Chapter 4

Decisions: Statistical methods

In the last chapter I took a somewhat jaundiced look at the raw ideas of modern statistics and probability theory; I can only hope that the thoughtful reader is still with us and has not abandoned the entire enterprise in disgust. In that chapter, I drank deep of Philosophical matters to an extent which can drive many an engineer bananas. Some people like that kind of thing, but it doesn't go with the hard headed attitude of those who want to get on with programming their robot to distinguish between eggshells and the best china. The trouble is, that it is possible to sell a kind of intellectual snake oil to such engineers, as the existence of all those books on Fuzzy Set Theory makes clear, so those engineers should start getting more sceptical. Looking critically at the relatively honest and only faintly crackpot ideas of the probabilists is a good start. What looks like a promising avenue of enquiry to some can look to others like the ravings of a glue-sniffing, born-again fruit bat. While, in the fullness of time, truth will out and either your programs will work or they won't, it can take a long time to get to this happy state, and some critical thought about what kind of methods merit consideration can save a lot of effort.

In other words, I am only mildly apologetic. If that.

In this chapter I shall give some statistical methods for coping with the situation where some measurements have been made, the data consists of some collection of points in $\mathbb{R}^n$, each point labelled as belonging to some finite set of categories, and the problem is to decide to which category a new, as yet unlabelled, point belongs. If the attempt looks to be a desperate one, fraught with risk and uncertainty, and if the justifications offered seem to be shonky and unconvincing, we can only say with Bruce Fairbairn's famous character, 'if you know a better 'ole, go to it.' The reader who still has his feet up and is getting slightly sloshed would do well to sit up straight and drink some coffee.

First an important principle in any branch of Science or Engineering: eyeball the data. Look at it, run it through your fingers, get a feel for it. Learn to know its little vagaries. In these days of instant packages which allow you to get out results using methods you don't understand, implemented by people you don't know, on equipment you haven't tested, it is very easy to produce totally mindless garbage, orders of magnitude faster and stupider than at any time in history. Have no part of this. Mindless stupidity is always popular with the mindlessly stupid, but it doesn't cut the mustard. So, I repeat, **EYEBALL THE DATA!** In the case of data in dimension 12, say, this is not so easy as if the dimension is 2, when you can plot O's and X's as I did in chapter one. But given the powers of the computer it is not much harder to project it from dimension n onto the screen of a computer, and to spin it and look at it from several angles. So we start off with a painless bit of linear algebra.

4.1 The view into $\mathbb{R}^n$

Suppose you have a few hundred vectors of two categories, each vector of dimension 12. It may be possible to look at the column of numbers and tell by eye which is which. If the points are well separated this can sometimes be done. This primitive and basic eyeballing of the data is well worth trying. Sometimes it can show that you have done something daft, such as having a zero vector in both categories, which has been known to be produced by a program bug.

If the number of categories is bigger than two, or the data less well separated, looking at the columns of numbers may be uninformative. But just as a photograph of, say, a tree is a projection of a three dimensional object onto two dimensions, so it is possible, indeed easy, to project from $\mathbb{R}^n$ down to $\mathbb{R}^2$ for any n. And to get from any bounded piece of $\mathbb{R}^2$ to the computer screen is just shrinking or stretching and shifting.

Any view of a piece of $\mathbb{R}^3$ will leave somethings hidden behind others and leave it unclear as to the relative distances, but doing any rotation about almost any axis will show us the relative spacings and reveal what was hidden. If we can rotate $\mathbb{R}^n$ and look at the resulting data from different perspectives, we can sometimes see some useful structure in the data.

Suppose we have two orthogonal vectors $\mathbf{u}, \mathbf{v}$ in $\mathbb{R}^n$ of length 1. Then if $\mathbf{x}$ is a point in $\mathbb{R}^n$ and $\cdot$ is the usual dot product, the projection of $\mathbf{x}$ onto the plane spanned by $\mathbf{u}, \mathbf{v}$ is $(\mathbf{x} \cdot \mathbf{u})\mathbf{u} + (\mathbf{x} \cdot \mathbf{v})\mathbf{v}$. So by taking the two numbers $\mathbf{x} \cdot \mathbf{u}, \mathbf{x} \cdot \mathbf{v}$ we map the point $\mathbf{x}$ into $\mathbb{R}^2$. It is then straightforward to plot these points on the computer screen, and one of the programs on disk does this for you.

Rotations in $\mathbb{R}^n$ are most easily thought about as composites of rotations in a choice of plane. The three dimensional case is rather special, since in choosing a plane to rotate in $\mathbb{R}^3$ we are also choosing an orthogonal line which is fixed. In $\mathbb{R}^4$, we can leave a two dimensional subspace fixed, we rotate about a plane. So it is simpler to stop thinking in terms of axes of rotation and to concentrate on the plane that is moved into itself. It is easiest to take these to be defined by some choice of two different axes.

We therefore obtain a basic rotation in $\mathbb{R}^n$ by choosing distinct i and j between 1 and n. We write down the identity n by n matrix, and change the locations $(i,i),(i,j),(j,i)$ and (j,j). In the $(i,i),(j,j)$ locations we write $cos(\theta)$, for some choice of θ, in location (i,j) we write $sin(\theta)$ and in location (j,i) we write $-sin(\theta)$ for the same choice of θ. The new matrix rotates $\mathbb{R}^n$ by an angle θ in the (i,j) plane.

By making any choices of i,j and θ, we get different rotations. By composing one after another we can get any special orthogonal matrix.

Initialising the vectors $\mathbf{u}$ and $\mathbf{v}$ to being, say, the first two elements in the standard basis for $\mathbb{R}^n$, and then applying a sequence of rotations, we can view the points of the data set from a variety of angles. We regard our two 'window vectors' as the things to be operated on rather than the data points, as this is a lot quicker.

There are some obvious modifications to the above algorithm, such as scaling (zooming in or out) and shifts in the space. After a large number of rotations, it may happen that the window vectors get deformed out of true: they may no longer be orthonormal as a result of the accumulation of rounding errors in the matrix multiplications. It is fairly easy to renormalise the window vectors if this happens.

There are several advantages to inspecting the data by this method; outliers may be found (and sometimes recognised as errors), and the decision as to whether to use Statistical or Neural Net methods or alternatives may be taken in some cases. It has been found, for example, that some data appear to consist of one gaussian for each category of point under every explored projection, and in this case it is tempting to automate the visual inspection: make random choices of rotation, for each projection test to see if the distribution is acceptably gaussian, using any of the standard statistical methods, and repeat. If it is gaussian from enough directions, where 'enough' depends on n, we might feel that it would be sensible to use a gaussian distribution to model the data. If it looks very much like a collection of convex sets bounded by hyperplanes, one might use the classical neural nets.

On the disk may be found some images of vowels obtained by extracting from the NIST data base samples of many speakers saying 'Ah' and

'Oo'. Each sound was sampled and a Fast Fourier Transform applied to obtain the power spectrum. The results were binned into 12 different frequency bands, spaced more or less logarithmically along the frequency axis, and the resulting twelve numbers used to describe the sound. An utterance was thus turned into a trajectory in $\mathbb{R}^{12}$, although rather a short one in this case. The collection of such trajectories is fitted reasonably well by a twelve dimensional gaussian distribution, a different one for each vowel sound. Classifying vowels by these means may be accomplished by the methods of the next section but one. A projection program which allows you to look at the data is also available on the disk.

