Chapter 8

Syntactic Pattern Recognition

Returning now to the principal thread of static Pattern Recognition, and armed with the righteousness that comes from having read up a little algebra and the attendant feeling of virtue, we contemplate a few uncomfortable facts.

The first is the practical difficulty attached to segmenting before measuring and classifying. It is common to explain to outsiders something of the difficulties of pattern classification and have them explain to you that standard function fitting methods[1] will work. When you draw a '5' and do so with the horizontal stroke not connected, and ask how they propose to deal with this case, they shrug in irritation and assure you that these little details may be taken care of by a variety of means, none requiring intelligence. They hint that simple bottom-up procedures, involving minor extensions in the direction of existing line segments, will take care of the difficulties. They are wrong. Their eyes tell them it is perfectly in order to make a join in some cases but not in others, but close inspection reveals that they have already made the classification by this point. It is not done bottom-up at all.

[1] Or whatever is the pet method favoured by the audience.

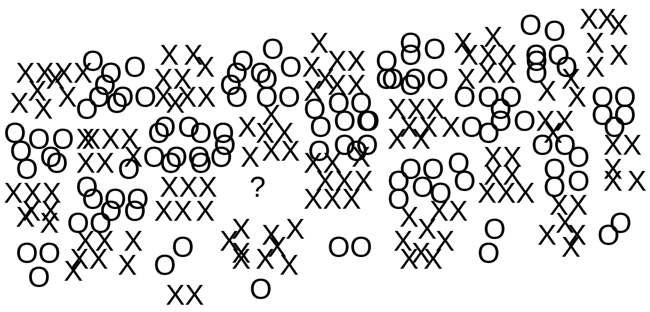

Figure 8.1: A classification problem every system discussed in this book, so far, will get wrong.

Segmentation is not, in short, something to be done independent of the classification process. But this seems to provide us with a chicken and egg situation. How do we decide when things belong together in order to feed them into a classifier, without classifying them? This is a very profitable question to think about, and it is left to the reader to do so.

The second is more prosaic and less abstract. Contemplate the simple diagram, of two data sets in the plane, shown in the first figure. I have labelled the two classes as noughts and crosses out of a fondness for the game known to the English Speaking World as Noughts and Crosses, and to the American Speaking World as Tic-Tac-Toe.

Now focus on the point at the dot belonging to the question mark. Would you expect this to be a nought (O) or a cross (X)? Note that although there is no answer to this question, since I am the one who generated the data and I am capable of anything, most people would agree that the category they would be most inclined to expect is a

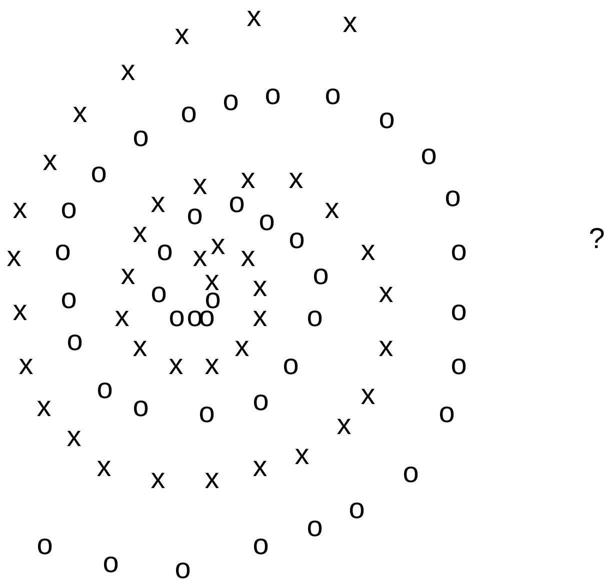

Figure 8.2: Another one.

nought, despite the fact that every method of pattern classification we have discussed would give the opposite answer. The same holds in reverse for fig.8.2, where most right thinking people would expect the ? to be a cross, despite the fact that its closest neighbours are mostly noughts.

