} 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T = x_1 + x_2$. If $\mathcal{N}(x_1|x_{10}, \sigma_1^2)$ and $\mathcal{N}(x_2|x_{20}, \sigma_2^2)$ show that z is also a Gaussian variable and calculate is mean and variance. Hint: Let $z_1 = x_1 + x_2$ and define $z_2 = x_2$. Assume that $x_1$and x_2 are jointly Gaussian and write down the joint distribution for $\mathbf{x} = \begin{bmatrix} x_1 & x_2 \end{bmatrix}^T$. For simplicity first assume that $x_1$and $x_2$are independent. Then investigate the situation when they are dependent.

Now consider the transformation $\mathbf{z} = A\mathbf{x}$ where

$$A = \begin{bmatrix} 1 & 1 \\ 0 & 1 \end{bmatrix}$$

and $\mathbf{z} = \begin{bmatrix} z_1 & z_2 \end{bmatrix}^T$. Calculate the joint Gaussian distribution of $z_1$and z_2 and then find the distribution of z_1 by marginalizing over z_2. (Do it for both cases, when x_1 and x_2 are independent and when they are dependent.)

Returning to our problem, since we know that $\mathbf{y}$ is Gaussian, we only need to calculate its mean and covariance. To wit,

$$\begin{aligned} \mathbb{E}[\mathbf{y}] &= \Phi\mathbb{E}[\mathbf{w}] = 0 \\ \operatorname{cov}[\mathbf{y}] &= \mathbb{E}\left[\mathbf{y}\mathbf{y}^T\right] = \Phi\mathbb{E}\left[\mathbf{w}\mathbf{w}^T\right]\Phi^T = \alpha^{-1}\Phi\Phi^T = K. \end{aligned}$$

Note that K is a Gram matrix with elements

$$K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m) = \alpha^{-1}\boldsymbol{\phi}(\mathbf{x}_n)^T\boldsymbol{\phi}(\mathbf{x}_m)$$

where $k(\mathbf{x}, \mathbf{x}')$ is the kernel function.

Note what we have achieved: Given a function $y(\mathbf{x})$ together with its mean $m(\mathbf{x})$ and covariance in the form of a kernel $k(\mathbf{x}, \mathbf{x}')$ we can sample $y(\mathbf{x})$ at an arbitrary set of points $\mathbf{x}_m,\ m = 1, \ldots, N$ such that the set of values at these points are jointly Gaussian with mean defined by $m(\mathbf{x})$ and the covariance K by the kernel $k(\mathbf{x}, \mathbf{x}')$, i.e.

$$\begin{aligned} m(\mathbf{x}) &= \mathbb{E}[y(\mathbf{x})] \\ k(\mathbf{x}, \mathbf{x}') &= \mathbb{E}\left[(y(\mathbf{x}) - m(\mathbf{x}))(y(\mathbf{x}') - m(\mathbf{x}'))\right]. \end{aligned}$$

It is not important that, given $m(\mathbf{x})$ and $k(\mathbf{x}, \mathbf{x}')$, we cannot write the Gaussian distribution explicitly (or not as far as I can see), we can always evaluate a specific realization as any number of discrete points using the procedure above.

In general we define a Gaussian process as a probability distribution over functions $y(\mathbf{x})$ such that a set of values of $y(\mathbf{x})$ evaluated at an arbitrary set of points jointly have a Gaussian distribution, see Figure 5.1.1. Just to reiterate how the specification of the covariance (kernel function) implies a distribution over functions, choose any number of input points $\mathbf{x}_m,\ m = 1, \ldots, N$ and write out the corresponding covariance matrix K with elements

$$K_{nm} = k(\mathbf{x}_n, \mathbf{x}_m)$$

and mean $\mathbf{m}$ with elements

$$m_n = m(\mathbf{x}_n).$$

Then generate a random Gaussian vector with this mean and covariance from

$$\mathcal{N}(\mathbf{y}|\mathbf{m}, K).$$

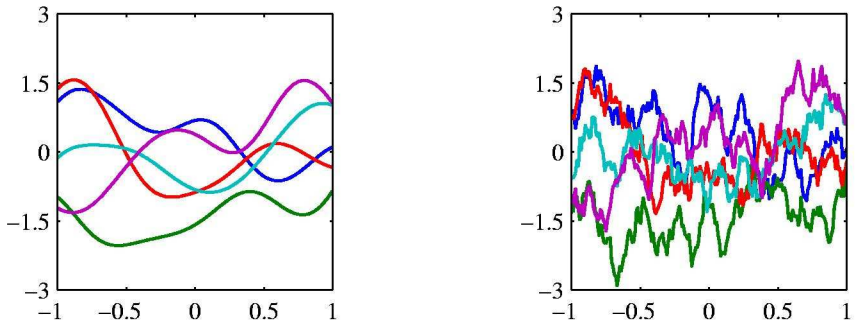


FIGURE 5.1.1. Samples from a Gaussian processes.

5.1.2. Gaussian processes for regression. Let us be specific and use the squared exponential kernel (also a Radial Basis Function)

$$k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{1}{2}\|\mathbf{x} - \mathbf{x}'\|^2\right).$$

There are of course many other possibilities. But notice that the kernel function should express the property that for similar points $\mathbf{x}_n$ the corresponding values of $y_n = y(\mathbf{x}_n)$ should be more strongly correlated that dissimilar points. What we mean by 'similar' depends on the application. For convenience we take the mean of the Gaussian process to be zero.

Since we only have access to noisy data the observed target values are given by

$$t_n = y_n + \epsilon_n.$$

Note that before the values y_n were typically calculated from the input values $\mathbf{x}_n$ through something like $y_n = \mathbf{w}^T\boldsymbol{\phi}(\mathbf{x}_n)$ where the parameters $\mathbf{w}$ are drawn from a Gaussian prior. In our new way of thinking the y_n are obtained from a Gaussian process prior, as described above. Thus the covariance K between the outputs y_n is written as a function of the inputs $\mathbf{x}_n$ through the kernel function and

$$p(\mathbf{y}) = \mathcal{N}(\mathbf{y}|\mathbf{0}, K).$$

The process noise ϵ_n is assumed to be zero mean and independent of y_n, i.e.

$$p(t_n|y_n) = \mathcal{N}(t_n|y_n, \beta^{-1})$$

where β is a hyper-parameter representing the precision of the noise. Assuming that the noise is independent for each data point the joint distribution of the target values $\mathbf{t} = \left[\begin{array}{ccc} t_1 & \cdots & t_N \end{array} \right]^T$ conditioned on the input values $\mathbf{y} = \left[\begin{array}{ccc} y_1 & \cdots & y_N \end{array} \right]^T$ is given by

$$p(\mathbf{t}|\mathbf{y}) = \mathcal{N}(\mathbf{t}|\mathbf{y}, \beta^{-1}I).$$

Note that this is the distribution of the output values, given the input values $\mathbf{y}$ (through the data points $\mathbf{x}_n$), *before we have actually observed any values* t_n. It is now simple matter of finding the marginal distribution $p(\mathbf{t})$,

$$\begin{aligned} p(\mathbf{t}) &= \int p(\mathbf{t}, \mathbf{y}) d\mathbf{y} \\ &= \int p(\mathbf{t}|\mathbf{y}) p(\mathbf{y}) d\mathbf{y} \\ &= \mathcal{N}(\mathbf{t}|\mathbf{0}, C) \end{aligned}$$

where the covariance is given by

$$C(\mathbf{x}_n, \mathbf{x}_m) = k(\mathbf{x}_n, \mathbf{x}_m) + \beta^{-1}\delta_{nm}.$$

Note that we are actually done, we know everything there is to be known about the distribution of the target values, still prior to actually having observed any of them. Suppose we now observe some of the target values $\mathbf{t}_b$, associated with a corresponding set of input values. Writing

$$\mathbf{t} = \left[\begin{array}{c} \mathbf{t}_a \\ \mathbf{t}_b \end{array} \right]$$

we are after $\mathbf{t}_a$ given $\mathbf{t}_b$, i.e. we are after $p(\mathbf{t}_a|\mathbf{t}_b) = \mathcal{N}(\mathbf{t}_a|\mathbf{m}(\mathbf{x}_a), C(\mathbf{x}_a))$. But we already know how to calculate this from the factorized mean and covariance of $p(\mathbf{t})$.

Writing

$$C = \left[\begin{array}{cc} C_{aa} & C_{ab} \\ C_{ba} & C_{bb} \end{array} \right]$$

it follows from (2.5) and (2.6) that

$$C(\mathbf{x}_a) = C_{aa} - C_{ab}C_{bb}^{-1}C_{ba}$$

and

$$\mathbf{m}(\mathbf{x}_a) = C_{ab}C_{bb}^{-1}\mathbf{t}_b.$$

Note that the order of the observed and inferred values are different from that of Bishop.

CHAPTER 6

Principal Component Analysis

In facial recognition we are interested in facial images, and only facial images. Any image is represented as an $m \times n$ matrix where every entry in the matrix (every pixel) represents a shade of gray (the dimension increases three-fold for color images). It should be clear that an $m \times n$ gray-scale images is a specific point in an $m \times n$-dimensional vector space. This can be very high even for low resolution images. Since we are interested in only facial images it might just be possible that the facial images occupy lower dimensional subspace of the full $m \times n$-dimensional images space.

In this chapter we are interested in finding this lower dimensional facial space. In general, we are after a lower dimensional subspace that retains the essential properties of the structures we are studying. As before we need to 'learn' this subspace from representative examples.

6.1. Principal Components

Given N observations $\mathbf{x}_n,\ n = 1, \ldots, N$ with dimension D, we want to find an $M < D$ dimensional subspace with maximum variation. Let us first project onto a one dimensional subspace defined by $\mathbf{u}_1$, where $\mathbf{u}_1^T\mathbf{u}_1 = 1$. This means that we project onto the space spanned $\mathbf{u}_1$, i.e. $\mathbf{x}_n$ is projected onto the scalar value $\left(\mathbf{u}_1^T\mathbf{x}_n\right)$. The idea is to choose $\mathbf{u}_1$ in such a way that the variance of the projected values is a large as possible. If the sample mean is given by

$$\overline{\mathbf{x}} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{x}_n$$

then the mean of the projected data is $\mathbf{u}_1^T\overline{\mathbf{x}}$ and the variance of the projected data is

$$\begin{aligned}\frac{1}{N}\sum_{n=1}^{N}\left(\mathbf{u}_1^T\mathbf{x}_n - \mathbf{u}_1^T\overline{\mathbf{x}}\right)^2 &= \frac{1}{N}\sum_{n=1}^{N}\mathbf{u}_1^T(\mathbf{x}_n-\overline{\mathbf{x}})(\mathbf{x}_n-\overline{\mathbf{x}})^T\mathbf{u}_1\\ &= \mathbf{u}_1^T S\mathbf{u}_1\end{aligned}$$

where S is the sample covariance. Introducing a Lagrange multiplier, we maximize

$$\mathbf{u}_1^T S\mathbf{u}_1 + \lambda(\mathbf{u}_1^T\mathbf{u}_1 - 1),$$

subject to $\mathbf{u}_1^T\mathbf{u}_1 = 1$. This gives the eigenvalue problem

$$S\mathbf{u}_1 = \lambda\mathbf{u}_1,$$

and using the constraint gives

$$\lambda = \mathbf{u}_1^T S\mathbf{u}_1.$$

The projected variance is therefore equals to the eigenvalue λ and attains its maximum if we choose the largest eigenvalue, with $\mathbf{u}_1$ the corresponding eigenvector. This is our first principal vector.

The rest of the principal vectors are obtained by each time projecting onto a vector orthogonal the all the previous principal directions.

Summary: The principal directions are the eigenvectors of the sample covariance matrix belonging to the largest eigenvalues, and the eigenvalues are the variances in these directions.

EXERCISE 16. Show that the eigenvalues of the sample covariance matrix are all non-negative; a fact that is important for our discussion above.