There is a program called *fview* written by Gareth Lee which allows projection of views and rotations of higher dimensional data. The program is available by anonymous FTP from "ciips.ee.uwa.edu.au" (I/P number 130.95.72.69), is located in directory "pub/speech/tools/fview", and is called "fview1.0.tar.Z". It takes up about 2.5Mbytes. It should run straight-out-of-the-box on SparcStations and does not require XView to be installed. It will need to be compiled to run on other machines and will need XView. The same site contains papers and data concerned with Pattern Recognition in various sub-directories.

I cannot stress too much the fundamental silliness of trying to solve a problem of pattern recognition using a package that saves anybody from ever looking at the data or thinking about how it was obtained or generated. Any method of examining the data by eye is a help. Trying to get out of the business of getting your hands dirty is to embrace intellectual death, you turn into one of those arty types who can talk about everything but can't actually *do* anything. This is even worse than being one of the hearty, unintellectual types who can remember the recipes but has no idea why they work or how anybody devised them.

4.2 Computing *PDF*s: Gaussians

By way of light relief, this section will concentrate on the simple and practical problem of modelling the distribution of points using the multivariate gaussian function. I have already indicated how this can be used for making decisions as to what category a point belongs to, and I shall go into this matter in more detail shortly. At present, let us settle for an attempt to fit a good gaussian function to a set of data. Whether we see this as a form of data compression which replaces detailed knowledge of the data with some rather vaguer approximation thereto, or an estimate of some deeper underlying process responsible for having generated the data by some ineffable mechanisms it is better not to eff, does not much matter. Either way we go through a certain computation. The terminology tends to suggest the estimation of some underlying ineffable whatnot, but it would be a mistake to suppose that the accompanying metaphysical baggage is either necessary or desirable, still less that the writer has a commitment to it.

To focus ideas, the reader might like to recall the problem of telling the guys from the gals by fitting gaussian distributions to the points on the height-weight diagram of Figure 1.2. back in chapter one, or of telling 'aah' from 'ooh' in twelve dimensions if he has had a look at the data of the disk or played with the fview program.

4.2.1 One Gaussian per cluster

Suppose I have a cluster of points in $\mathbb{R}^n$, say the heights and weights of a number of adult males, as in chapter one, *Fig. 1.2*, looking at only the Xs. In low dimensions such as this we can look at them and say to ourselves 'yes, that can be modelled by a gaussian.' Well *of course* it can. Any set of points can be. Let us remind ourselves of the elementary methods of doing so.

Let

$$\mathcal{X} = \left\{ \begin{pmatrix} x_1^1 \\ x_1^2 \\ \vdots \\ x_1^n \end{pmatrix}, \begin{pmatrix} x_2^1 \\ x_2^2 \\ \vdots \\ x_2^n \end{pmatrix}, \cdots, \begin{pmatrix} x_M^1 \\ x_M^2 \\ \vdots \\ x_M^n \end{pmatrix} \right\}$$

be a set of M points in $\mathbb{R}^n$. The *centre* of the M points (or *centroid*, or *centre of gravity*), is the point

$$\mathbf{m} = \begin{pmatrix} \bar{x}^1 \\ \bar{x}^2 \\ \vdots \\ \bar{x}^n \end{pmatrix}$$

where $\bar{x}^i = \frac{1}{M} \sum_{j=1}^{j=M} x_j^i$, i.e. each component is the mean or average of the M values of that component obtained from the M points.

In vector notation we write simply:

$$\mathbf{m} = \frac{1}{M} \sum_{\mathbf{x} \in \mathcal{X}} \mathbf{x}$$

We can take it that the centroid of a set of points contains some first order approximation to saying something about the set in terms of a simple vector. To say more about the set we proceed to the second order moments. These are the average values of the terms $(x^i - \bar{x}^i)(x^j - \bar{x}^j)$ in the given vectors, and there are n^2 of them, so we arrange them in an n by n matrix called the *covariance matrix*. Formally, the covariance matrix, $\mathbf{V}$ will have the entry in row i and column j given by

$$\mathbf{V}_j^i = \frac{1}{(M-1)} \sum_{r=1}^{M} (x_r^i - \bar{x}^i)(x_r^j - \bar{x}^j)$$

In vector notation this may be compactly written as

$$\mathbf{V} = \frac{1}{(M-1)} \sum_{\mathbf{x} \in \mathcal{X}} (\mathbf{x} - \mathbf{m})(\mathbf{x} - \mathbf{m})^T$$

The use of $M-1$, where the innocent might have expected M, occurs for reasons into which I shall not go; intuitively one can argue that in subtracting off the mean, $\mathbf{m}$, from every point, we have thrown away one point's worth of information. It is not unlike subtracting off a point from a set of M points in linear algebra in order to see if the original points lie on a line or plane; one looks at the $M-1$ non-zero points to see if they are linearly dependent. Here we have not subtracted off one of the points, we have subtracted off $(\frac{1}{M})^{th}$ of each of them.

The matrix $\mathbf{V}$ is of course symmetric by construction, and (although this is not so immediately evident) positive semi-definite. Since it is symmetric, elementary linear algebra assures us that it may be diagonalised by a rotation matrix, i.e. there is an orthogonal matrix with determinant 1, $\mathbf{Q}$, and a diagonal matrix $\mathbf{\Lambda}$, so that $\mathbf{Q \Lambda Q^{-1}} = \mathbf{V}$. The diagonal elements of $\mathbf{\Lambda}$ are called the *eigenvalues* of $\mathbf{V}$, the image of the standard basis in $\mathbb{R}^n$ by $\mathbf{Q}$ is called the *eigenbasis* for $\mathbf{V}$ and the image of the basis vectors is called the set of *eigenvectors*. Traditionally, any multiple of an eigenvector is also an eigenvector, so when the books refer to an eigenvector they often mean a one dimensional eigenspace, and sometimes they just meant eigenspace.

$\mathbf{Q}$ is unique provided that the eigenvalues are all different, whereupon the eigenspaces are all one dimensional. Each eigenvalue is non-negative (an immediate consequence of the positive semi-definiteness). If the eigenvalues are all positive, then the matrix $\mathbf{V}$ is non-singular and is positive definite. It is again a trivial consequence of elementary linear algebra (and immediately apparent to the meanest[1] intellect) that in this case $\mathbf{V^{-1}}$ is diagonalisable by the same matrix $\mathbf{Q}$ and has diagonal matrix the inverse of $\mathbf{\Lambda}$, which simply has the reciprocals of the eigenvalues in the corresponding places. Moreover, there is a symmetric square root of $\mathbf{V}, \mathbf{V}^{\frac{1}{2}}$ which can be diagonalised by the same matrix $\mathbf{Q}$

[1] If your intellect isn't very mean, and if these things are not immediately obvious, you need some remedial linear algebra.

and has diagonal terms the square roots of the eigenvalues in the corresponding places. By a square root, I mean simply that $\mathbf{V}^{\frac{1}{2}}\mathbf{V}^{\frac{1}{2}} = \mathbf{V}$.

Having obtained from the original data the centre $\mathbf{m}$ and this matrix $\mathbf{V}$, I shall now define the function

$$g_{[\mathbf{m},\mathbf{V}]} : \mathbb{R}^n \longrightarrow \mathbb{R}$$

by

$$g_{[\mathbf{m},\mathbf{V}]}(\mathbf{x}) = \frac{1}{(\sqrt{2\pi})^n \ (\sqrt{det(\mathbf{V})})} e^{-\frac{(\mathbf{x}-\mathbf{m})^T\mathbf{V}^{-1}(\mathbf{x}-\mathbf{m})}{2}}$$

and assert that this function is going to be my probabilistic model for the process which produced the data set, or alternatively my preferred device for representing the data compactly.