The fact that the automatic systems discussed so far would all give different results from the eye of a child, merits a little thought. It suggests something essentially circumscribed about our methodology.

It is, of course, easy enough to devise a modification to the existing methods which will give the 'right' answer. This is to miss the point. We aren't looking for a quick fix; we thought we had the business of pattern recognition reduced to finding clusters and suddenly we discover that life is much harder than that. So we need a fix that will work not just on the two figures shown, but on lots more which may be sprung on us any minute now. We need to ask ourselves what it is about the data sets that means our simple systems don't work.

Contemplate a very similar situation: The word 'profes5or', where the '5' is replaced by an 's', despite the low level data suggesting that '5' is what is there. In all cases, there is a low level or neighbourhood expectation, and a higher level of structure to the data which is given precedence by the human eye. In the case of 'profes5or', the level above the letter level is the word level. In the case of fig.8.1 the level above the local is the alternating or chess-board pattern to the data. In the case of fig.8.2 it is the spiral curves which are extrapolated by the eye. I claim that in all cases, it makes sense to say that there are two levels of structure and that the higher level is accorded precedence in the human decision as to which class should be assigned to a new point. I claim that just as there is a process of UpWriting strings of letters into words, there must be a way of UpWriting sets of points in the plane into something else, so that we have some basis for making decisions at a different level. I claim that the intuitive idea of levels of structure, which can be shown to make good algorithmic sense for natural language can also be found to make sense in the case of more general, geometric data.

8.1 Precursors

Syntactic Pattern Recognition was inaugurated by King Sun Fu, who wanted to extend some syntactic ideas to images. He tried to do it by the following approach: first he would take his image and scrutinise it carefully for bits of subimage that, more or less, recurred. If, for example, he had handprinted alphabetics, he would find bits of curves and lines from which he could regenerate the image, or at least an approximation to it, by gluing them together in some way. He assigned a symbol to each of his image primitives, and constrained the relationships between the primitives. In the simplest case it was just an order relation, and the image decomposed into a string of primitives and hence was described by a symbol string. Rafael Gonzalez has also written about this approach. Pavlidis has also considered the issue of structured objects, usually images.

The set of strings which you get from a family of images which are of the same type, for example handwritten words, or Kanji characters, where repetitions of the same word or character generate generally different strings which are nevertheless classified together, constitutes a language in the formal sense, indeed a stochastic language. Now we produce a grammar for this language, usually a stochastic grammar. So we have one grammar for each of the families of image. Finally, for a new object, we reduce it to a string of primitives and calculate the probability for each grammar. The best bet wins. This is similar to straight statistical modelling of points in $\mathbb{R}^n$ but gives us a class of models for languages instead. It should be noted that this is almost exactly what is done in the standard VQ/HMM automatic Speech Recognition systems. The space is chopped up into lumps, and it is the lump through which the trajectory goes which determines the symbol. The symbol strings are then used to generate a hidden markov model, a stochastic grammar.

There are four problems associated with this plan:

First, the order relation may not suffice for a specification of the way the primitives are related to each other, and generally does not.

Second, the system for choosing the primitives is frequently fraught with problems and normally involves human intervention. It should, one feels, be taken care of automatically.

Third, the system of matching the actual signal to the appropriate primitive may be equally doubtful and must inevitably entail some degree of forcing a continuum of possible forms into a discrete set of allowed descriptors. It is reasonable to want this to emerge from the data rather than to be forced upon it in essentially arbitrary ways.

Fourth and finally, the business of inferring a grammar from a sample of strings is often difficult and sometimes impossible to do in any manner which does not feel arbitrary and insanitary.