6.2. Numerical Calculation

It is not a good idea to calculate the eigenvalues of the sample covariance matrix numerically since this procedure is unstable. Instead we form the matrix

$$X = \frac{1}{\sqrt{N}}\begin{bmatrix}\mathbf{x}_1 - \overline{\mathbf{x}} & \cdots & \mathbf{x}_N - \overline{\mathbf{x}}\end{bmatrix}$$

and calculate its SVD, $X = U\Sigma V^T$. The columns of U are the principal directions and the squares of the singular values are the variances along these directions. Using

the first M columns of U, i.e. the principal directions belonging to the M largest singular values, the projection onto the M-dimensional space with maximum variances along the principle directions, is given by

$$\mathbf{z} = U_M^T \mathbf{x},$$

where U_M consists of the first M columns of U.

Note: The procedure described on p570 of Bishop is unstable.

6.3. Probabilistic PCA

PCA, as discussed in the previous sections have little, if any, in terms of a probabilistic description. We now discuss a probabilistic alternative.

Suppose $\mathbf{x}$ is the D-dimensional observed vector and $\mathbf{z}$ a corresponding M-dimensional latent vector with

$$\mathbf{x} = W\mathbf{z} + \boldsymbol{\mu} + \boldsymbol{\epsilon}, \tag{6.1}$$

where the noise term $\boldsymbol{\epsilon}$ is a D-dimensional Gaussian variable with zero and covariance $\sigma^2 I$. Note that (6.1) means that $\mathbf{x} - \boldsymbol{\mu} \in \text{col } W$ and the $\mathbf{z}$ are the coordinates. If we also assume that $\mathbf{z}$ is a Gaussian variable with $p(\mathbf{z}) = \mathcal{N}(\mathbf{z}|\mathbf{0}, I)$, then $p(\mathbf{x}|\mathbf{z})$ is also a Gaussian variable given by

$$p(\mathbf{x}|\mathbf{z}) = \mathcal{N}(\mathbf{x}|W\mathbf{z} + \boldsymbol{\mu}, \sigma^2 I).$$

The marginal distribution $p(\mathbf{x})$ is given by

$$\begin{aligned} p(\mathbf{x}) &= \int p(\mathbf{x}, \mathbf{z}) d\mathbf{z} \\ &= \int p(\mathbf{x}|\mathbf{z}) p(\mathbf{z}) d\mathbf{z}. \end{aligned}$$

Recall that this is also a Gaussian variable

$$p(\mathbf{x}) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, C)$$

where the covariance is given by

$$C = WW^T + \sigma^2 I.$$

Note:

(1) This can also be obtained from a direct calculation using (6.1) .
(2) There is considerable redundancy in this formulation. Replacing W with WR, where R is an orthogonal matrix $RR^T = I = R^T R$, leaves the covariance, hence the distribution of $\mathbf{x}$ unchanged.

We need C^{-1}, and a useful formula is

$$C^{-1} = \sigma^{-2} I - \sigma^{-2} W M^{-1} W^T$$

where the $M \times M$ matrix M is given by

$$M = W^T W + \sigma^2 I.$$

The posterior $p(\mathbf{z}|\mathbf{x})$ is also a Gaussian distribution and is given by

$$p(\mathbf{z}|\mathbf{x}) = \mathcal{N}(\mathbf{z}|M^{-1}W^T(\mathbf{x} - \boldsymbol{\mu}), \sigma^{-2}M).$$

We have not made any estimates yet, but note that when W, $\boldsymbol{\mu}$, and σ are known,

$$\mathbf{z} = M^{-1}W^T(\mathbf{x} - \boldsymbol{\mu})$$

gives us the projection of $\mathbf{x}$ onto its principal coordinates.

The various unknowns, W, $\boldsymbol{\mu}$, and σ, are estimated using maximum likelihood. Assuming data $X = \{\mathbf{x}_n,\ n = 1, \ldots, N\}$ the data log likelihood is given by

$$\begin{aligned} \ln p(X|W, \boldsymbol{\mu}, \sigma) &= \sum_{n=1}^{N} p(\mathbf{x}_n|W, \boldsymbol{\mu}, \sigma) \\ &= -\frac{1}{2} N D \ln 2\pi - \frac{N}{2} \ln |C| - \frac{1}{2} \sum_{n=1}^{N} (\mathbf{x}_n - \boldsymbol{\mu})^T C^{-1} (\mathbf{x}_n - \boldsymbol{\mu}) \end{aligned}$$

where $C = WW^T + \sigma^2 I$. Maximizing the log likelihood with respect to $\boldsymbol{\mu}$ is easy and we find that

$$\boldsymbol{\mu} = \overline{\mathbf{x}},$$

the sample mean. Maximizing with respect to W and σ is not easy and we quote Bishop,

$$W_{ML} = U_M (L_M - \sigma^2_{ML} I)^{1/2} R$$

where U_M consists of the D eigenvectors of the sample covariance matrix S belonging to the D largest eigenvalues, i.e. these are exactly the same as the principal directions

of standard PCA. L_M is a diagonal matrix with the corresponding eigenvalues on its diagonal, and R is an arbitrary orthogonal matrix (for simplicity choose $R = I$). Finally,

$$\sigma^2_{ML} = \frac{1}{D-M} \sum_{i=M+1}^{D} \lambda_i,$$

i.e. σ^2_{ML} is the average variance associated with the discarded dimensions.

The important question is, what do we buy with probabilistic PCA? Let us five one answer. The PCA subspace is still large and contains elements that can be far removed from the features we are interested in. In facial recognition for example, the PCA 'face space' contains some decidedly non-facial images. Without a probabilistic representation it is hard to decide whether a feature in the PCA space is actually close to the training set. Once we have equipped the feature space with a probabilistic description we have a systematic way of deciding whether a feature in the PCA space is close to the training set, and therefore something we might be interested in.

CHAPTER 7

Partially observed data and the EM algorithm

In our previous discussion of classification we assumed fully observed data, i.e. each observation came with a class label. We are now ready to start discussing partially observed data. If for example, the data does not come with class labels, the class labels need to be inferred from the observed data. In general, if we have missing data, the missing data need to be inferred from the observed data. We will always assume that, having observed the massing data, the training can be done with ease. A general tool for dealing with partially observed data is the Expectation Maximization (EM) algorithm. The basic idea is very simple. Since we can proceed with the training (estimation of the parameter values) for the fully observed data, we estimate the missing values by calculating an expectation, based on the current estimate of the parameters. Once we have values for the missing data, we proceed to maximize a likelihood to get an updated estimate for the parameters. These are then used to re-estimate the missing data, etc.

The simplest example the the EM algorithm is K-means clustering, and that is where we start.

7.1. K-Means Clustering

We have N observations, each observation belonging to one of K clusters, but we don't know which one. Our task is to assign each observation to an appropriate cluster. As soon as we have done that, our usual classification techniques apply.

First we introduce some notation. With each D-dimensional observation $\mathbf{x}_n$ we associate a cluster indicator $\mathbf{t}_n = \begin{bmatrix} t_{n1} & \cdots & t_{nK} \end{bmatrix}^T$ where $t_{nk} = 1$ if $\mathbf{x}_n$ belongs to cluster k, otherwise it is zero. In the classification problem the $\mathbf{t}_n$ are known, in the clustering problem they are unknown and need to be inferred from the data. Given K, D-dimensional vectors $\boldsymbol{\mu}_k$, which we think of as cluster centers, we define an

objective function as

$$J = \sum_{n=1}^{N} \sum_{k=1}^{K} t_{nk} \left\| \mathbf{x}_n - \boldsymbol{\mu}_k \right\|^2 .$$

Our goal becomes to find the t_{nk} and $\boldsymbol{\mu}_k$ so as to minimize J. This proceeds in two steps. First we choose initial values for the $\boldsymbol{\mu}_k$. Keeping these fixed we minimize J with respect to t_{nk}. In the second step we keep the t_{nk} fixed and minimize J with respect to the $\boldsymbol{\mu}_k$. These steps are repeated until convergence.

The t_{nk} are easily determined. For each data point $\mathbf{x}_n$ assign it to the cluster for which $\left\| \mathbf{x}_n - \boldsymbol{\mu}_k \right\|^2$ is the smallest. This means that each data point is assigned to its nearest cluster, as defined by its estimated cluster center $\boldsymbol{\mu}_k$.

Setting the derivative of J with respect to $\boldsymbol{\mu}_k$ equal to zero, we get

$$\boldsymbol{\mu}_k = \frac{\sum_n t_{nk} \mathbf{x}_n}{\sum_n t_{nk}}.$$

This means that we set $\boldsymbol{\mu}_k$ equal to the sample mean of all the points assigned to cluster k.

Note:

(1) It is common practice to rescale the data before the clustering algorithm is applied, by, for example, whitening the data. This ensures that the data is normalized so that the distance measures have some objective meaning. Note however, the warning given by Duda and Hart that by doing this one runs the risk of rescaling distances when they actually five an indication of between cluster separation.

(2) The K-means algorithm uses a *hard* assignment of each data point to a specific class. This does not take into account that at least for some points there may be considerable ambiguity as to which class they belong. In such cases it may be better to use a *soft* assignment where data points are assigned to clusters in such a way as to reflect the uncertainty.

7.2. Mixture of Gaussians

We have seen that Gaussian pdf's have particularly useful analytical properties. In practice however, one often encounters situations where the data cannot be accurately represented by a single Gaussian pdf. This is the case for all multi-modal data sets. One possibility is to represent the data by a mixture of K Gaussians,

$$p(\mathbf{x}) = \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k). \tag{7.1}$$

The parameters to estimate are the class mean and covariance, $\boldsymbol{\mu}_k$ and Σ_k respectively, as well as the 'class probabilities', π_k. As always these need to be estimated from the data, $\mathbf{x}_n,\ n = 1, \ldots, N$.

If we knew to which component of the mixture to assign each data point, the learning problem would be trivial. Since we don't know it, we need to proceed in a more principled way. Accordingly, we introduce a latent random variable $\mathbf{z} = \begin{bmatrix} z_1 & \cdots & z_K \end{bmatrix}^T$ where $z_k \in \{0, 1\}$ and $\sum_k z_k = 1$. This means that $\mathbf{z}$ can be in one of K states, according to which element is non-zero. We'll construct the joint distribution $p(\mathbf{x}, \mathbf{z})$ from the marginal $p(\mathbf{z})$ and the conditional distribution $p(\mathbf{x}|\mathbf{z})$. The marginal contribution over $\mathbf{z}$ is defined in terms of the mixing coefficients,

$$p(z_k = 1) = \pi_k,$$

where $0 < \pi_k < 1$ and $\sum_k \pi_k = 1$. Note that we can also write,

$$p(\mathbf{z}) = \prod_{k=1}^{K} \pi_k^{z_k}.$$

The conditional distribution is given by

$$p(\mathbf{x}|z_k = 1) = \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)$$

which can also be rewritten as

$$p(\mathbf{x}|\mathbf{z}) = \prod_{k=1}^{K} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)^{z_k}.$$

The joint distribution is given by $p(\mathbf{x}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{z})$ and the marginal distribution $p(\mathbf{x})$ becomes

$$\begin{aligned} p(\mathbf{x}) &= \sum_{\mathbf{z}} p(\mathbf{x}|\mathbf{z})p(\mathbf{z}) \\ &= \sum_{\mathbf{z}} \prod_{k=1}^{K} \pi_k^{z_k} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)^{z_k} \\ &= \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k). \end{aligned} \tag{7.2}$$

Thus the marginal distribution is a mixture model of the form (7.1).

We now compute the conditional probability $p(\mathbf{z}|\mathbf{x})$. Defining $\gamma(z_k) = p(z_k = 1|\mathbf{x})$, it follows from Bayes' theorem that

$$\begin{aligned} \gamma(z_k) &= \frac{p(z_k = 1)p(\mathbf{x}|z_k = 1)}{\sum_{j=1}^{K} p(z_j = 1)p(\mathbf{x}|z_j = 1)} \\ &= \frac{\pi_k \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_j, \Sigma_j)}. \end{aligned}$$

We view π_k as the prior probability of $z_k = 1$, and $\gamma(z_k)$ as the posterior probability once we have observed $\mathbf{x}$. Alternatively, $\gamma(z_k)$ can be viewed as the *responsibility* that the k-th component takes for 'explaining' the observation $\mathbf{x}$.