If a rationale for this choice rather than some other is required, and the enquiring mind might well ask for one, I offer several:

First, it is not too hard to prove that of all choices of $\mathbf{m}$ and $\mathbf{V}$, this choice gives the maximum likelihood for the original data. So given that I have a preference for gaussians, this particular gaussian would seem to be the best choice.

Second, I can argue that just as the centre $\mathbf{m}$ contains first order information telling us about the data, so $\mathbf{V}$ gives us second order information- the central moments of second order are being computed, up to a scalar multiple.

Third, it is easy to compute the vector $\mathbf{m}$ and the matrix $\mathbf{V}$.

Fourth, I may rationalise my preference for gaussians via the central limit theorem; this allows me to observe that quite a lot of data is produced by some process which aims at a single value and which is perturbed by 'noise' so that a large number of small independent factors influence the final value by each adding some disturbance to the target value. In the case where the number of factors gets larger and larger and the additive influence of each one becomes smaller and smaller, we get a gaussian distribution. Of course, it is hard to see how such an assumption about the data could be verified directly, but many people find this argument comforting.

And finally, if it doesn't look to be doing a good job, it is possible to discover this fact and abandon the model class, which *is* comforting. In that event, I have other recourses of which you will learn more ere long.

These justifications are not likely to satisfy the committed Platonist philosopher, who will want to be persuaded that this choice is transcendentally right, or at least as close as can be got. But then, keeping Platonist philosophers happy is not my job.

In dimension one we can draw the graph of the function and the data set from which it was obtained by the above process. The covariance matrix $\mathbf{V}$ reduces to a single positive number, the *variance*, and its square root is usually called the *standard deviation* and written σ. The points satisfying $(\mathbf{x}-\mathbf{m})^T\mathbf{V}^{-1}(\mathbf{x}-\mathbf{m}) = 1$ are therefore the numbers $\mathbf{m} \pm \sigma$.

In dimension two, it is possible to draw the graph of the function, but also possible to sketch the sections, the curves of constant height. These are ellipses. In particular, the points $\mathbf{x}$ satisfying $(\mathbf{x}-\mathbf{m})^T\mathbf{V}^{-1}(\mathbf{x}-\mathbf{m}) = 1$ fall along an ellipse in the plane. A simple computation shows that the integral of the function over the interior of this ellipse has the value $1 - e^{-\frac{1}{2}}$ which is about 0.4. Thus almost 40% of the points distributed according to a gaussian lie within one 'standard deviation' of the centre in two dimensions. This contrasts with the one dimensional case of course, where it is nearer 67%.

In dimension n, the set

$$\{\mathbf{x} \in \mathbb{R}^n : (\mathbf{x}-\mathbf{m})^T\mathbf{V}^{-1}(\mathbf{x}-\mathbf{m}) = 1\}$$

which might reasonably be called the set of points at one standard deviation, is a hyperellipsoid. This follows of course from $\mathbf{V}$ being positive definite. If any of the eigenvalues are zero, the hyperellipsoid is said to be *degenerate*. The proportion of the distribution within one standard deviation goes down as the dimension goes up.

It is easy enough for even the least imaginative of computer programmers to visualise a cluster of points sitting in three dimensions and a sort of squashed football sitting around them so as to enclose a reasonable percentage. Much past three, only the brave and the insane dare venture; into which category we fall I leave to you to determine.

Dimension 2

In dimension two, it is often convenient, in these days of nice computer graphics, to draw the ellipse and display the data and the ellipse on a screen. This can conveniently be accomplished by finding a parametric representation of the ellipse.

To see how this may be done, shift back to the origin by subtracting $\mathbf{m}$ from everything. We now have an ellipse given by

$$\{\mathbf{x} \in \mathbb{R}^2 : \mathbf{x}^T \mathbf{V}^{-1} \mathbf{x} = 1\}$$

Now suppose $\mathbf{V}^{-1}$ has a square root $\mathbf{A}$ which, like $\mathbf{V}$ and $\mathbf{V}^{-1}$ is symmetric. Then we can write the above equation as

$$\{\mathbf{x} \in \mathbb{R}^2 : \mathbf{x}^T \mathbf{A}^T \mathbf{A} \mathbf{x} = 1\}$$

since $\mathbf{A}^T \mathbf{A} = \mathbf{A}\mathbf{A} = \mathbf{V}^{-1}$.

Now if $\mathbf{y} = \mathbf{A}\mathbf{x}$ we have $\mathbf{y}^T \mathbf{y} = 1$, which means that $\mathbf{y}$ is on the unit circle, S^1, in $\mathbb{R}^2$. We can therefore describe the ellipse as

$$\{\mathbf{x} \in \mathbb{R}^2 : \mathbf{y} = \mathbf{A}\mathbf{x} \ \& \ \mathbf{y} \in S^1\}$$

or

$$\{\mathbf{x} \in \mathbb{R}^2 : \mathbf{x} = \mathbf{A}^{-1}\mathbf{y} \ \& \ \mathbf{y} \in S^1\}$$

or

$$\left\{ \mathbf{x} \in \mathbb{R}^2 : \exists \theta \in [0, 2\pi], \mathbf{x} = \mathbf{A}^{-1} \begin{pmatrix} cos(\theta) \\ sin(\theta) \end{pmatrix} \right\}$$

It is easy to draw sets on a computer when they have been parametrised by a single real number. The following few lines of C code indicate how we can trace the path of a circle with time running from 0 to 2π, or at least a discrete approximation to it:

```
for(int_time = 0; int_time < 629; int_time++){
   time = int_time/100;
   x =  100* cos(time) + 200;
   y =  200 - 100*sin(time);
   putpixel(x,y);
};
```

This will draw a circle at centre (200,200) of radius 100. The choice of 629 calls for some explanation which I leave to the reader. The drawing ellipse, $\mathbf{A}^{-1}$ is simply $\mathbf{V}^{\frac{1}{2}}$ which we have already seen how to compute. This operates on the points of the unit circle to stretch them in the right way to draw a one standard deviation ellipse: if you want more than one standard deviation, the modification is straightforward.

The result of generating points according to a few gaussian distributions (well, more or less. It was faked, of course) in the plane and displaying them on a computer is shown in *Fig.4.1*. Six gaussians were chosen so as to produce a ghostly hand which may be seen with the eye of faith.

The ellipses representing not one but 3 standard deviations are (mostly) drawn in the next diagram, *Fig.4.2*, where they should satisfy the most captious that they are doing something to represent the distribution of the points in the data set all on their own.

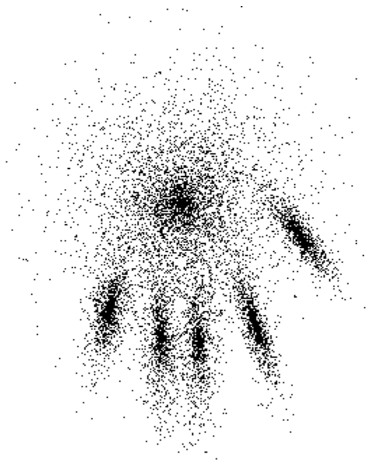

Figure 4.1: Simulated gaussians in $\mathbb{R}^2$.