To bring home all of them at once, suppose we feed the system a set of images of equilateral triangles, all scalar multiples by an integer of a smallest such one. We provide the system with primitives as shown in

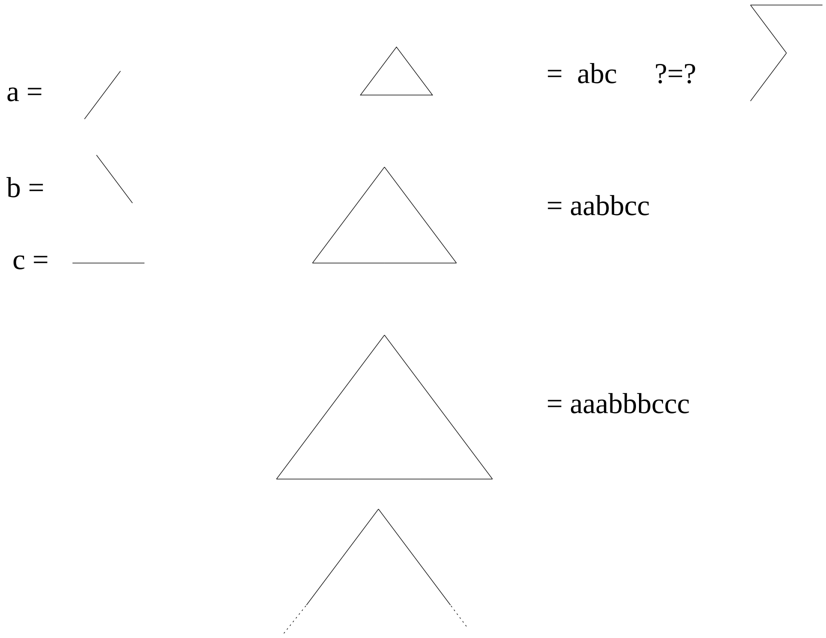

Figure 8.3: Classical Syntactic Decomposition.

fig.8.3. Then the string for triangles of side n units is $a^n b^n c^n$ which is a context sensitive language.

With this example, one might wonder why abc couldn't also describe a non-triangular shape like the one shown on the right of fig.8.3. One might wonder if choosing three corners mightn't have been better, or a useful adjunct. Going from the triangle to the string seems to entail some convention which is less than clear.

The system seems absurdly rigid, having no way of describing a triangle half as big again as the basic one. One could find ways around this, but what if the edges didn't quite meet? What if the lines were curved a little? It looks as though some sort of a symbolic strait-jacket is being forced onto a geometric world.[2] The straight-jacket approach leaves one

[2]To a geometer, it looks as though the algebraists have gotten into the cookie-jar when everyone knows they are there to tidy things up and nit-pick about details, not go around being creative. Or, even worse than algebraists, gasp, *logicians*. When logicians go around being creative, universes totter and crumble.

wondering how one is going to decide to push cases into the formalism in general.

And finally, one might be a little unhappy about a system which had to do such careful counting in order to infer the language. Inferring regular languages is easy, but inferring context sensitive languages does not sound a good way to go.

If one contemplates this example and the fixes required to make it fly, one feels some sympathy for the basic insight: we need some mechanism for obtaining descriptions of at least some geometric objects in terms of levels of structure, just as we have with natural language. But it is less than clear that the program inaugurated by Fu is the way to go forward. It seems more natural to preserve the geometric structure of images for as long as possible, until the description at a symbol level occurs naturally.

Other kinds of grammars have been devised, in particular graph-grammars for graphs and stochastic grammars for images (Markov Random Fields) but none have been free of flaws. When they allowed inference, it did not lead to hierarchies of description which look the least like natural language structures. In what follows, I shall outline a theory which does it in a more agreeable manner. In order to avoid excursions into category theory, I leave the details of the abstraction to papers mentioned in the bibliography, and shall instead concentrate on giving algorithms which perform the needful operations. I start with some very simple cases in order to focus the mind on the essentials.

8.2 Linear Images

In fig.8.4 we have a representation of a finite pixel set such as might be seen on the screen of a computer. It is easy to see some structure in the image: it is far from being a random set of points inside the rectangle which is the computer screen.