EXAMPLE 17. Use *ancestral sampling* to generate samples from the joint probability using three mixture components gives Figure 7.2.1. In (a) the samples as drawn from the different components are shown. (This corresponds to fully observed data where each data point is assigned a 'cluster'.) In (b) the data is shown without its mixture components; this is the way it will be presented. In (c) the responsibility that each component takes for every data point is indicated with different color shades.

Suppose we are given N, D-dimensional data points $\mathbf{x}_n$ collected into an $N \times D$ matrix X. The data likelihood is defined as the joint distribution $p(X, \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma)$ as usual. It may be useful to note that we write down the joint distribution between the observed data, and the parameters to be estimated. The latent variables enter

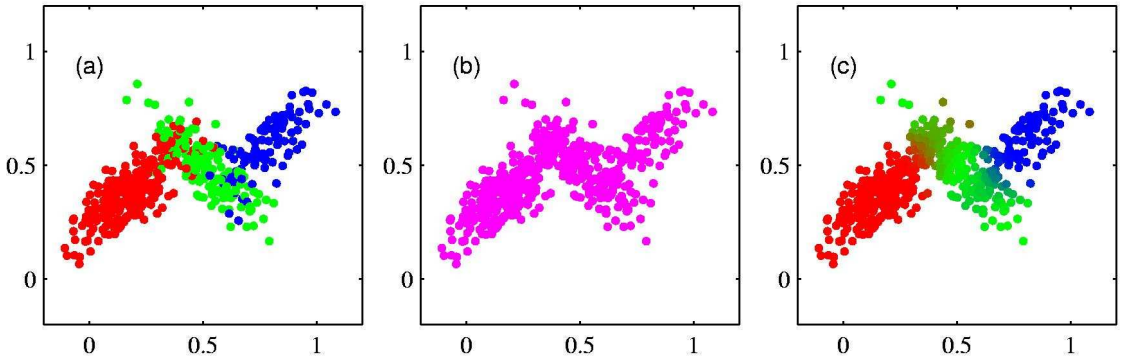


FIGURE 7.2.1. Generating samples from a mixture of 3 Gaussians.

through the $\boldsymbol{\pi}$ parameters. The data likelihood is therefore given by

$$\begin{aligned} p(X, \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) &= p(X|\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) p(\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) \\ &= p(\pi, \boldsymbol{\mu}, \Sigma) \prod_{n=1}^{N} p(\mathbf{x}_n|\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) \\ &= p(\pi, \boldsymbol{\mu}, \Sigma) \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k) \end{aligned}$$

where we have used (7.2) in the last step. The log likelihood is therefore given by

$$\ln p(X, \boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) = \ln p(\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) + \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k) \right\}.$$

If we can't be bothered with providing prior probabilities over the parameters (or maximizing over this more complex expression), we simplify and deal directly with

$$\ln p(X|\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) = \sum_{n=1}^{N} \ln \left\{ \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k) \right\}. \tag{7.3}$$

It is important to note that for mixture models the likelihood can actually become infinity, i.e. it has a singularity. This means that one has to be very careful to avoid runing into the singularity. To see how this can happen, consider a mixture model where all the components have covariances of the form $\sigma_k^2 I$. Suppose one of the mixture components has its mean, say $\boldsymbol{\mu}_j$, exactly equal to one of the data points, i.e. $\boldsymbol{\mu}_j = \mathbf{x}_n$. The contribution of this data point to the likelihood function is of the

form

$$\mathcal{N}(\mathbf{x}_n|\mathbf{x}_n, \sigma_j^2 I) = \frac{1}{\sqrt{2\pi\sigma_j^2}}.$$

This goes to infinity as $\sigma_j \to 0$. Thus this particular component collapses onto a specific data point. This is always a possibility and one has to take steps in order to avoid this situation. (Maybe it was not such a good idea after all to get rid of the priors above.) Also note that this danger does not exist for single Gaussians.

7.3. EM for Gaussian Mixture Models

Setting the partial derivative of the log-likelihood (7.3) with respect to $\boldsymbol{\mu}_k$ equal to zero gives

$$\begin{aligned} 0 &= -\sum_{n=1}^{N} \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \Sigma_j)} \Sigma_k (\mathbf{x}_n - \boldsymbol{\mu}_k) \\ &= -\sum_{n=1}^{N} \gamma(z_{nk}) \Sigma_k (\mathbf{x}_n - \boldsymbol{\mu}_k), \end{aligned}$$

or

$$\boldsymbol{\mu}_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk}) \mathbf{x}_n, \tag{7.1}$$

where

$$N_k = \sum_{n=1}^{N} \gamma(z_{nk}).$$

Note that all the data points contribute towards the mean of the k-th mixture component, weighted with its responsibility to the k-th component. The expression for the covariance follows similar lines,

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^{N} \gamma(z_{nk})(\mathbf{x}_n - \boldsymbol{\mu}_k)(\mathbf{x}_n - \boldsymbol{\mu}_k)^T.$$

Again all the data points contribute towards the k-th mixture component, again weighted with its responsibility.

In order to find the mixing coefficients we need to impose the constraint $\sum_{k=1}^{K} \pi_k = 1$. This is done by introducing a Lagrange multiplier into the log-likelihood,

$$\ln p(X|\boldsymbol{\pi}, \boldsymbol{\mu}, \Sigma) + \lambda \left(\sum_{k=1}^{K} \pi_k - 1 \right).$$

Maximizing this with respect to π_k, yields,

$$0 = \sum_{n=1}^{N} \frac{\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_j \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \Sigma_j)} + \lambda.$$

Multiply both sides with π_k and summing over k gives $\lambda = -N$. Again multiplying both sides with π_k gives

$$0 = \sum_{n=1}^{N} \gamma(z_{nk}) - \pi_k N,$$

or

$$\pi_k = \frac{N_k}{N},$$

which is the average responsibility for the k-th mixture component.

Note these equations are not solvable in closed form since the responsibility depends in a complicated way on the parameters. Their form however, does suggest the following algorithm:

ALGORITHM 18. *Expectation Maximization for GMMs*

1. Initialize the the different coefficients, $\boldsymbol{\pi}$, $\boldsymbol{\mu}$, and Σ. One possibility is to use the K-mean algorithm. Using these estimates evaluate the log-likelihood.

*2. **E step**. Evaluate the responsibilities using the current estimates,*

$$= \gamma(z_{nk}) = \frac{\pi_k \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k, \Sigma_k)}{\sum_{j=1}^{K} \pi_j \mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_j, \Sigma_j)}.$$

3. ***M step***. *Re-estimate the parameters using the current responsibilities*

$$\begin{aligned}\boldsymbol{\mu}_k^{new} &= \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})\mathbf{x}_n \\ \Sigma_k^{new} &= \frac{1}{N_k}\sum_{n=1}^{N}\gamma(z_{nk})(\mathbf{x}_n-\boldsymbol{\mu}_k)(\mathbf{x}_n-\boldsymbol{\mu}_k)^T \\ \pi_k^{new} &= \frac{N_k}{N}\end{aligned}$$

where

$$N_k = \sum_{n=1}^{N}\gamma(z_{nk}).$$

4. *Evaluate the log likelihood*

$$\ln p(X|\boldsymbol{\pi},\boldsymbol{\mu},\Sigma) = \sum_{n=1}^{N}\ln\left\{\sum_{k=1}^{K}\pi_k\mathcal{N}(\mathbf{x}_n|\boldsymbol{\mu}_k,\Sigma_k)\right\},$$

check for convergence of either the parameters, or the log likelihood.

5. *Keep iterating from step 2 until convergence.*

CHAPTER 8

Kalman Filters

Up to now we have only used i.i.d. data. The next major step is to drop this assumption. The first example that we'll encounter where the observations are no longer independent is the Kalman filter.

8.1. Kalman Filter Equations

Suppose we know that the dynamics of the system under consideration is given by

$$\mathbf{x}_{n+1} = A_n \mathbf{x}_n + \boldsymbol{\omega}_n \tag{8.1}$$

where $\mathbf{x}_n$ is the feature vector describing the system at time n, A_n is a matrix describing the transition from $\mathbf{x}_n$to $\mathbf{x}_{n+1}$. This means that the process is linear. This is of course a strong assumption. It is possible to generalize, in fact, the derivation of the Kalman filter given in this Chapter is eminently suitable for generalization. The final term $\boldsymbol{\omega}_n$ is a noise term. It may represent inadequate knowledge of the complete system, and/or simply noise introduced during the evolution of the system. We assume that

$$p(\boldsymbol{\omega}_n) \quad = \quad \mathcal{N}(\boldsymbol{\omega}_n | \mathbf{0}, Q_n)$$

and that the $\boldsymbol{\omega}_n$ are independent for different n.

Apart from the dynamics, we are also able to observe the system. The observation equations are given by

$$\mathbf{z}_{n+1} = H_n \mathbf{x}_{n+1} + \boldsymbol{\nu}_{n+1}. \tag{8.2}$$

The system is not directly observed and the observations $\mathbf{z}_{n+1}$ are linked to the features by the matrix H_n. Again a linear relationship is observed, something that can and should be generalized. The observation noise is given by $\boldsymbol{\nu}_{n+1}$and is given

by

$$\boldsymbol{\nu}_{n+1} \;=\; \mathcal{N}(\boldsymbol{\nu}_{n+1}|\mathbf{0}, R_n).$$

We again assume that the $\boldsymbol{\nu}_{n+1}$ are independent for different n. Since we assume linear processes and Gaussian noise, it should be clear that the state variables are also Gaussian, assuming Gaussian initial values. This makes life much easier, in fact, it allows analytical solutions of the relevant equations.

Our task is to obtain the best possible estimate of the system $\mathbf{x}_n$, given the observations $Z_n = \{\mathbf{z}_j\}\,,\;\; j = 1, \ldots, n$, as well as our knowledge of the noise terms, i.e. given the covariances Q_n and R_n. For this purpose we rewrite the equations in probabilistic form. The dynamic equation becomes,

$$p(\mathbf{x}_{n+1}|\mathbf{x}_n, Z_n) = \mathcal{N}(\mathbf{x}_{n+1}|A_n\mathbf{x}_n, Q_n, Z_n) \tag{8.3}$$

and the observation equation becomes

$$p(\mathbf{z}_{n+1}|\mathbf{x}_{n+1}, Z_n) \;=\; p(\mathbf{z}_{n+1}|\mathbf{x}_{n+1}) \tag{8.4}$$

$$\;=\; \mathcal{N}(\mathbf{z}_{n+1}|H_{n+1}\mathbf{x}_{n+1}, R_n) \tag{8.5}$$

Assuming that we know $p(\mathbf{x}_n|Z_n)$, our approach is then to calculate $p(\mathbf{x}_{n+1}|Z_{n+1})$, given the latest observation $\mathbf{z}_{n+1}$. It will be useful to write

$$p(\mathbf{x}_n|Z_n) = \mathcal{N}(\mathbf{x}_n|\widehat{\mathbf{x}}_n, P_n, Z_n).$$

Note:

(1) The dynamic equation explicitly acknowledge that we take all previous observations into account.