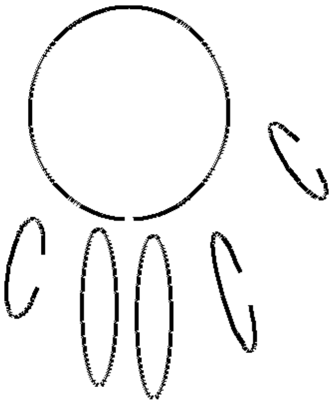

Figure 4.2: Ellipses drawn at three standard deviations for the above data.

4.2.2 Lots of Gaussians: The EM algorithm

Fitting a single gaussian to the point set of *Fig.4.1* would leave out a lot of the structure. One way to improve the description of the point set would be to go to higher order moments, since we are using only second order moments here. If we do this we certainly get more information about the distribution of the points. Another approach is the *localise the data* approach. This amounts to fitting several ellipses, or if you prefer, several gaussians.

If you think of how you might describe handprinted characters as vectors, either by moments or some other means, then it is reasonable that much of the time two exemplars of the same character will be fairly close in the representation space. On the other hand, there are equally clearly cases where they won't; if for example half of the digits of a postcode are written by Europeans with a predilection for putting a horizontal bar across the vertical stroke of a seven, and the other half are written by Americans or British who have no such urge, then the sevens will form two quite distinct clusters in the representation space. Each cluster might, if we are optimistic, be representable by a single gaussian, but the two clusters together might not. So it would be as well to have several gaussians available for modelling digits, several for each digit according to the various strategies which are available for forming them.

The use of a *mixture of gaussians* is well recognised as a statistical technique, and the so called *EM algorithm* can be used to fit a fixed number of gaussians to a data set such as *Fig.4.1*. The bibliography contains pointers to books telling you the theory of handling mixture models quite generally. I shall concentrate exclusively on mixtures of gaussians, although it is not hard to see how to adapt the methods to other *pdf*s. It is found experimentally that the EM algorithm, although very fashionable, is very vulnerable to initialisation, and converges at a rate somewhat slower than a slug bungee-jumping in treacle. I shall have a little to say about methds of doing something about this a little later.

Given a data set such as that of *Fig.4.1*, it is not at all unreasonable to decide to fit six gaussians. It is also reasonable to wonder what one ought to do in higher dimensions where it might not be so easy to decide what a good number of gaussians would be. This is an important point I shall come to later, but for the time being let us suppose that we have a data set X of points all of the same category in $\mathbb{R}^n$, some number, k, of gaussians for distributing over the points, and we seek the maximum likelihood model. Algebraically, we have to find the maximum likelihood model for

$$\sum_{i=1}^{k} w_i g_{[\mathbf{m}_i,\mathbf{V}_i]}(\mathbf{x})$$

where the w_i are weights which have to be chosen so that the integral over the whole space is 1. So we have k weights, each a real number, k centres, each a vector in $\mathbb{R}^n$, and k symmetric positive definite matrices, each n by n, each to be found so as to give the product of the likelihoods for each data point a maximum, that is:

$$\prod_{\mathbf{x}\in\mathcal{X}} \sum_{i=1}^{k} w_i g_{[\mathbf{m}_i,\mathbf{V}_i]}(\mathbf{x})$$

where of course each gaussian $g_{[\mathbf{m}_i,\mathbf{V}_i]}$ is a function of the form:

$$g_{[\mathbf{m}_i,\mathbf{V}_i]}(\mathbf{x}) = \frac{1}{(\sqrt{2\pi})^n\ (\sqrt{det(\mathbf{V}_i)})} e^{-\frac{(\mathbf{x}-\mathbf{m}_i)^T \mathbf{V}_i^{-1} (\mathbf{x}-\mathbf{m}_i)}{2}}$$

and the whole has to be maximised by suitable choice of the weights and parameters for the gaussians. This is a rather horrid problem in constrained optimisation. Rather than try to solve it exactly, it is common to use the *EM* algorithm to find a solution. EM stands for 'Expectation Maximisation' and describes the two steps in the algorithm. See the references for the origins and generality of the EM algorithm; here I shall give a procedure for obtaining the Maximum Likelihood Gaussian mixture model. I am most grateful to Arthur Nadas of the Automatic

Speech Recognition group, IBM's Thomas J. Watson Research Center, Yorktown Heights for explaining this to me, among other things, in the dim and distant past.

The EM algorithm for Gaussian Mixture Modelling

1. Initialise the k gaussians so that each of the $\mathbf{V}_i$ is the identity matrix, each of the w_i is $\frac{1}{k}$, and the k centres are taken to be some small random step in $\mathbb{R}^n$ away from a randomly selected point of the data set X.

2. For each point $\mathbf{x} \in \mathcal{X}$, and for each i between 1 and k, compute the weighted likelihood $w_i g_{[\mathbf{m}_i, \mathbf{V}_i]}(\mathbf{x})$ using the last estimate of the parameters $w_i, \mathbf{m}_i, \mathbf{V}_i$. For each point $\mathbf{x}$, let $\mathcal{L}_i(\mathbf{x})$ be the likelihood assigned to it by the i^{th} member of the mixture:

$$\mathcal{L}_i(\mathbf{x}) = w_i g_{[\mathbf{m}_i, \mathbf{V}_i]}(\mathbf{x})$$

and let S_x be the sum

$$S_x = \sum_{i=1}^{k} w_i g_{[\mathbf{m}_i, \mathbf{V}_i]}(\mathbf{x}) = \sum_{i=1}^{k} \mathcal{L}_i(\mathbf{x})$$

and let

$$P_i(\mathbf{x}) = \frac{\mathcal{L}_i(\mathbf{x})}{S_x}$$

We think of the point $\mathbf{x}$ as ready to be split up between the k gaussians according to the k fractions

$$\frac{w_i g_{[\mathbf{m}_i, \mathbf{V}_i]}(\mathbf{x})}{S_x}$$

In consequence, we have that $\sum_{\mathbf{x} \in \mathcal{X}} \sum_{i=1}^{k} P_i(\mathbf{x}) = |\mathcal{X}|$, the cardinality of $\mathcal{X}$.

3. Now re-estimate the w_i as

$$w_i' = \frac{\sum_{\mathbf{x} \in \mathcal{X}} P_i(\mathbf{x})}{|\mathcal{X}|}$$

and the centres $\mathbf{m}'_i$ as the weighted centroids

$$\mathbf{m}'_i = \frac{\sum_{\mathbf{x} \in \mathcal{X}} P_i(\mathbf{x})\mathbf{x}}{\sum_{\mathbf{x} \in \mathcal{X}} P_i(\mathbf{x})}$$

and finally, the i^{th} covariance matrix, $\mathbf{V}'_i$ is re-estimated as

$$\mathbf{V}'_i = \frac{1}{(\sum_{\mathbf{x} \in \mathcal{X}} P_i(\mathbf{x}) - 1)} \sum_{\mathbf{x} \in \mathcal{X}} P_i(\mathbf{x}) \left[(\mathbf{x} - \mathbf{m}'_i)(\mathbf{x} - \mathbf{m}'_i)^T \right]$$

4. Run through steps **2** and **3** until there is no significant change in any of the numbers being estimated, then stop.

It may be shown that in principle this converges to a maximum likelihood mixture model for the data set X with k gaussians in the mixture. In practice it is found that the initialisation is rather critical and one does not always get good results. It is easy to experiment with data such as *Fig.4.1* in two dimensions, where the data comes from a known number of gaussians, and to discover that very unsatisfactory 'solutions' are found by the above process, as *Fig.4.3* shows.