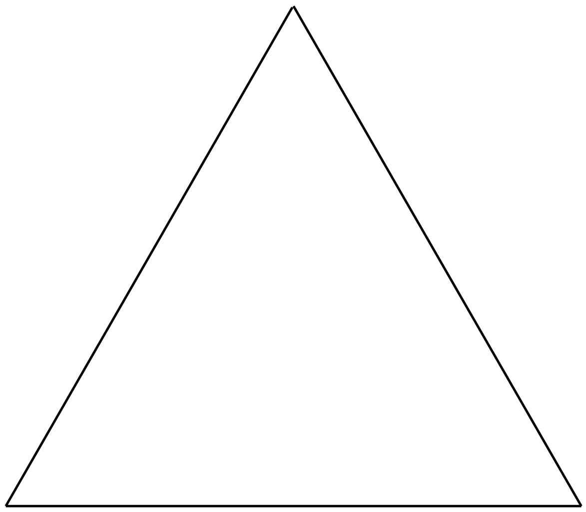

Figure 8.4: A pixel set with some structure.

Let us first choose to regard this as a finite set of points in a bounded region of $\mathbb{R}^2$. This representation allows us to treat resolution issues as a separate matter. From now on I use the term *pixel* and *point of the image* interchangeably.

I propose to treat every point of the image as being a *symbol*, in an abstract sense which I shall clarify by example rather than by formal definitions. I want to apply to the image the same technique which chunks a stream of symbols (letters) into a new stream of higher level symbols (words) from a different alphabet. I shall thus have a chunking UpWrite process which will take some subset of the image and declare it to be a single chunk.

I proceed in a manner essentially similar to the chunking process for letters. I choose a *resolution radius,* k, corresponding to the k in a choice of $k$grammar. I choose a pixel and set up a ball of radius k, which need not be an integer. I then take the set of all points of the image within the ball. This has to be turned into a local predictor of

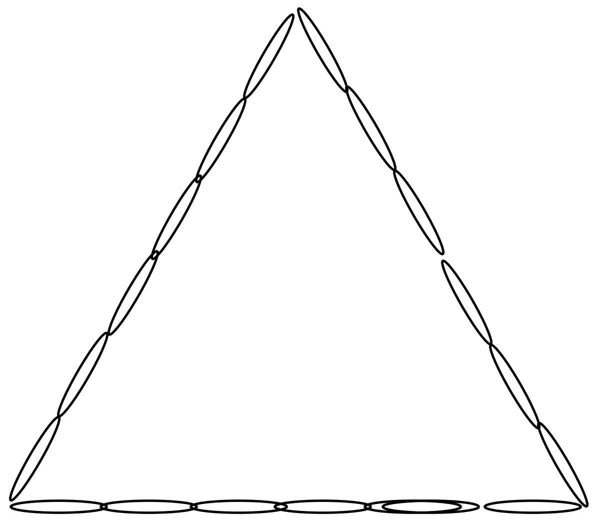

Figure 8.5: Prelude to Chunking.

other points. There are many possibilities, but I shall simply compute low order moments for the set of points in the ball. For simplicity of exposition, I shall compute zeroth, first and second order moments of the set. This is equivalent to computing the number of pixels, the centroid of the set, and the covariance matrix of the set of points inside the ball. I may conveniently represent the quadratic form specified by the first and second order moments by drawing an ellipse, as in chapter four. This becomes a predictor of adjacent points, and also of adjacent ellipses, in an obvious way: we may expect points within some Mahalanobis distance of the centre, and we expect other balls to be placed on the image, so we expect other ellipses with centres about Mahalanobis distance two apart, most probably, therefore, along the major axis. I remove the point selected as my first centre or even all the points in the ball, and repeat with another choice of ball. In this way I can rapidly reduce the diagram to the set of quadratic forms shown in $fig.$8.5.