(2) Although the observations are not independent, they are *conditionally* independent. This means that $\mathbf{z}_{n+1}$ becomes independent of all earlier observations once $\mathbf{x}_{n+1}$ is known. More specifically, this means that

$$p(\mathbf{z}_{n+1}|\mathbf{x}_{n+1}, Z_n) = p(\mathbf{z}_{n+1}|\mathbf{x}_{n+1}).$$

Conditional independence is a fundamental concept. Its power derives from the fact that is allows significant simplifications, something that is fully epxploited in Graphical Models. In general, $\mathbf{x}$ is conditionally indepent of $\mathbf{y}$

given $\mathbf{z}$ is written as

$$\mathbf{x} \amalg \mathbf{y}|\mathbf{z}.$$

Recall that the mathematical statement for $\mathbf{x}$ to be independent of $\mathbf{y}$ is given by $p(\mathbf{x}, \mathbf{y}) = p(\mathbf{x})p(\mathbf{y})$. The equivalent statement for conditional independence is

$$p(\mathbf{x}, \mathbf{y}|\mathbf{z}) = p(\mathbf{x}|\mathbf{z})p(\mathbf{y}|\mathbf{z})$$

or equivalently

$$p(\mathbf{x}|\mathbf{y}, \mathbf{z}) = p(\mathbf{x}|\mathbf{z})$$

the form that we used above in connection with the conditionaly independence of the observations.

(3) This formulation leads naturally to a recursive implementation. Given $p(\mathbf{x}_0|Z_0)$ initially, we calculate $p(\mathbf{x}_n|Z_n)$ every time we receive a new observation $\mathbf{z}_n$.

The marginal distribution is then given by

$$\begin{aligned} p(\mathbf{x}_{n+1}|Z_n) &= \int p(\mathbf{x}_{n+1}, \mathbf{x}_n|Z_n)d\mathbf{x}_n \\ &= \int p(\mathbf{x}_{n+1}|\mathbf{x}_n, Z_n)p(\mathbf{x}_n|Z_n)d\mathbf{x}_n. \end{aligned}$$

Since we know both Gaussian distributions under the integration, it follows immediately that

$$p(\mathbf{x}_{n+1}|Z_n) = \mathcal{N}(\mathbf{x}_{n+1}|A_n\widehat{\mathbf{x}}_n, Q_n + A_nP_nA_n^T). \tag{8.6}$$

Implicit in this equation is the assumption that the state features $\mathbf{x}_n$ are independent of the noise.

Suppose we need the best estimate of the state variables $\mathbf{x}_{n+1}$ prior to observing $\mathbf{z}_{n+1}$. Usually one has to be careful in specifying exactly what is meant with 'best' estimate. Since we are dealing with Gaussian densities however, 'best' estimate in just about every meaningful sense of the word amounts to the same quantity namely the mean. Rewriting (8.6) as

$$p(\mathbf{x}_{n+1}|Z_n) = \mathcal{N}(\mathbf{x}_{n+1}|\mathbf{x}_{n+1}^-, P_{n+1}^-)$$

with

$$\mathbf{x}_{n+1}^{-} = A_n \widehat{\mathbf{x}}_n \tag{8.7}$$

and

$$P_{n+1}^{-} = Q_n + A_n P_n A_n^T$$

the best estimate of $\mathbf{x}_{n+1}$ prior to observing $\mathbf{z}_{n+1}$ is given by (8.7).

The marginal distribution of the observation $\mathbf{z}_{n+1}$ is given by

$$\begin{aligned} p(\mathbf{z}_{n+1}|Z_n) &= \int p(\mathbf{z}_{n+1}, \mathbf{x}_{n+1}|Z_n) d\mathbf{x}_{n+1} \\ &= \int p(\mathbf{z}_{n+1}|\mathbf{x}_{n+1}, Z_n) p(\mathbf{x}_{n+1}|Z_n) d\mathbf{x}_{n+1} \\ &= \int p(\mathbf{z}_{n+1}|\mathbf{x}_{n+1}) p(\mathbf{x}_{n+1}|Z_n) d\mathbf{x}_{n+1}. \end{aligned}$$

Since both terms in the integrand are Gaussian, it follows in a straightforward way that the observation marginal is given by

$$p(\mathbf{z}_{n+1}) = \mathcal{N}(\mathbf{z}_{n+1}|\mathbf{z}_{n+1}^{-}, S_{n+1})$$

where

$$\mathbf{z}_{n+1}^{-} = H_{n+1} \mathbf{x}_{n+1}^{-} \tag{8.8}$$

and

$$S_{n+1} = R_{n+1} + H_{n+1} P_{n+1}^{-} H_{n+1}^T. \tag{8.9}$$

Note:

(1) Since $\mathbf{x}_{n+1}^{-}$ is the best estimate of $\mathbf{x}_{n+1}$, $\mathbf{z}_{n+1}^{-}$ given by (8.8) is the best estimate of what we expect the next measurement to be.

Since we expect to next measure $\mathbf{z}_{n+1}^{-}$, the extent to which the actual measurement deviates from this expectation, is a measure of the how much we learn from the

measurement at $n+1$. Using Bayes' theorem we now calculate

$$\begin{aligned} p(\mathbf{x}_{n+1}|Z_{n+1}) &= p(\mathbf{x}_{n+1}|\mathbf{z}_{n+1}, Z_n) \\ &= p(\mathbf{z}_{n+1}|\mathbf{x}_{n+1}, Z_n)p(\mathbf{x}_{n+1}|Z_n)/p(\mathbf{z}_{n+1}|Z_n) \\ &= p(\mathbf{z}_{n+1}|\mathbf{x}_{n+1})p(\mathbf{x}_{n+1}|Z_n)/p(\mathbf{z}_{n+1}|Z_n) \end{aligned}$$

where we made use of the various conditionally independencies. It again follows in s straightforward manner that

$$p(\mathbf{x}_{n+1}|Z_{n+1}) = \mathcal{N}(\mathbf{x}_{n+1}|\widehat{\mathbf{x}}_{n+1}, P_{n+1})$$

where

$$\widehat{\mathbf{x}}_{n+1} = P_{n+1}\left(P_{n+1}^{-}\right)^{-1}\mathbf{x}_{n+1}^{-} + P_{n+1}H_{n+1}^{T}R_{n+1}^{-1}\mathbf{z}_{n+1} \tag{8.10}$$

and

$$P_{n+1} = \left[\left(P_{n+1}^{-}\right)^{-1} + H_{n+1}^{T}R_{n+1}H_{n+1}\right]^{-1}. \tag{8.11}$$

We are almost done. These equations give us exactly what we want. The best estimate of $\mathbf{x}_{n+1}$, given the observation sequence Z_{n+1}, is given by $\widehat{\mathbf{x}}_{n+1}$with associated covariance P_{n+1}. All that remains to be done is to rewrite these equations in a more convenient form. For that purpose we define the so-called Kalman gain

$$W_{n+1} = P_{n+1}^{-}H_{n+1}^{T}S_{n+1}^{-1}. \tag{8.12}$$

It takes a bit of manipulation but it is not hard to show that this allows us to write

$$P_{n+1} = P_{n+1}^{-} - W_{n+1}S_{n+1}W_{n+1}^{T} \tag{8.13}$$

and

$$\widehat{\mathbf{x}}_{n+1} = \mathbf{x}_{n+1}^{-} + W_{n+1}\left(\mathbf{z}_{n+1} - \mathbf{z}_{n+1}^{-}\right). \tag{8.14}$$

ALGORITHM. *Given $\widehat{\mathbf{x}}_n$ and P_n:*

1. $\mathbf{x}_{n+1}^{-} = A_n\widehat{\mathbf{x}}_n$, and $P_{n+1}^{-} = Q_n + A_nP_nA_n^T$ (using the dynamic information)

2. $\mathbf{z}_{n+1}^{-} = H_{n+1}\mathbf{x}_{n+1}^{-}$ and $S_{n+1} = R_{n+1} + H_{n+1}P_{n+1}^{-}H_{n+1}^{T}$ (using the measurement equation)

3. $W_{n+1} = P_{n+1}^{-}H_{n+1}^{T}S_{n+1}^{-1}$ (Kalman gain)

4. Given the new measurement $\mathbf{z}_{n+1}$*, calculate*

$$\widehat{\mathbf{x}}_{n+1} = \mathbf{x}_{n+1}^- + W_{n+1}\left(\mathbf{z}_{n+1} - \mathbf{z}_{n+1}^-\right) \text{ and } P_{n+1} = P_{n+1}^- - W_{n+1}S_{n+1}W_{n+1}^T.$$

8.1.1. Simplifying Using the Kalman Gain. We give the details of deriving (8.13) and (8.14) from (8.10) and (8.11) respectively.

It is the easiest to verify that

$$\begin{aligned} P_{n+1} &= \left[\left(P_{n+1}^-\right)^{-1} + H_{n+1}^T R_{n+1} H_{n+1}\right]^{-1} \\ &= P_{n+1}^- - W_{n+1}S_{n+1}W_{n+1}^T \end{aligned}$$

by showing that

$$\left[\left(P_{n+1}^-\right)^{-1} + H_{n+1}^T R_{n+1} H_{n+1}\right]\left[P_{n+1}^- - W_{n+1}S_{n+1}W_{n+1}^T\right] = I.$$

Multiplying out we find that

$$\begin{aligned} &\left[\left(P_{n+1}^-\right)^{-1} + H_{n+1}^T R_{n+1} H_{n+1}\right]\left[P_{n+1}^- - W_{n+1}S_{n+1}W_{n+1}^T\right] = \\ &I - H_{n+1}^T S_{n+1}^{-T} H_{n+1}\left(P_{n+1}^-\right)^T + H_{n+1}^T R_{n+1}^{-1} H_{n+1} P_{n+1}^- - \\ &\qquad H_{n+1}^T R_{n+1}^{-1} H_{n+1} P_{n+1}^- H_{n+1}^T S_{n+1}^{-T} H_{n+1}\left(P_{n+1}^-\right)^T = \\ &I - H_{n+1}^T S_{n+1}^{-T} H_{n+1}\left(P_{n+1}^-\right)^T + H_{n+1}^T R_{n+1}^{-1} H_{n+1} P_{n+1}^- - \\ &\qquad H_{n+1}^T R_{n+1}^{-1}\left(S_{n+1} - R_{n+1}\right) S_{n+1}^{-T} H_{n+1}\left(P_{n+1}^-\right)^T = I, \end{aligned}$$

where we have used (8.9) and the fact that covariance matrices are symmetric.

Starting from (8.10) we find that

$$\begin{aligned} \widehat{\mathbf{x}}_{n+1} &= P_{n+1}\left(P_{n+1}^-\right)^{-1}\mathbf{x}_{n+1}^- + P_{n+1}H_{n+1}^T R_{n+1}^{-1}\mathbf{z}_{n+1} \\ &= \left(P_{n+1}^- - W_{n+1}S_{n+1}W_{n+1}^T\right)\left(P_{n+1}^-\right)^{-1}\mathbf{x}_{n+1}^- + \left(P_{n+1}^- - W_{n+1}S_{n+1}W_{n+1}^T\right)H_{n+1}^T R_{n+1}^{-1}\mathbf{z}_{n+1} \\ &= \mathbf{x}_{n+1}^- - P_{n+1}^- H_{n+1}^T S_{n+1}^{-1} S_{n+1} S_{n+1}^{-T} H_{n+1} P_{n+1}^-\left(P_{n+1}^-\right)^{-1}\mathbf{x}_{n+1}^- \\ &\quad + \left(P_{n+1}^- - W_{n+1}S_{n+1}W_{n+1}^T\right)H_{n+1}^T R_{n+1}^{-1}\mathbf{z}_{n+1} \\ &= \mathbf{x}_{n+1}^- - P_{n+1}^- H_{n+1}^T S_{n+1}^{-1}\mathbf{z}_{n+1}^- + \left(P_{n+1}^- - W_{n+1}S_{n+1}W_{n+1}^T\right)H_{n+1}^T R_{n+1}^{-1}\mathbf{z}_{n+1} \\ &= \mathbf{x}_{n+1}^- - W_{n+1}\mathbf{z}_{n+1}^- + W_{n+1}S_{n+1}\left(I - W_{n+1}^T H_{n+1}^T\right)R_{n+1}^{-1}\mathbf{z}_{n+1}. \end{aligned}$$

If now use (8.9) it follows that

$$\begin{aligned} S_{n+1}\left(I - W_{n+1}^T H_{n+1}^T\right) R_{n+1}^{-1} &= I + H_{n+1} P_{n+1}^- H_{n+1}^T R_{n+1}^{-1} - \\ & \quad S_{n+1} S_{n+1}^{-1} H_{n+1} P_{n+1}^- H_{n+1}^T R_{n+1}^{-1} \\ &= I, \end{aligned}$$

and we are done.