The problem of suitably initialising the EM algorithm will be discussed later, as will the issue of finding the 'right' value of k, the number of gaussians. In the chapter on Quadratic Neural Nets I shall describe a wholly dynamical approach to the business of modelling point sets which has affinities with mixture modelling.

Inspecting the data by projection can help in suggesting a suitable number of gaussians to employ in a mixture model, and can also suggest reasonable initialisations of the centres.

It is worth noting that the estimates of the eigenvalues obtained from diagonalising the covariance matrix are biassed in an interesting way: the true ellipsoid is likely to be shorter and fatter than the estimate obtained from a data set of limited size. In other words the larger eigenvalues are overestimated and the smaller ones underestimated. One can accomodate this in various ways which are left to the imagination and enterprise of the reader.

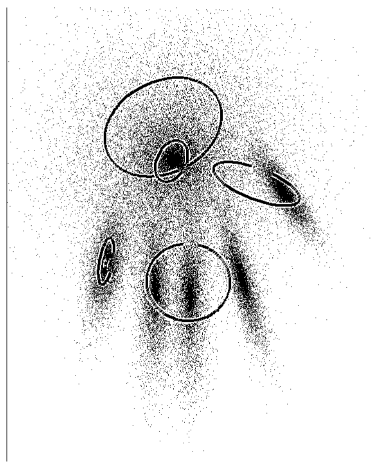

Figure 4.3: A 'solution' found by the EM algorithm with poor initialisation.

4.2.3 Other Possibilities

We might find cases where gaussians in arbitrary numbers look silly, perhaps by eyeballing the data under projection, perhaps by knowing something about the process which generated the data points, and it is a straightforward matter to select other families of distributions from those in the standard books introducing probability theory to the young. It is possible to choose some quantisation level and produce a histogram. And it is possible to use nearest neighbour methods as mentioned in chapter one.

It is also possible to prefer other than maximum likelihood models; for example we may want to use gaussian mixtures but have strong prejudices about where they should be, or to believe that the covariance matrices should all be equal, or something equally implausible *a priori*. It is not hard to make the appropriate changes to the algorithms to deal with all of the above.

In the next section, we suppose that we have obtained, by hook or by crook, some pdf for each of the categories of data point in the vicinity of a new data point, and we set about determining the category of the new point, that is we solve the standard supervised learning pattern classification problem.

4.3 Bayesian Decision

Suppose now that we have, for each category of data point, modelled the set of points in that category by some suitable *pdf*, with a gaussian or a mixture of gaussians being likely to be suitable in practical cases quite a lot of the time. We have, in the vernacular, *trained* the model on the data, and now wish to use it on some new data, either to test the model to see how reliable it is or because we want to make an informed guess at what the category of a new datum is and this is our preferred method. I shall explain the ideas behind the best known decision methods.

Recall from the last chapter when we explained Bayes' Theorem and its use in decision, that we considered the case where $g_m(x)$ and $g_f(x)$ were the likelihoods produced by two competing models, one gaussian g_m for the guys and another g_f for the gals. We decided, you may recall, to regard these two numbers as the conditional probability of having got x given model m and the conditional probability of having got x given model f, respectively. We wrote this, with some reservations, as $p(x|m)$ and $p(x|f)$ respectively. These were interpreted by the sanguine, and us, as measuring the probability that the datum x will be observed given that model m (respectively, f) is appropriate. Now what we want is $p(m|x)$ and $p(f|x)$ respectively, the probabilities that the models are true given the observation x. By applying Bayes Theorem in a spirit of untrammelled optimism, we deduced that

$$p(m|x) = \frac{p(x|m)p(m)}{p(x)}$$

with a similar expression for $p(f|x)$.

In the situation of the first problem at the end of chapter one; you will doubtless recall, yet again if you read the last chapter, the half naked ladies and possibly the trees from which they were to be distinguished. Now it can be, and was, argued that in the absence of any data from an actual image, we would rate the probability of the image being of a tree as 0.9 and of it being a naked lady as 0.1, on the grounds that these are the ratios of the numbers of images. Bayesians refer to these as the *prior probabilities* of the events. So in the above formulae we could put $p(m)$ and $p(f)$ in as numbers if we had some similar sort of information about the likelihoods of the two models. This leaves only $p(x)$ as a number which it is a little hard to assign. Happily, it occurs in both expressions, and if we look at the *likelihood ratio*, it cancels out. So we have:

$$\frac{p(m|x)}{p(f|x)} = \frac{p(x|m)p(m)}{p(x|f)p(f)}$$

and the right hand side, known as the *likelihood ratio* is computable. Well, sort of.

We concluded in the last chapter that if you are prepared to buy this, you have a justification for the rule of thumb of always choosing the bigger value, at least in the case where $p(m)$ and $p(f)$ are equal. In this case, $p(m|x)$ is proportional to $p(x|m)$ and $p(f|x)$ is proportional to $p(x|f)$ and with the same constant of proportionality, so choosing whichever model gives the bigger answer is the Bayes Optimal Solution. More generally than the crude rule of thumb, if it is ten times as likely to be m as f which is responsible for a datum in a state of ignorance of what the actual location of the datum is, then we can use this to obtain a bias of ten to one in favour of m by demanding that the ratio of the likelihoods be greater than ten to one before we opt for f as the more likely solution.

4.3.1 Cost Functions

In the argument for choosing between models m and f using Bayes' theorem, we had a prior probability for $p(m)$ and $p(f)$ and we could calculate $p(x|m)$ and $p(x|f)$. There is an extension to the idea which tries to choose not just the most likely model, but the model which will be the least expensive. If we make a decision to opt for m and we turn out to be wrong, there is, on this model, some cost associated with the error. Likewise if one turns out to be wrong in a choice of f. Presumably, being right does not incur any costs. If it does, it might be prudent to refuse to make a decision at all.

Suppose $C(m)$ is a cost associated with choosing m and being wrong, and $C(f)$ is a similar cost of wrongly choosing f under the impression it is right when it isn't. Suppose we are about to make a decision for a particular x, assigning it to m or f.

If we choose m and we are right, the cost is zero, if we choose m and we are wrong, then it is really an f, and the fraction of times it is an f is $p(f|x)$. So a strategy of always choosing m on observing x would have an expected loss of :

$$\mathcal{E}(m, f) = p(f|x)C(m)$$

Similarly, the strategy of always choosing f after having observed x would have a total expected cost of:

$$\mathcal{E}(f, m) = p(m|x)C(f)$$

In order to minimise the total cost, at each decision point we simply choose the smaller of these two numbers. Making the Bayesian substitution to obtain things we can more or less calculate, we get that the rule is to pick the smaller of

$$\frac{C(m)p(x|f)p(f)}{p(x)}, \quad \frac{C(f)p(x|m)p(m)}{p(x)}$$

i.e. We get the *Bayes Classifier Rule*:

Choose m iff $\frac{p(x|m)}{p(x|f)} > \frac{C(m)p(f)}{C(f)p(m)}$.

In the event of there being more than two categories, some minor modifications have to be made, which will be left as an exercise for the diligent student.

The above analysis is rather simple minded; for a more sophisticated discussion see the paper *On the Derivation of the Minimax Criterion in Binary Decision Theory* by Pyati and Joseph, as well as the following note by Professor H.V. Poor in the IEEE Information Theory Society Newsletter, Vol. 43, No.3. September 1993.