We have now got something which is analogous to the $k$gram predictor,

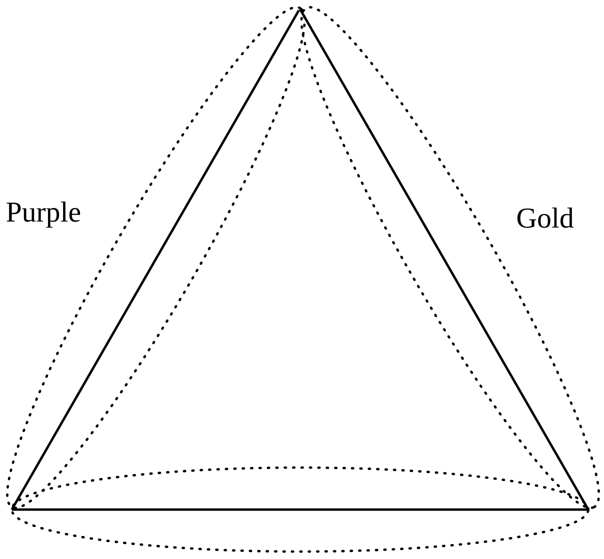

Figure 8.6: Chunks.

but which works in a wholly geometric setting. We use it to perform entropic chunking of the pixel set. More precisely, we 'colour' a set of pixels inside one ellipse, say, purple. If the pixels in a neighbouring ellipse are reasonably well predicted, i.e. if they are in the same line in roughly the expected place, we colour them purple as well. If not, we don't. We proceed in both possible directions, colouring pixels purple, until we come to an end where the behaviour is different. We then select another pixel, uncoloured, and another colour, vermillion say, and repeat. The effect of this on our triangle is clear enough: each side gets painted a different colour and we wind up with three colours. Each colour denotes a chunk of pixels.

The final stage of the chunking UpWrite is to assign a symbol to each chunk from a new alphabet. Since we want geometric representations, we want a single vector which describes the set of points along an edge. Since we want it to work for other sets in other images, it must not be specific to line segments but must work for any shaped set of points . We choose, from many possible representations, to compute low order

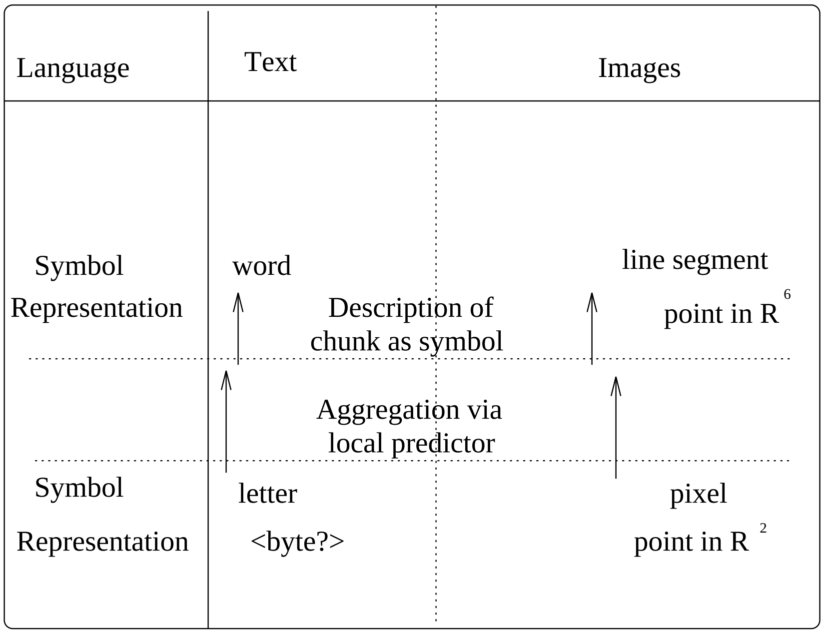

Figure 8.7: A useful Analogy.

moments. For straight line segments, zeroth, first and second order suffice. So we simply compute three sets of moments, one for each side of the triangle, and since the original points are in $\mathbb{R}^2$, and there is one number for the zeroth order moment, two first order moments and three second order moments, we wind up at the UpWrite stage with three points in $\mathbb{R}^6$ as a description of the triangle.