CHAPTER 9

HIDDEN MARKOV MODELS

9.1. Introduction.

Hidden Markov models (HMMs) and Kalman filters are related in the sense that both are models where the usually unobserved internal state governs the external observations that we make. Furthermore the internal state at the next time step is only determined by the current internal state (the so-called 1st-order Markov property). However, they are also very different. Firstly, while the Kalman filter has a continuous state space (i.e. its internal state is described by a set of real-valued numbers), the HMM has a discrete state space (i.e. its state can be depicted by a limited set of integers). The Kalman filter changes its state in a linear manner (i.e. a matrix multiplication will take you from its current state to the next). The HMM changes its state in a probabilistic manner, a transition matrix specifies which states can follow on the current one and with what probability this will happen. This makes its behavior to be very non-linear. Although these links are intriguing, we will develop the our understanding of the HMM by taking a further step back to pick up the thread from the Naive Bayes approach we previously encountered.

In order to set the scene we first take a small detour and discuss a relatively straightforward application of HMMs, namely signature verification.

9.2. Signature Verification

One might well ask whether automated signature verification systems are still relevant. After all, there is such a wide choice of personal biometric identification systems available that signatures might appear distinctly old fashioned. The best answer probably is: By necessity. Commercial banks still process a huge number of checks, despite all attempts to move to a paperless, electronic environment. And of

course, checks are endorsed by a signature. In some, but certainly not all, banks these signatures are verified electronically.

Signatures on documents such as checks are known as 'static' signatures because the only information one has about the signature is in terms of a static image—the image on the document. There is another possibility and that is of capturing signatures by means of a digitizing tablet. Modern tablets have the ability of capturing the signature as a parametrized curve—the x and y coordinates of the signature, the pen pressure, the pen direction and the pen tilt, all as functions of time. Since the values are sampled at a constant rate, about 125 times a second, it is straightforward to calculate the pen speed or rhythm used to produce the signature. Thus one can think of a signature as an $m \times n$ array S where m denotes the number of features (x– and y coordinates, pen pressure, etc) and n the number of samples. Note that most of this *dynamic* information is hidden from any would-be forger and therefore hard to reproduce. It should be no surprise that dynamic signature verification systems based on digitizing tablets are much more accurate than their static counterparts.

The main difficulty in developing a signature verification system is not hard to guess—one has to distinguish a forgery from the natural variations between genuine signatures. Since we as humans are reasonable good at recognizing the different signatures by the same person, despite the variations[1], Ii is probably not unreasonable to suppose that there is a pattern to the variations—the variations between genuine signatures have structure that one can try and exploit. The only information about this possible structure comes from experience. Any signature verification system, as humans, need to 'see' samples of genuine signatures. We therefore assume that we have access to multiple training examples of each signature. There are different ways in which to use these training sets (one set of signatures for each signatory) to develop an automated verification system. We'll discuss three possible approaches to the problem, in order of increasing sophistication and (hopefully), increasing reliability.

The first step in any automated system is to normalize the signatures with respect to translation, scale and rotation. Normalization with respect to translation and scale is straightforward to achieve. Rotation is a little harder and in our experience a Radon transform-based approach gives reasonable results. In a more straightforward

[1]But also recognizing that we can be mislead by skilled forgeries.

approach one can normalize rotation based on the direction of principle axis. We assume that the normalization has been done.

Note: Every pattern recognition task inevitably involves a certain amount of preprocessing of the data, either to extract relevant features and/or to normalize the data. In the signature verification problem for example, one may add features such as pen speed (easily calculated from the given information) to the list mentioned above.

9.2.1. Clustering. Let us naively decide that that the order of the features is not important, i.e. each sample from the digitizing tablet is considered to be independent from all other samples. If we now collect all the samples of all the signatures in the training set, it is possible to do a clustering where each sample is assigned to a cluster, following either a hard or soft approach. Given a test signature for verification, all the samples of the test signature is then assigned to the different clusters of the signature it is compared against, preferably using a soft approach. The soft approach returns a confidence of the assignment for each sample. These can then be combined to a single confidence measure.

This approach does take some of the structure of the natural variations into account, it fails for the obvious reason that it does not take the statistical dependence of the samples into account—the clustering algorithms assume i.i.d. data.

9.2.2. Dynamic Programming. In the next section we describe an algorithm that allows one to compare different dynamic sequences. Without going into detail, we note that this is a viable approach in many situations. It is particularly effective if one has only a single sample against which to compare. The implication is that it is not really possible to build any sophisticated model of the variations between genuine signatures.

9.2.3. Hidden Markov Models. We are finally ready to build a probabilistic model of the variations between different signatures of the same individual. The HMM achieves this by using an approach somewhat between the clustering and dynamic programming approaches mentioned above. It also divides the samples comprising a single signature and 'cluster' into different 'states'. All the samples within a specific state are assumed to be i.i.d. From the samples assigned to each

state it is possible to estimated a pdf associated with that state. Since the samples are assumed to be i.i.d. one can use any of the techniques discussed earlier in these notes.

So far we have not achieved much. In particular we have not yet taken the dynamics into account, i.e. we have not taken the statistical dependence of the samples into account. This is achieved by introducing probabilistic state transitions, i.e. the transition from one state to the next is also governed by a probabilistic process. The exact sequence of state transitions is not known to the user in advance, it is *hidden* from the user. Since the state transitions are not known, the assignment of samples to states is also unknown and has to be inferred from the data. Not only do we need to estimate the state pdfs, but also the transition probabilities. Before we describe the procedure in detail, we first discuss what has become widely known as the Dynamic Programming (DP) algorithm.

9.3. Dynamic Programming.

In practice we are often faced with a situation where it is necessary to compare different *signals* of different length. For instance, if one is interested in identifying a word or a phrase in a speech recognition application, one realizes that different speakers utter the same phrase in different ways, and with different durations. Comparing these signals is exactly the same problem we face in a signature verification problem. Again we have two similar signals, with differences in details and of different durations to compare. The algorithm we are about to describe was therefore extensive used in speech processing applications until it was largely replaced by even more powerful Hidden Markov Models. It should become clear that there is a straightforward generalization to vector-valued signals; for the moment we keep things as simple as possible and concentrate on single signals. The basic idea is to first define a suitable 'cost function' that gives an indication of the difference (total cost) between the signals. One then proceeds to find a map that maps one signal onto the other in such a way that the cost function is minimized. In this sense one therefore finds the best possible match between the two signals, taking into account that they are usually very different. One can then relate the total 'cost' (difference between the

two signals) to a confidence value—the lower the cost, the higher the confidence that the two signals are actually different renderings of the same entity.

It might be interesting to note that the algorithms is much more general than just for comparing signals. It is a powerful algorithm for calculating maps minimizing an appropriate cost function. It is particularly useful in situations where one has only one example against which to compare, i.e. when it is not possible to build a model from a number of examples (as an additional advantage we can use it as a pro-type for the Viterbi algorithms that we will encounter a little later). For example, the line-break, paragraph-blocking algorithm used by TEX is based on this algorithm. The naïve approach is to estimate the number number of words that will fit into a line, given the page width, and then calculate the inter-word distances so that the line is of exactly the specified length. The problem with this algorithm is that the result does not look good. Sometimes it is better to move a word to the next line for improved overall appearance, etc. Thus the best appearance is achieved by analyzing a whole paragraph and not simply each line separately. The way Knuth [**?**] does this is to define a suitable cost function that is constructed with overall appearance in mind. A mapping is then calculated that fills a whole paragraph in block form.

Let us have a closer look at the problem. In Figure 9.3.1 we show two different signatures by the same person with their y coordinates shown in Figure 9.3.2. It should be clear that the natural way of comparing these signals is to find matching points on the two signatures, such as the peaks indicated in Figure 9.3.2. This clearly requires a nonlinear stretching or 'warping' of the signatures to best fit each other. Mathematically this amounts to a *re-parametrization* of the two signatures.

Given the two discrete signals, $\mathbf{y}_1 := \{y1(t), \quad t = 1, \ldots, L_1\}$ and $\mathbf{y}_2 := \{y_2(s), \quad s = 1, \ldots, L_2\}$, the problem is to find re-parametrizations $p(w)$ and $q(w)$ such that one can identify $y1(p(w))$ with $y_2(q(w)), \quad w = 1, \ldots, L$ in such a way that the difference between the two resulting function is minimized. In order for $p(w)$ and $q(w)$ to be proper re-normalizations they need to be *non-decreasing* functions of w. Since the value of L depends on y_1 and y_2 it is determined as part of the algorithm.

Let us now become very specific and assume that the two signals we want to compare are given by,

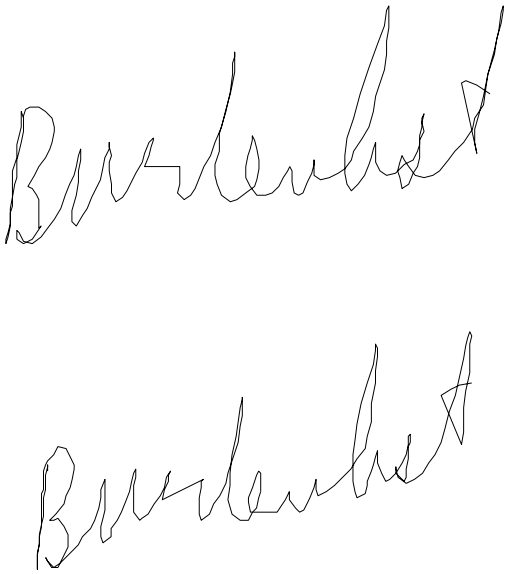


FIGURE 9.3.1. Two different signatures by the same signatory for comparison.

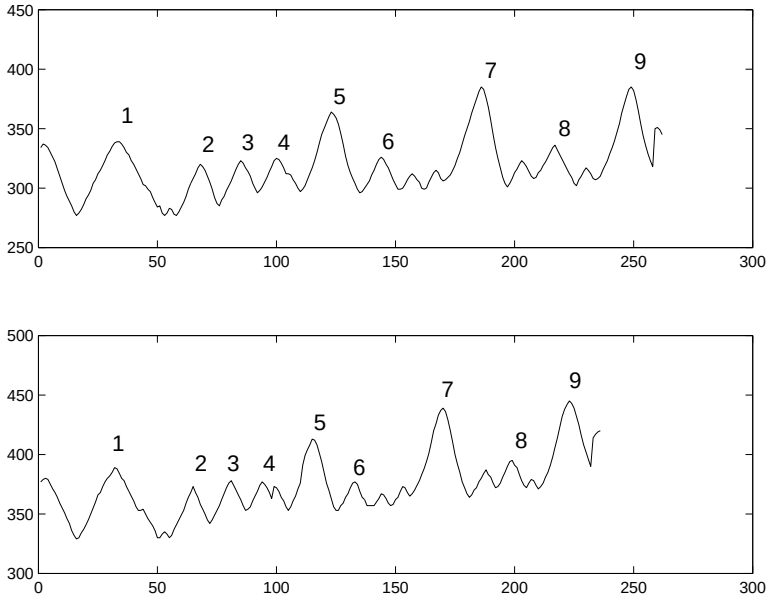


FIGURE 9.3.2. The y coordinates of the two signatures.

$$\begin{aligned} \mathbf{y}_1 &= [1\ \ 0\ \ 1\ \ 2\ \ 1] \\ \mathbf{y}_2 &= [1\ \ 0\ \ 2\ \ 1]. \end{aligned}$$

Since we want to compare each value of $\mathbf{y}_1$ with each value of $\mathbf{y}_2$ it is convenient to draw a grid as shown in Figure 9.3.3. If we now draw a curve consisting of straight lines connecting the lower left-hand corner with the upper right-hand corner (the reader may wish to have a peak at Figure 9.3.7) below, any such curve defines a mapping between the two signals. It is important to note however that the monotonicity constraint on $p(w)$ and $q(w)$ implies that grid-point (i, j) can only be reached from either $(i-1, j)$, $(i, j-1)$ or $(i-1, j-1)$, as indicated (if there is a tie, we give preference to the diagonal). It is this curve that we are after, constructed in such a way that the two signals are aligned in the best possible way (in a sense that has to be made precise). Also note that our assumption that the curve begins and ends at the two corners implies that we match the start– and endpoints of the two signals. In some applications it is useful to relax this constraint—it is absolutely straightforward to relax the endpoint matching constraint, as will become clear.