4.3.2 Non-parametric Bayes Decisions

If we assume that there is a model m generating the points of category X and another, f, generating the points of category O in $\mathbb{R}^n$, then the above recipe can be applied providing we have estimates of the relative likelihoods of a new datum for each model, and prior probabilities for the models.

Using a neighbourhood and counting the fraction of the existing data which is of category X can be defended as giving an estimate of the likelihood function for model m in the neighbourhood. Taking the total number of points in each category as an estimate of the prior probabilities, we get that the likelihood ratio is simply the ratio of the different counts of the two categories in the neighbourhood. If the ratio stayed approximately the same as the size of the neighbourhood got smaller until the numbers were too small to inspire confidence in the estimate, then one might be inclined to buy this.

Similarly, we could find the closest k points to our new point and then count the fraction which are of category X to estimate the local *pdf*. The same kind of reasoning applies.

Observe that both of these methods presuppose that the metric is known and not negotiable. This may not be a convincing assumption. It is necessary to take a critical view of fundamental assumptions throughout this section: all too often there is something nasty hiding under the algebra.

4.3.3 Other Metrics

Suppose we have fitted a gaussian to some data but have some difficulty in believing that the distribution is truly gaussian. We may nevertheless feel that in fitting a quadratic form to the data we have done *something* to say a little about how the data is distributed. For example, if we had the data of fig.1.4 and a gaussian for each category of point, we might believe that it told us something about the right metric to use. Calculating likelihoods requires some assumptions about how the density of the data falls off with distance, and we may feel uncommitted to the gaussian model, but we may nevertheless feel that the gaussian model is telling us something about the choice of units and the way in which distances ought to be measured.

Given a gaussian distribution containing the quadratic form

$$(\mathbf{x} - \mathbf{m})^T \mathbf{V}^{-1} (\mathbf{x} - \mathbf{m})$$

we may use this to calculate a number for any $\mathbf{x}$. If we get the result 1, we have said that the result is the distance of $\mathbf{x}$ from $\mathbf{m}$ is one standard deviation, generalising the one dimensional sense. In general the square root of the result for any $\mathbf{x}$ is called the *Mahalanobis distance* of $\mathbf{x}$ from $\mathbf{m}$ relative to the form $\mathbf{V}^{-1}$. Alternatively, the Mahalanobis distance is just the distance measured in standard deviations.

A *Riemannian Metric* on the space $\mathbb{R}^n$ is specified by assigning a positive definite, symmetric, quadratic form to every point of the space. If you have a curve in the space, you can take a set of points along it, and compute the Mahalanobis distance of each point from its predecessor, then add them up. Doing this with more and more points gives, in the limit, the length of the curve in the given metric. To actually compute the distance between two points in the given metric, take all curves joining them and find the length of the shortest.

The assignment of a quadratic form to each point of a space constitutes an example of a *tensor field*, and classical physics is full of them. General Relativity would be impossible without them.

We are not going to obtain a Riemannian metric from a finite data set without a good deal of interpolation, but there are occasions when this might be done. In the case of $fig.$1.4 for example, we can argue that there are grounds for putting on the same quadratic form everywhere, which is equivalent to squashing the space along the X-axis until the ellipses fitting the data turn into circles, and then using the ordinary Euclidean distance.

It is generally quicker to compute Mahalanobis distances (if not say them) than likelihoods since we save an exponentiation, and simply using the Mahalanobis distance to the various centres and choosing the smallest may be defended as (a) quick, (b) using a more defensible metric and (c) containing less compromising assumptions. If the determinants of the different forms are all the same, this will give the same answers anyway. And if they do not differ by very much, we can argue that we are kidding ourselves if we are pretending they are really known accurately anyway, so why not go for the quick and dirty?

It is not uncommon to take the logarithms of the likelihoods, which amounts to just the Mahalanobis distance plus a correction term involving the determinant, and since the logarithm is a monotone function, this will give precisely the same answers as the likelihood if we are just choosing the greater likelihood source. The same simplification may be introduced in the case of gaussian mixture models.

4.4 How many things in the mix?

In the last chapter I sketched the theory of Rissanen on *Stochastic Complexity*, as described in his book Rissanen(1989) and papers (see the bibliography of the last chapter). As explained above, it offers a philosophically quite different approach to finding the 'best' model to account for the data, and an approach which avoids some of the objections I have pointed out to the classical theory. In particular it appears to solve elegantly the issue of how many gaussians one ought to have in a gaussian mixture model for a particular data set. (The answer may be 'none', if the data set is small enough!)

As explained earlier, the general question as to order of a model occurs in other settings also. At the simple level of fitting a polynomial to a set of data points, the question as to what degree of polynomial is appropriate arises. At one extreme one can have the degree so high that every point is fitted, at the other we can assume it is constant. The same problem arises in the case of Neural Nets, as we shall see in a later chapter.

The gaussian mixture modelling problem, how many gaussians should we use, makes the problem quite stark. There will be an increase in the log-likelihood of the data with respect to the model whenever we increase the number of gaussians, until the degenerate case of non-invertible covariance matrices makes it infinite. But there is a penalty to be paid in terms of the number of parameters needed to specify the increasing number of gaussians. It is sensible to try to penalise the model in some way by offsetting the increase in log-likelihood with a corresponding increase in the number of parameters, and seeking to minimise the combination. But then the question arises, what should be the rate of exchange between log-likelihood and the parameter count? Rissanen gives us a natural and plausible rate of exchange. In this section I shall give some examples of computations on fitting gaussians in one dimension to data; see the discussion in the last chapter for the framework of thought which justifies the procedure.

4.4.1 Overhead

Suppose we are agreed that we are going to use k gaussians in $\mathbb{R}^n$ for modelling our data, subject to the caveats given above on boundedness, but we have no deep feelings about what k ought to be. If we are going to encode our data set relative to the *pdf* given by the k gaussians and their weights, we need to transmit or store not just the data encoded (N data points, each comprising n real numbers, each to precision P) using the gaussian mixture model but also (1) the number of gaussians k, (2) the k weights, real numbers encoded to some precision P', (3) the k centres, each a vector in $\mathbb{R}^n$, and (4) the k covariance matrices, each having $n(n+1)/2$ values. We can make some plausible simplifications: let us assume the precision of the parameters, P', is to be the same as the precision of the components of the vectors, P. This merits some thought, but let's duck the issue. Let us also suppose that the parameters can take any values inside some bounded region of $\mathbb{R}$ with a uniform *pdf*. let us also suppose that k is going to vary only within some small range, from, say 1 to 20, uniformly. Thus we lose generality but make the sums easier for the kinds of cases we need to consider. Doing this in full generality is interesting to statisticians, but we have more concrete problems in mind at present.

I shall suppose that the receiver knows what a gaussian is, and that I don't have to teach him what the idea of a function is, and so on. This is not a trivial matter if we believe that we learn to use some compression systems, but I simplify down to comparing between gaussian mixtures only. Extra overheads will therefore be the same and can be neglected in the calculation.

Then with these crude assumptions, I can code N data points in $\mathbb{R}^n$ directly by sending you NnP bits. I assume here that the precision P affects all the components of the vectors, including the digits representing the integer part. If necessary, fill to the left with zeros. This is wasteful, but not as wasteful as sending you the length each time if the length variation is small. [2]

[2] Of course, I can compress this information better if I know anything about it,

If I send the mixture model, I get a fixed number of bits to specify k, and then kP bits for the weights, nkP bits for the centres, and $kPn(n+1)/2$ bits for the covariance matrices.