The brute-force way of solving the problem is to investigate all possible paths connecting the lower left-hand corner with the upper right-hand corner. This is a use number and the reader may find it a fun exercise to derive the following formula for the total number of possible paths

$$\mathcal{N} = \sum_{s=0}^{\min(L_1,L_2)} \frac{(L_1 + L_2 - s)!}{(L_1 - s)!(L_2 - s)!s!},$$

constrained only by our monotonicity requirement. This is an enormous number, dominated by the first term which counts only the number of paths not containing a diagonal,

$$(L_1 + L_2)!/L_1!L_2!.$$

Even this is too large to investigate exhaustively by computer for all but the smallest number of samples. We clearly need to do better.

All the most efficient algorithms are based on a very simple observation (which the reader can prove for herself): Each sub-path of an optimal path, is also optimal. This means that the global optimal path is pieced together from local optimal paths—exactly the defining strategy of Dynamic Programming (DP). The key is to define

a local distance measure (local cost function) that measures the difference between the two functions. An obvious choice is

$$C_{i,j} = d(y1(i), y2(j)) := |y_1(i) - y_2(j)|, \quad i = 1, \ldots, L_1; \quad j = 1, \ldots, L_2. \tag{9.1}$$

For our example, this is written as

$$C = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 \\ 1 & 2 & 1 & 0 & 1 \\ 1 & 0 & 1 & 2 & 1 \\ 0 & 1 & 0 & 1 & 0 \end{bmatrix},$$

where one should note that the ordering corresponds to the ordering of the grid of Figure 9.3.4. Also note that in Figure 9.3.4 that it is convenient for us to number the grid points consecutively from 1 to 20. For example, grid point 9 has coordinates $(4, 2)$ and its local distance value is $C_{4,2} = 2$. Based on this local distance measure, the total cost function for the two signals $y_1(t)$ becomes

$$C = \sum_{w=1}^{L} |y_1(p(w)) - y_2(q(w))| . \tag{9.2}$$

Once the the local costs have been calculated, the rest of the algorithm proceeds without any reference to the original signals—it allows us to calculate the optimal path from the lower left-hand corner to every other point on the grid. This may sound wasteful but with the slight modification explained below, it is still the most efficient algorithm known. The result is shown in Figure 9.3.5. The Figure shows the optimal path to each grid point with the total cost of each path. For example, the cost of reaching point 19 with coordinates $(4, 4)$, is 2. In order find its optimal path, we start at point and then backtrack until we reach point 1. Note that all we need is to know from where a specific point is reached, i.e. point 19 is reached from point 13, is reached from point 7, etc. Accordingly, we keep track of all the different paths, by defining an array $I(i), i = 1, \ldots, L_1 L_2$ where $I(k)$ denotes the point from which point k is reached. For our example, Figure 9.3.5, we find that $I(20) = 14, I(19) = 13$, etc. Thus the optimal path is obtained from $I(20) = 14,\ I(14) = 8,\ I(8) = 7,\ I(7) = 1$.

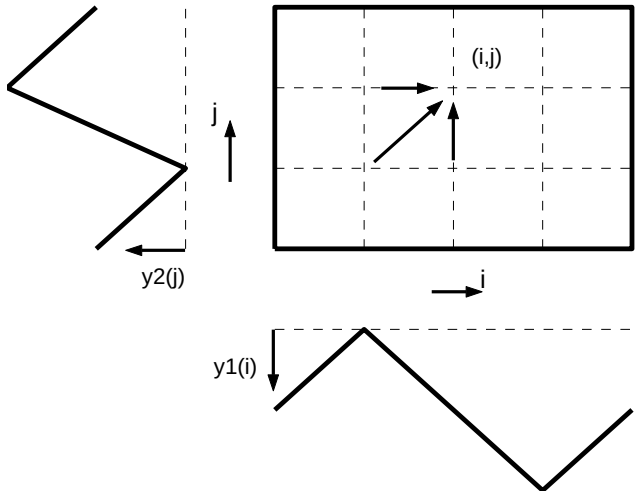


FIGURE 9.3.3. The two signals to be compared.

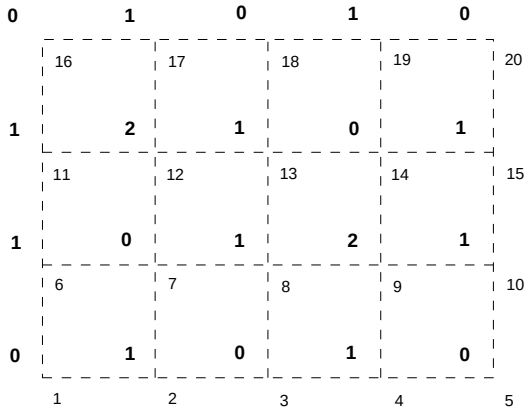


FIGURE 9.3.4. The local cost grid.

Let us proceed with the description of the algorithm. Since the local cost for point 1 is zero (see Figure 9.3.4), the total cost to reach grid point 1 is $TC(1) = C_1 1 = 0$. Noting that point 2 can only be reached from point 1, there is no decision to make and the total cost of reaching point 2 is: $TC(2) = TC(1) + C_{21} = 1$. We also record that it is reached via the optimal (and only) path from point 1, i.e. we set $I(2) = 1$. For points 3, 4, 5 and 6 we do the same, $TC(3) = TC(2) + C_{31} = 1$, $I(3) = 2$, $TC(4 = TC(3 + C_{41} = 2$, $I(4) = 3$, $TC(5 = TC(4 + C_{51} = 2$, $I(5) =$

4, $TC(6 = TC(1 + C_{12} = 1, \quad I(6) = 1$. However, point 7 may be reached from either points 1, 2 or 6. We already know the optimal paths to points 1, 2 and 6. Since the cost of the path to point 1, is less or equal to the cost to points 2 or 6, we reach point 7 via point 1 and set $TC(7) = TC(1) + C_{22} = 0, \quad I(7) = 1$. Now we do the same for the rest of the points on the grid:

$$\begin{aligned}
TC(8) &= TC(7) + C_{32} = 1, \quad I(8) = 7 \\
TC(9) &= TC(3) + C_{42} = 3, \quad I(9) = 3 \\
TC(10) &= TC(4) + C_{52} = 3, \quad I(10) = 4 \\
TC(11) &= TC(6) + C_{13} = 2, \quad I(11) = 6 \\
TC(12) &= TC(6) + C_{23} = 2, \quad I(12) = 6 \\
TC(13) &= TC(7) + C_{33} = 1, \quad I(13) = 7 \\
TC(14) &= TC(8) + C_{43} = 1, \quad I(14) = 8 \\
TC(15) &= TC(14) + C_{53} = 2, \quad I(15) = 14 \\
TC(16) &= TC(11) + C_{14} = 2, \quad I(16) = 11 \\
TC(17) &= TC(11) + C_{24} = 3, \quad I(17) = 11 \\
TC(18) &= TC(13) + C_{34} = 1, \quad I(18) = 13 \\
TC(19) &= TC(13) + C_{44} = 2, \quad I(19) = 13 \\
TC(20) &= TC(14) + C_{54} = 1, \quad I(20) = 14.
\end{aligned}$$

With the help of I it is now a simple matter of finding the optimal path as pointed out above, and shown in Figure 9.3.6, with its total cost $TC(20) = 1$.

The computational cost of this algorithm amounts to three tests at each grid point, i.e. it is of $O(L_1 L_2)$. This can be further reduced by realizing that the optimal path should not stray too far from the diagonal, at least not for similar signals. One can therefore restrict the search to a band around the diagonal, further reducing the computational cost. If we restrict the search to a band of width d, the computational cost is of $O(L_1)$ with a possibly large constant, depending on d (assuming $L_1 \geq L_2$).

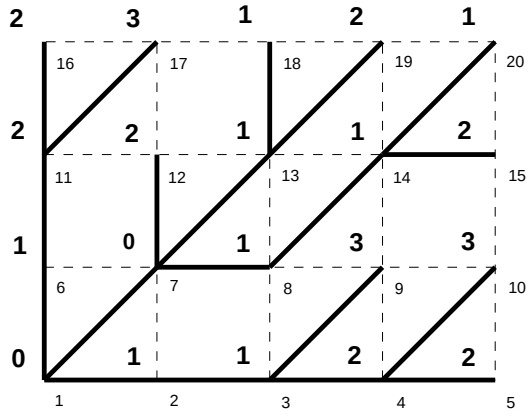


FIGURE 9.3.5. The optimal paths to all the grid points.

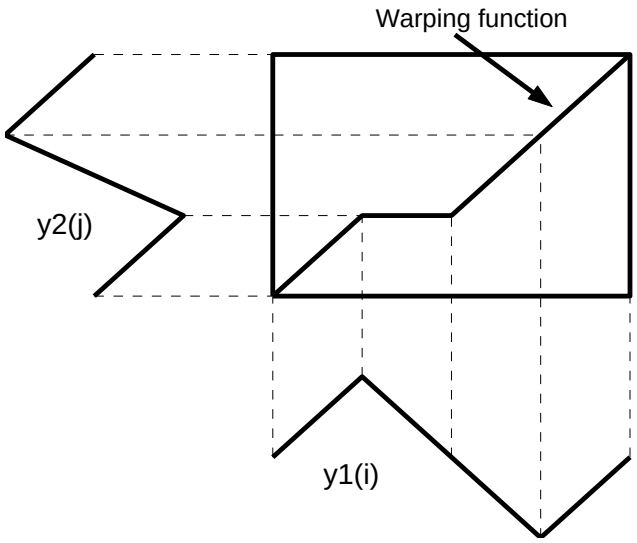
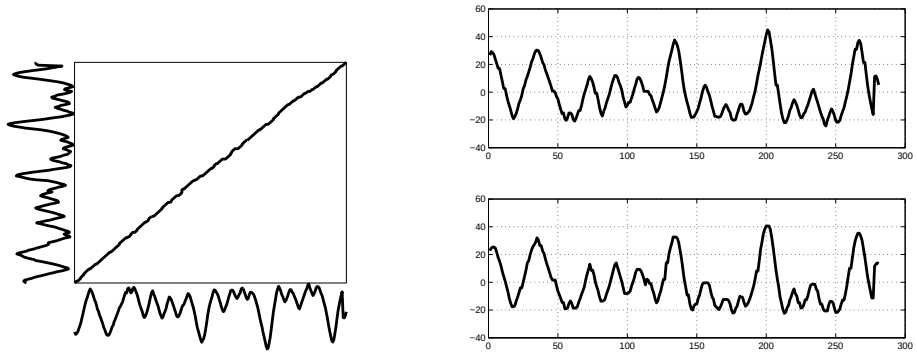


FIGURE 9.3.6. The optimal path.

Figure 9.3.7(a) shows the matching of the y coordinates of different signatures belonging to the same person. It is interesting to note how close the warping function remains to the diagonal, an indication that there is a good correspondence between the two functions. If one now plots y_1 and y_2 against their re-parametrizes indexes,

Figure 9.3.7(b) shows how the different peaks are now perfectly aligned. Incidentally, the total cost in this case is 374.1.



(a) (b)

FIGURE 9.3.7. Matching two similar signatures.(a) The warping path. (b) The two signals aligned.

Let us now do the same thing for two completely different signatures as shown in Figure 9.3.8. One may think of the second signature as a *casual* forgery since the forger clearly had no idea what the original looked like. The point is that the algorithm still finds the best possible match, which in this case is not good at all.