The total number of bits I get to use is therefore approximately

$$NPn + N\int_B f(x)\log_2(\frac{1}{f(x)})dx + kP(\frac{n^2+3n+2}{2})$$

where $f(x)$ is the gaussian mixture model discussed earlier, and where an additive constant for specifying k, which I shall suppose does not depend on k (because it will be the same for all the values of k within our range) has been omitted. I also take it that I am integrating the gaussian f over some bounded region B in $\mathbb{R}^n$. Naturally, I should take B to be centred on the region where the data is. The obvious choice for B is to make it some fixed number of 'standard deviations' from the k centres. A simpler choice is to take some hypercube within which the data may be found and use that. This is what I shall do.

If I have faith in my (truncated) model, then the amount of compression can be computed directly from the above formula: it represents the best compression for which I can hope, for data sets of size N generated according to the model. A more cautious approach is to take the actual data and merely try to compress that, using the model as a convenient device for doing so.

What we do here is to sum the logarithms of the improbabilities of each data point according to the model. If each real number is specified to P bits, and if we have normalised everything to lie inside the unit hypercube of dimension n, then there are 2^{nP} cells each having measure 2^{-nP}. The probabilities are the likelihoods multiplied by this last number. So the compressed data will have size in bits given by the

which is the point of modelling the data. Likewise I can compress the parameters better if I know anything about how *they* are distributed, which gives us the Bayesian prior. And smaller fleas...

sum

$$NPn - \sum_{\mathbf{x} \in \mathcal{X}} \log_2(f(\mathbf{x})) + kP(\frac{n^2 + 3n + 2}{2})$$

It must be remembered that the function f needs to be adjusted to have integral 1 over the hypercube into which the data has been compressed.

Now given two such functions, all we need to do to get maximum compression in the case where k is the same and the data set fixed, is to make the sum of the log likelihoods as big as possible. This of course maximises the likelihood product. If the two fs also differ in respect of the number k, the penalty for increasing k has to be balanced against an increase in the log likelihood sum. This gives you, according to Rissanen, the better of the two models.

4.4.2 Example

Suppose we have data consisting of the 18 points:

$$\{-1.2, -1.1, -1.1, -1.0, -1.0, -1.0, -0.9, -0.9,$$

$$-0.8, 0.8, 0.9, 0.9, 1.0, 1.0, 1.0, 1.1, 1.1, 1.2\}$$

I decide to model this with two gaussians or maybe just one, on the interval from -2 to 2. I decide that I only care about the data accuracy to one part in 64, which is to say 6 bits. Rather than normalise, I shall assume that all data must come in the interval from -2 to 2. My first model for maximum likelihood with 2 gaussians will have equal weights and a centre at -1 with a variance of .015 and hence one standard deviation is 0.1225. The same applies to the second gaussian except that here the centre is at $+1$. With this value for σ (to 6 bit precision) the model for f is

$$\frac{1}{2}\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x+1)^2}{2\sigma^2}} + \frac{1}{2}\frac{1}{\sigma\sqrt{2\pi}}e^{-\frac{(x-1)^2}{2\sigma^2}}$$

The likelihoods may be approximated (by ignoring the contribution from the other gaussians) as:

data point	likelihood	log-likelihood
-1.2	0.4293	-0.845567
-1.1	1.16699	0.154433
-1.1	1.16699	0.154433
-1.0	1.628675	0.487767
-1.0	1.628675	0.487767
-1.0	1.628675	0.487767
-0.9	1.16699	0.154433
-0.9	1.16699	0.154433
-0.8	0.4293	-0.845567
0.8	0.4293	-0.845567
0.9	1.16699	0.154433
0.9	1.16699	0.154433
1.0	1.628675	0.487767
1.0	1.628675	0.487767
1.0	1.628675	0.487767
1.1	1.16699	0.154433
1.1	1.16699	0.154433
1.2	0.4293	-0.845567

This gives the sum of the log-likelihoods as 0.78 which tells us that the model is so bad that it would almost be better to send the data without any coding. I have taken natural logarithms here, if I convert to bits instead of nats, I get 1.1 bits.

The bit cost is therefore a total of

$$(18)(6) - 1.1 + (2)(6)(3)$$

bits, that is 142.9.

The corresponding calculation for one gaussian has $\sigma \approx 1.06$ and all the likelihoods are around 0.22, so the sum of the log likelihoods is about -27.25 nats. This works out at around a 39.3 bit disadvantage. This should tell you that a bad model is worse than the uniform model, which is to say worse than sending the data. The bit cost of using a single gaussian centred on the origin therefore works out at

$$(18)(6) + 39.3 + (1)(6)(3)$$

that is, 165.3 bits. This is bigger, so it is better to use two gaussians than one to model the data, but it is even better to just send the data at a cost of 108 bits. You should be prepared to believe that more data that fitted the model more convincingly would give a genuine saving. Even *I* think I could believe that, although, of course, I can't be sure I could.

Better check my arithmetic, it has been known to go wrong in the past.

4.4.3 The Akaike Information Criterion

The Akaike Information Criterion (AIC) is another attempt to tackle the problem of working out how much to penalise the parameters of the model. It proceeds by making some rather strong assumptions about the family of models, and also that the data is in fact generated by some member of the family, which is a hierarchically ordered set of manifolds, each manifold embedded at height zero in its successor. By investigating the way in which the maximum likelihood model will overestimate the expected log-likelihood (by fitting itself to noise, just like you seeing patterns in the fire), Akaike came out with the rule:

(AIC) Maximise the function (Log-Likelihood(Data) - Number of parameters)

This is easy to compute for gaussian mixture models, and omits all reference to the question of the precision of the data or the parameters. All we have to do is to use EM to get the maximum likelihood model (but see below) and then compute the Log-Likelihood and subtract the number of parameters. Do this for a reasonable range of the number of gaussians and pick the maximum.

It will be seen that the only differences between the two formulae are the base of the logarithms (natural logs for the AIC) and the precision of the parameters, since the precision, dimension and number of the data are constant. Generally the Stochastic Complexity criterion will penalise the number of parameters more heavily than the AIC. The

AIC is only appropriate when there is a fairly large data set.

In addition to the AIC, the BIC due to Scwartz and Bayesian arguments may also be used to compute penalties for having too many parameters, but it is a matter of opinion as to which of these, if any, is appropriate for any particular problem. See the paper by Speed and Yu, in the Bibliography for the last chapter, for a contemporary examination of the issues for a special case.

I shall return to the matter of compression and optimal modelling when we come to Neural Net models, which are notorious for having enormous dimensions for representing the data and consequently large numbers of parameters.

4.4.4 Problems with EM

As was remarked earlier, it is found experimentally that the EM algorithm, although very fashionable, is very vulnerable to initialisation, and converges at a rate somewhat slower than a slug bungee-jumping in treacle. Given the effective limits of precision of computers, this can simply lead to wrong answers, as was shown earlier.

Reasons for preferring EM are that it does, given good initialisation, lead to a Maximum Likelihood solution in many cases. Thus one can feel at peace with statisticians and probabilists, and assuage one's conscience about the optimality of the classification. To those who have read the last chapter, this has about as much force as chanting mantras and using garlic to keep vampires at bay, but the unreflective need all the confidence and certainty about life they can get. In some cases, the constrained optimisation problem which is what finding the Maximum Likelihood solution amounts to for a mixture model, can be solved by traditional methods, usually involving some numerical work and Newton's Method. I shall not go into this approach because it is found that with reasonable initialisation, EM works reasonably well, and I shall have something to say about Neural Model methods of accomplishing this a little later.