The y coordinates, their warping function as well as their best possible alignment are shown in Figure 9.3.9. Although the algorithm still finds the best match, the warping function deviates considerably from the diagonal. The big difference between the two signatures is reflected by the large value of its total cost function, 2 885.

9.4. Basic concepts and notation

In the naive Bayes approach we modeled a sequence of T feature vectors (or observations) $\mathbf{x_1^T}$ as $p(\mathbf{x}_1^T|M) = \prod_{t=1}^{T} p(\mathbf{x}_t|M)$[2]. We got from the joint density (pdf) to the product of individual densities by making a very strong assumption, namely

[2]Here we use the "$|M$" to indicate that the calculation is being done using a specific set of model parameters. When it is clear from the context that we are referring to a specific model we may omit this for the sake of simplicity.

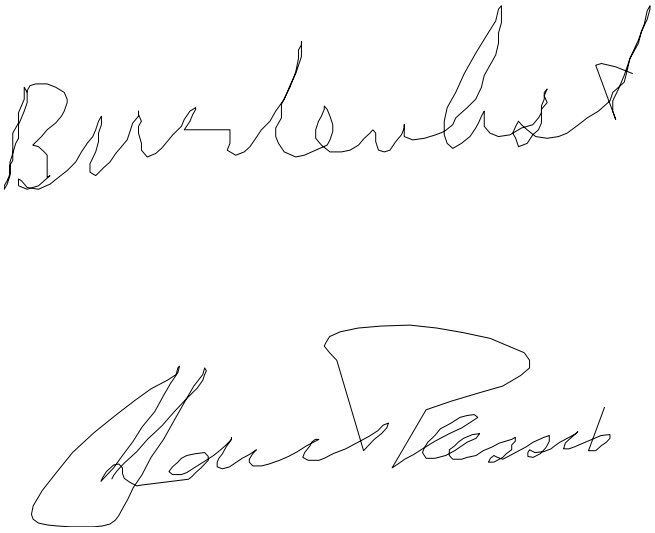
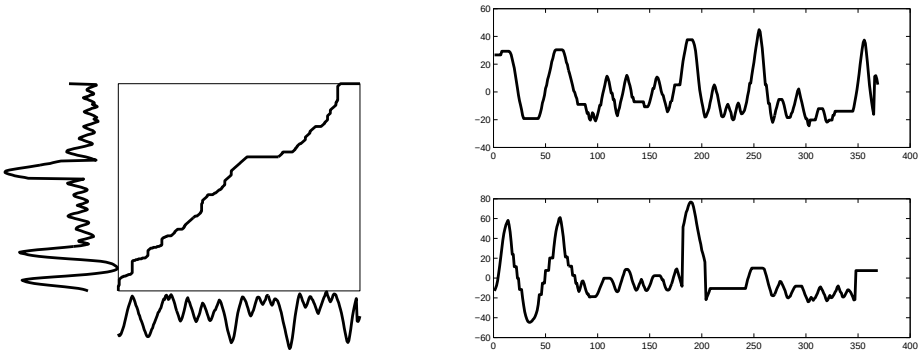


FIGURE 9.3.8. Two unrelated signatures.



(a) (b)

FIGURE 9.3.9. Matching two unrelated signatures.(a) The warping path. (b) The two signals aligned.

that each feature vector $\mathbf{x}_t$is statistically independent from every other one, while also sharing a common marginal pdf with all of them. This immediately destroys any hope that this model will be able to capture some time-dependent behavior between the various $\mathbf{x}_t$'s, in fact jumbling them into any arbitrary time order will not affect the resulting pdf value at all.

9.4.1. Emitting states. We now want to make less drastic assumptions that will allow us to model time-dependent behavior whilst at the same time keeping computations tractable. HMMs do this by introducing the concept that the model is in 1 of N states at any given time. However, we typically do *not* know in which state we are at such a given time - hence the name *hidden* Markov model. Each state $s = i$, $i = 1, \ldots, N$ on its own behaves very much like the naive Bayes / i.i.d. model we previously encountered. It has a pdf describing the feature vectors associated with it, i.e. $p(\mathbf{x}|s)$. In this way one can think of the naive Bayes model as a 1 state HMM, or alternatively of an HMM as a series of naive Bayes models with probabilistic jumps between them. We will need to refer generically to the state that is active at time t, we will use the notation s_t for this.

9.4.2. Transitions. These states are coupled to each other with transition probabilities $a_{i,j} = P(s_{t+1} = j|s_t = i)$. In other words, it indicates the probability that we will transit to state j at the next time step *if* we knew that we are in state i at the current time step (which we of course unfortunately do not know). Of course, once one has made a transition to a new state, a new pdf apply, thereby giving the HMM the ability to model temporal patterns. This basically allows us to change the model behavior as a function of time. These probabilities are collected in a transition matrix A, its row indexes indicating the source state and the columns indicating the destination state of the transition. Since all the probabilities departing from a given state must sum to one, each row in this matrix will sum to one.

9.4.3. Non-emitting/null states. In addition to the above we also need to be able to specify the states in which our model can start (the initial states) and those in which it can terminate. We do this by adding two special states without any densities, namely state 0 and state $N+1$[3] These states are called null or non-emitting states to distinguish them from the normal/emitting states we encountered above [4]. State 0 will have no links entering it, and state $N+1$ will have no links leaving from it. We require that the process always start in state 0 and terminate in state $N+1$.

[3]It is also possible and useful to use such null states in other places in the model. Since it complicates algorithms we will here refrain from this.

[4]Our notation deviates here from the common convention to use a vector π to indicate initial states, and possibly another vector $\mathbf{F}$ to indicate final/terminating states.

As can be seen from figure 9.4.1 this allows for arbitrary initial and terminal states. We extend the s_t notation to allow for these states by also including a s_0 and s_{T+1}. Note, however, that the transition to the final state does not occupy an extra time step since it has no pdf. Therefore s_{T+1} actually is active at the same time as s_T.

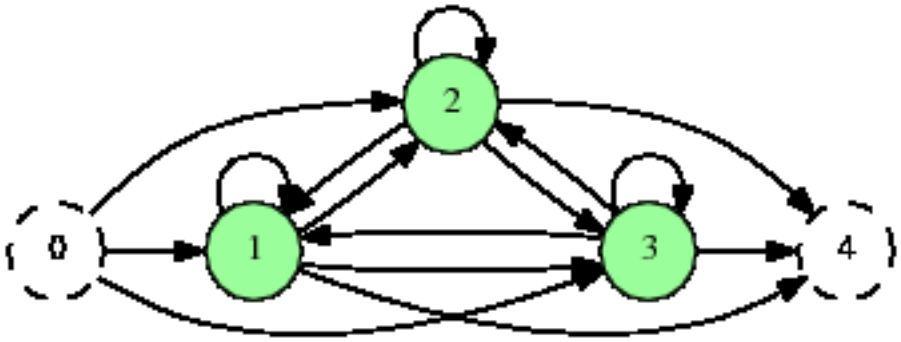


FIGURE 9.4.1. A simple fully connected HMM

9.4.4. Fundamental HMM assumptions. The above model makes two important assumptions about the relationship between the feature vectors:

(1) The observation independence assumption states that:

$$p(\mathbf{x}_t|\mathbf{x}_1^{t-1}, s_0^t) = p(\mathbf{x}_t|s_t). \tag{9.1}$$

This means that the likelihood of the t'th feature vector is dependent only on the current state and is therefore otherwise unaffected by previous states and feature vectors. This assumption is not affected by the order of the HMM.

(2) The first-order[5] Markov assumption:

$$P(s_t|s_0^{t-1}, \mathbf{x}_1^{t-1}) = P(s_t|s_{t-1}). \tag{9.2}$$

This states that, apart from the immediately preceding state, no other previously observed states or features affect the probability of occurrence of the next state.

[5]This can also be generalized to (more powerful) higher Markov orders.

In addition to the above the reader will note that our notation $a_{i,j}$ for transition probabilities do not allow them to be time dependent. We assume the transition probability between two states to be constant irrespective of the time when the transition actually takes place.

9.4.5. A few basic HMM topologies. Topology refers to which states are connected to each other. Many configurations are popular, it usually makes sense to carefully match the topology to the characteristics of the observations that are being modeled. The most generic version is the fully connected topology shown in Figure 9.4.1. It is often used in applications where observations repeats over time (for example a text-independent speaker recognition system).

When the observations has a well-defined sequential nature, such as encountered in for instance word recognition, a left-to-right form such as shown in figure 9.4.2 may be more appropriate.

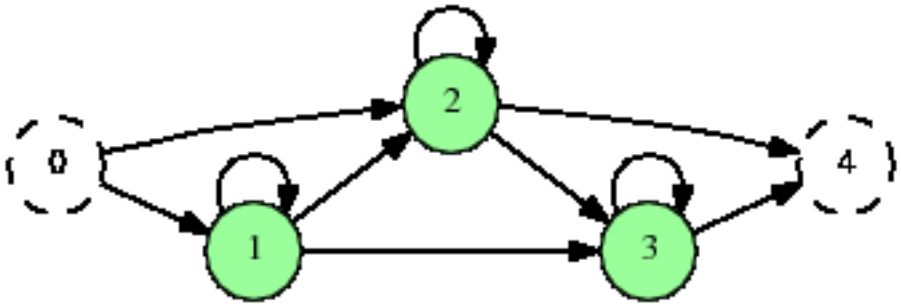


FIGURE 9.4.2. A simple left-to-right HMM

As said, many other topologies are useful, this is to be discussed at a later stage.

9.5. Calculating $p(\mathbf{x}_1^T|M)$.

In the following we are going to repeatedly use two results from basic probability theory namely conditional probability (the product rule)

$$p(a, b|c) = p(a|b, c)p(b|c) \tag{9.1}$$

and total probability (the sum rule)

$$p(a) = \sum_i p(a, b_i) \tag{9.2}$$

where b_i forms a partition. For simplicity we will (mostly) also omit the reference to the model M.

9.5.1. A direct approach. We can then write the required probability as:

$$p(\mathbf{x}_1^T) = \sum_{\forall s_0^{T+1}} p(\mathbf{x}_1^T, s_0^{T+1}) \qquad \text{(sum rule)} \tag{9.3}$$

But

$$\begin{aligned} p(\mathbf{x}_1^T, s_0^{T+1}) &= P(s_{T+1}|s_0^T \mathbf{x}_1^T) p(\mathbf{x}_1^T, s_0^T) \qquad \text{(prod. rule))} \\ &= a_{s_T,N+1} p(\mathbf{x}_T|\mathbf{x}_1^{T-1}, s_0^T) p(\mathbf{x}_1^{T-1}, s_0^T) \qquad \text{(Markov ass., prod. rule)} \\ &= a_{s_T,N+1} p(\mathbf{x}_T|s_T) p(\mathbf{x}_1^{T-1}, s_0^T) \qquad \text{(observ. indep.)} \end{aligned}$$

The last term on the right hand side is exactly of the same form as the left hand side, therefore we can recursively complete the evaluation to yield:

$$\begin{aligned} p(\mathbf{x}_1^T, s_0^T) &= a_{s_T,N+1} p(\mathbf{x}_T|s_T) a_{s_{T-1},s_T} p(\mathbf{x}_{T-1}|s_{T-1}) \times \\ &\quad a_{s_1,s_2} p(\mathbf{x}_1|s_1) P(s_1|s_0) P(s_0) \\ &= a_{s_T,N+1} p(\mathbf{x}_T|s_T) a_{s_{T-1},s_T} p(\mathbf{x}_{T-1}|s_{T-1}) \times \\ &\quad a_{s_1,s_2} p(\mathbf{x}_1|s_1) a_{0,s_1} \end{aligned} \tag{9.4}$$

Since all the values on the right hand side are either known, or can be readily calculated, it would seem that we have succeeded in providing an approach towards calculating equation 9.3. The snag lies in the $\forall s_0^{T+1}$. In a fully connected model with N states and T time steps the number of possible state sequences is N^T —which very rapidly becomes prohibitively large[6]. We need to find another method that can do this more efficiently.