4.5 Summary of Chapter

The chapter started with a review of the preceding chapters and a promise to discuss the standard statistical methods for classifying points. The first section however comprised a short homily on the merits of investigating the data by inspecting the numbers. Most of us who have worked with data collected by other people have cautionary tales of its unreliability and the general fecklessness of other folk. Let us make sure nobody speaks so of *us*. Let us, in fact, scrutinise all data in as many ways as possible, and approach it on the hypothesis that it has been collected by politicians. Let us in particular project the data down from dimension umpteen onto the computer screen, and inspect it as if it might be suffering from a social disease.

The next section was where I came good on the much more practical matter of how you did the sums. I showed how you could compute the ML Gausian for a set of data points in $\mathbb{R}^n$. If the data set doesn't look as if one category of data point is usefully describable by a single gaussian, I indicated that the EM algorithm can be used, after suitable initialisation, to get the ML gaussian mixture model. The idea is to take each cluster of points in one category and model the cluster as closely as possible by some number of gaussians.

The following section showed what to do when two clusters had been modelled by competing *pdf*s. It derived the Bayes Optimal Classifier rule. It also threw in, for good measure, a quick and dirty rationalisation for the use of neighbourhood counting methods, a discussion on the Bayesian modelling of data by choosing a prior *pdf* and having the data convert us to a posterior *pdf*.

Finally, I described how the Rissanen Stochastic complexity/ Minimum Description Length approach allows one to deal with the vexing question of the order of a model. I took some 18 points judiciously arranged in the interval $[-2, 2]$ and showed how by considering the overhead of specifying the model, it was possible to justify using two gaussians instead of one in a formal manner. I described the alternative AIC rather briefly but gave you a rule of thumb which can be applied. And *really* finally, I wrote this summary, the exercises and the bibliography.

4.6 Exercises

1. Write a program to generate some random dots in a more or less gausian distribution on the real line. Now modify it so that it generates a two dimensional approximately gaussian cluster so that if I specify the mean and covariance matrix I get a suitable point set. There are good and efficient ways of doing this, but yours can be quick and dirty. Next compute the mean and covariance of the data so generated. Does it bear a passing resemblance to what you started out with? Can you account for any discrepancies? Draw the one standard deviation ellipse for your data.

2. Use the above program to generate a 'gaussian hand' as in Fig. 4.1. Try to use the EM algorithm with random initialisation to find a maximum likelihood fit for six gaussians. What conclusions do you draw about the algorithm?

3. Get someone to use your program to generate a mixture of gaussians and then try to fit the mixture without knowing how many terms in the mixture. Use the AIC and the Rissanen Stochastic Complexity measures to find the optimum number of model elements. Try this with varying numbers of points and varying numbers of mixture terms in various dimensions. Do the results surprise you? Depress you?

4. Photocopy an image of a real hand by putting your palm on the photocopier, cut it out, paint it black and scan it into a tif file. Use the EM algorithm to represent the pixel array as a gaussian mixture model. Can you find an initialisation process which gives reasonable results regardless of the orientation or position of the

hand in the image? Are the results worse than for the gaussian hand?

5. The six quadratic forms specifying the components of a hand comprise six points in $\mathbb{R}^5$. Compute the centre and covariance matrix for these points. This gives 5 numbers for the centre and a further 15 numbers for the covariance matrix. This makes a hand into a point in $\mathbb{R}^{20}$. Repeat with a new handprint in the same location and the same orientation, and with the same handedness, i.e. stick to right hand prints, giving a set of points in $\mathbb{R}^{20}$. Now fit a gaussian to these points. Repeat by turning the paper hand over so as to reverse its 'handedness'. Obtain a similar cluster for these mirror hands. Now test to see if you can classify left hands and distinguish them from right hands by looking to see which cluster they fall closest to. How badly do things go wrong if the fingers are sometimes together? (With judicious and ingenious initialisation, it is possible to get consistent fits and reliable discrimination.)

6. Repeat with paper cut-outs of aeroplane silhouettes, and see if you can distinguish between a fighter and a passenger jet. Make them about the same size, so it isn't too easy.

7. Try gaussian mixture modelling an image of some Chinese characters. Can you use the same trick of describing the gaussians by computing gaussians for them (UpWriting) in order to tell different characters apart?

8. Compute Zernike moments for some characters of printed English text and collect examples of the same character. Examine the clustering for variants of the characters by projecting down onto the computer screen (the programs with the book will save you some coding if you have a workstation). Compare the character '9' with a comma. Amend the Zernike moments to give a size measure.

9. Construct a program which does printed Optical Character Recognition. You will find problems with characters joined together and

characters which are disconnected; you will find limited font independence a problem. This is a lot of work but *very* educational, and later exercises will allow you to extend the scope of it considerably. It should be possible to get good recognition results for clean text correctly oriented by several of the methods outlined in the preceding chapters.

4.7 Bibliography

1. W.Feller, An Introduction to Probability Theory and its Applications Vols 1,2, New York: Wiley 1957, 1971.

2. D.M. Titterington, A.F.M. Smith, U.E. Makov , *Statistical analysis of finite mixture distributions* New York : Wiley, 1985.

3. B.S. Everitt and D.J. Hand, *Finite mixture distributions* New York: Chapman and Hall, 1981.

4. A.P. Dempster, N.M. Laird and D.B. Rubin, *Maximum Likelihood from Incomplete Data via the EM Algorithm*,Proc.Roy.Stat.Soc., 1, 1977.

5. Torfinn Taxt, Nils Lid Hjort and Line Eikvil, *Statistical Classification using a linear mixture of two multinormal Probability Densities* Pattern Recognition Letters 12 (1991) pp731-737.

6. James O. Berger, *Statistical decision theory, foundations, concepts, and methods* New York : Springer-Verlag, 1980.

7. James O. Berger, *Statistical decision theory and Bayesian analysis* 2nd ed. New York : Springer-Verlag, c1985

8. N.N. Cencov, (Tr: Lev J. Leifman), *Statistical decision rules and optimal inference* Providence, R.I : American Mathematical Society, 1982.

9. Enders A. Robinson, *Statistical reasoning and decision making* Houston, Tex : Goose Pond Press, 1981.

10. L. Weiss, *Statistical decision theory* New York : McGraw-Hill, 1961.

11. G. Hadley, *Introduction to probability and statistical decision theory* San Francisco : Holden-Day, 1967.

12. Abraham Wald, *Statistical decision functions* New York : Wiley, 1950.

13. Anatol Rapoport, *Decision theory and decision behaviour : normative and descriptive approaches* Dordrecht, the Netherlands ; Boston : Kluwer Academic Publishers, 1989.

14. Geoffrey J McLachlan *Discriminant analysis and statistical Pattern Recognition* Wiley 1992.

15. Geoffrey J McLachlan and Kaye E. Basford, *Mixture models : inference and applications to clustering* M. Dekker 1988.

16. Bertrand R. Munier(Ed.), *Risk, decision, and rationality* Kluwer Academic Publishers, 1988.

17. Bruce W. Morgan, *An introduction to Bayesian statistical decision processes* Prentice-Hall, 1968.