[6] with $N = 500$ and $T = 300$ we already have approximately 10^{800} paths!

9.5.2. The forward algorithm. Let us consider the calculation of the so-called forward likelihoods:

$$\begin{aligned}
\alpha_t(j) &= p(\mathbf{x}_1^t, s_t = j) \qquad (\text{with } t = 1, \ldots, T \text{ and } j = 1, \ldots, N) \\
&= p(\mathbf{x}_t|\mathbf{x}_1^{t-1}, s_t = j)p(\mathbf{x}_1^{t-1}, s_t = j) \qquad (\text{prod. rule}) \\
&= p(\mathbf{x}_t|s_t = j)\sum_{i=0}^{N} p(\mathbf{x}_1^{t-1}, s_{t-1} = i, s_t = j) \qquad (\text{observ. indep., sum rule}) \\
&= p(\mathbf{x}_t|s_t = j)\sum_{i=0}^{N} P(s_t = j|s_{t-1} = i, \mathbf{x}_1^{t-1})p(\mathbf{x}_1^{t-1}, s_{t-1} = i) \qquad (\text{product rule})
\end{aligned}$$

Recognizing that the last term on the right hand side is $\alpha_{t-1}(i)$, as well as using the Markov assumption this reduces to a very usable recursive form,

$$\alpha_t(j) = p(\mathbf{x}_t|s_t = j)\sum_{i=0}^{N} a_{i,j}\alpha_{t-1}(i),$$

or if we want to state it more generally to include state 0:

$$\alpha_t(j) = \begin{cases} p(\mathbf{x}_t|s_t = j)\sum_{i=0}^{N} a_{i,j}\alpha_{t-1}(i) & \text{with } t = 1, \ldots, T,\ j = 1, \ldots, N \\ 0 & \text{with } t = 1, \ldots, N,\ j = 0. \end{cases} \tag{9.5}$$

To start the recursion at $t = 0$ we need $\alpha_0(j) = P(s_0 = j)$,

$$\alpha_0(j) = \begin{cases} 1 & \text{with } j = 0 \\ 0 & \text{with } j > 0. \end{cases} \tag{9.6}$$

The total likelihood of the data given the model fits nicely into this framework,

$$
\begin{aligned}
p(\mathbf{x}_1^T|M) &= \sum_{j=0}^{N} p(\mathbf{x}_1^T,\ s_T = j,\ s_{T+1} = N+1) \qquad \text{(sum over all valid terminations)} \\
&= \sum_{j=0}^{N} P(s_{T+1} = N+1|s_T = j, \mathbf{x}_1^T) p(s_T = j, \mathbf{x}_1^T) \qquad \text{(prod. rule)} \\
&= \sum_{j=0}^{N} a_{j,N+1} \alpha_T(j) \qquad \text{(Markov ass.)} \ . \qquad (9.7)
\end{aligned}
$$

In a fully connected model with N emitting states and T time steps the number of calculations is now $O(N^2T)$, which is very do-able indeed.

9.5.3. Preventing numerical over-/underflow. Inspecting (9.5) reveals repeated multiplication of quantities that are either probabilities and therefore in the range $(0, 1)$, or high-dimensional pdf values that, although they could have any positive value, are likely to be very small. In practical terms the iterative calculation of the above α's *will* underflow and we need to take precautions against this. Two approaches are popular:

- Rescale the α's: After all the $\alpha_t(j)$'s at a specific time t have been calculated, divide them by their sum $\sum_j \alpha_t(j)$ while also recording this sum in log format in a separate variable. Accumulating these log-sums over time will yield the exact factor which should be added to the calculated/scaled $\log p(\mathbf{x}_1^T)$ to yield its correct value.
- Work with $\log \alpha_t(j)$ throughout: Working in log format should prevent any underflow problems. However, now we need to be careful with the summation in (9.5) which still needs to be done in the linear domain. Directly converting these log values to linear before summation will immediately reintroduce the underflow problems we are trying to prevent. We come up against expressions such as $L = \log(e^{L_1} + e^{L_2} + ... + e^{L_M} + ... + e^{L_N})$ where none of the individual terms e^{L_n} are expressible in linear form. This is not as daunting as it might seem at first and can be calculated as $L = L_M + \log(e^{L_1 - L_M} + e^{L_2 - L_M} + ... + 1 + ... + e^{L_N - L_M})$ where $L_M = \max(L_1, L_2, ...L_N)$.

9.6. Calculating the most likely state sequence: The Viterbi algorithm

We see in (9.3) that the total likelihood of the data involves a sum of all possible state sequences s_0^{T+1}. Due to the "hiddenness" of the HMM, we never know with certainty which of those many state sequences actually gives rise to our observed feature vectors $\mathbf{x}_1^T$. At least one of them will provides the biggest contribution to this sum. This is the most likely state sequence. We can find it easily with a small modification to (9.5) and (9.7). If we replace the summations there with maximizations, while also recording which specific previous state resulted in each such a maximum, we have calculated $p(\mathbf{x}_1^T, S^*)$ where S^* denotes the most likely state sequence. This most likely state sequence can be recovered by recursively backtracking from state $N+1$ to its most likely predecessor, until we reach state 0. It is left as an exercise for the reader to determine why this procedure results in the optimal state sequence (hint: the above is a dynamic programming algorithm). Once again one has to give consideration to possible underflow problems. However, by using these Viterbi maximizations, the total calculation has reverted to a single sequence of multiplications, exactly as we found in (9.4). It is particularly simple to work in the log domain—the sequence of multiplications simply changes to additions.

9.7. Training/estimating HMM parameters

The GMM we encountered in a previous chapter, weighed and combined a collection of observation pdfs. Because we did not know to which one of the mixture components a specific feature vector should be assigned, i.e. there was missing/latent information, we had no closed form solution for estimating its parameters. We had to resort to the EM algorithm to supply an interactively improving estimate. This estimate converged only to a local optimal set of parameters, there is no guarantee that the solution is a global optimum.

The HMM also combines a collection of observation pdfs. In this case the situation is more complex since the order of the observations is also important. Not only do we need to estimate the pdfs, we also need to estimate the transition probabilities. Once again there is missing information—we do not know to which state a specific feature vector should be assigned. Let us pause for a moment and suppose that somehow we are able to assign each feature vector to a a specific state. All

of a sudden it is entirely straightforward, at least in principle, to estimate both the state pdfs and the transition probabilities. Collecting all the features assigned to a specific state, those feature vectors are used to estimate the state pdf. Note that once we have identified the features belonging to a specific state, in the estimation of the state pdf we assume that its feature vectors are i.i.d., i.e. we can for instance use maximum likelihood to estimate the parameters of Gaussian densities, or the EM algorithm described in Section 7.3, if the state pdf happens to be a GMM. The transition probabilities are obtained by noting the state transitions. Since the features vectors are ordered, successive features will be assigned to states $s_t = i$ and $s_{t+1} = j$. By counting the relative number of state transitions from $s_t = i$ to $s_{t+1} = j$, we arrive at a estimate of the transition probability a_{ij}.

The description above should be familiar, it is reminiscent of the *EM* algorithm we derived for the estimating HMMs in Section 7.3. Thus we find that the HMM parameters can also be estimated via the EM algorithm. Once again the solution is only guaranteed a local optimum.

9.7.1. Knowing the (hidden) state sequence. Suppose for the moment that for each sequence of training observations/feature vectors $\mathbf{x}_1^T$ we somehow knew the corresponding state sequence s_1^T, i.e. each $\mathbf{x}_t$ is assigned to a specific state $s_t = i$. As pointed out above, this allows a straightforward estimation of the model parameters.

- **Transition probabilities:** For the transition probabilities a_{ij} we simply count the relative number of state transitions from $s_t = i$ to $s_{t+1} = j$,

$$\hat{a}_{ij} = \frac{\#(s_{t+1} = j | s_t = i)}{\#(s_t = i)}, \tag{9.1}$$

  where we use $\hat{}$ to indicate an estimate, and $\#$ to indicate the count of the number of occurrences of its argument. Note that the normalizing denominator ensures that $\sum_j a_{ij} = 1$.
- **Observation pdfs:** Similarly we calculate the parameters of the state pdfs $p(\mathbf{x}|s = i)$ by using all the feature vectors associated with state $s = i$. As mentioned above, the fact that the features vectors assigned to state $s = i$ are assumed to be i.i.d., the techniques described earlier in these notes can

be used to estimate the pdfs. The details depend on the specific pdf being used, it can be as simple as calculating the mean and variance vectors of a diagonal Gaussian[7].

Since the transition probabilities and state pdfs comprises our full HMM, it therefore is quite simple to estimate the model parameters *if* we know the (hidden/latent) state sequences. Unfortunately it is unknown. *If*, on the other hand, we somehow knew the HMM parameters, we can use the Viterbi algorithm to estimate the optimal state sequence S^* for each observation sequence $\mathbf{x}_1^T$. The Viterbi estimate is a substitute for the true state sequence. So we have a chicken-egg situation here. With known state sequences we can calculate the model parameters and with known model parameters we can calculate optimal state sequences. But unfortunately both are actually unknown. You may have guessed already, this is allows the use of the EM algorithm, now in the guise of the Viterbi Re-estimation algorithm.

9.7.2. Viterbi Re-estimation.

(1) **Initialization:**

(a) For every training observation sequence $\mathbf{x}_1^T$, assign in a sensible manner a corresponding state sequence s_1^T and extend it to s_0^{T+1} by adding the initial and termination states. This "sensible" manner depends on the desired topology. For a fully connected model clustering might provide initial labels, for left-to-right models each training sequence can be subdivide in N equal portions[8].

(b) From this initial state sequence, generate an initial model as discussed in section 9.7.1.

(2) **EM Re-estimation:**

(a) **Expectation step:** For the current model estimate, apply the Viterbi algorithm on every training sequence $\mathbf{x}_1^T$, to calculate $\log p(\mathbf{x}_1^T, S^*)$ where S^* is the expected state sequence for this observation sequence estimated by the Viterbi algorithm. Accumulate the "scores" i.e. $f = \sum_{\forall \text{ training } \mathbf{x}_1^T} \log p(\mathbf{x}_1^T, S^*)$, to be used later to test for convergence.

[7]It is quite common for the pdf to be a GMM.

[8]A random state assignment typically ultimately results in an inferior local optimum and is not recommended.

(b) **Maximization step:** Use *all* the training sequences, each with its Viterbi state sequence estimate S^* as obtained in the E-step to update the parameters of the HMM as discussed in section 9.7.1.

(3) **Termination:** Compare the total score f obtained in the E-step with that obtained from the previous E-step. If it is within an acceptable tolerance, terminate, otherwise continue with the re-estimation (i.e. step 2).

9.7.3. Baum Welch Re-estimation. The reader might recall that in the re-estimation of GMM's, each feature vector $\mathbf{x}$ was assigned a responsibility for each mixture component, i.e. each feature vector was given a soft assignment to each mixture component. The Viterbi algorithm, on the other hand assigns each feature vector to a specific state, i.e. it uses a hard assignment, along the lines of what happens with the $K-$means algorithm.

For all three training scenarios—$K-$means, GMM, and HMM— we actually have a choice whether we want to use the "hard" approach which associates each feature vector with only one missing label namely the most probable one, or the "soft" approach which generalizes this by probabilistically/partially associating the feature vector with multiple labels according to our probabilistic knowledge of the situation.

The "soft" version of the Viterbi re-estimation is called Baum Welch re-estimation. To do this we use the so-called "backward algorithm" to calculate another set of likelihoods (β's) related to the α's of the forward algorithm. These are then combined with the α's to yield state and transition *probabilities*, in contrast to only the optimal states and transitions resulting from the Viterbi algorithm. Performance-wise this is a slightly better approach. In practical terms, however, the accuracy of these algorithms are very similar. We are not going to investigate this further here, and refer the reader to the literature.