We have now UpWritten a set of symbols (points in $\mathbb{R}^2$) to a smaller set of symbols from a different alphabet (points in $\mathbb{R}^6$) in a way which is analogous to the way in which we extract words from strings of letters. The situation can be described by the diagram of $fig.$8.7 :

Having done it once, there is nothing to stop us doing it again. In this case a judicious choice of radius will include all three points, and there is only one chunk. The whole lot may be UpWritten to a set of moments. In this case, again the zeroth, first and second order moments suffice to specify the set of three points. We obtain one point in $\mathbb{R}^{28}$ describing the triangle, at the second level of UpWrite.

Having done this with one triangle, we may do it with another which is smaller or bigger than the original one. Suppose we take a scaling parameter, r, and multiply the original image by r in order to get the scaled version. I shall suppose that there is no translation of the images. Let us suppose, in a spirit of charity, that r varies from about 0.5 to 1.5. Now provided that the resolution radius is big enough to enclose enough pixels to avoid total degeneracy in the covariance calculation [3], and sufficiently small not to enclose more than one edge at once, the same process will work to the first level. Each edge will be in a different location and will be scaled appropriately. At the second level of UpWrite, the centroid will be in the same position, but the covariance will go up as the square of the scaling term. Thus the different triangles will UpWrite to points lying along a smooth curve in the topmost space, $\mathbb{R}^{28}$ in this case.

This set of 'triangles', can also be described by moments (although greater than second order might be expedient) to specify it as a point in yet another space. This point in the top level space corresponds to the language of fig.8.3. From it, we can generate the points of the triangle space. From each of these points we can DownWrite to a set of line segments, and from each of these to a set of pixels. The DownWrite is not generally unique, but DownWrites often aren't.

It should be clear that we have all the capacity to precribe equilateral triangles of the Fu program, and rather a lot more besides. If someone gives us a collection of triangles and a collection of squares, we will get two clusters in the space formerly containing only triangles as points. Providing we don't scale down to zero, the clusters will be disjoint. The clusters will be named as vectors at the top level, but because they are quite separate and no intermediate forms occur, they might just as well be described by symbols. Indeed, a symbol may now be interpreted as being a point in $\mathbb{R}^n$, assigning an alphabet of them a metric structure.

If instead of having a triangle we had presented the system with a drawing of a cube, as in fig.8.8

[3] mere singularity is unimportant; to have one eigenvalue collapsing to zero may be regarded as a misfortune, to have both collapse seems like carelessness

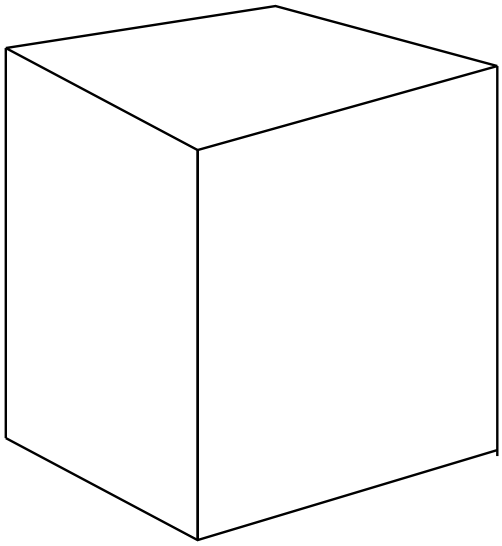

Figure 8.8: An object requiring more levels for its decomposition.

we should have been obliged, in order to do a good job of capturing the higher order structure, to go to a higher level of UpWrite. And of course, it can go further, as the world of block objects shows.

It is possible to write a program then, which can be fed as input a set of line segments at various orientations and locations. If we had given it a set of vertical short lines, and a set of horizontal long lines, disposed anywhere on the screen, the program would produce two clusters in $\mathbb{R}^6$. Given a label to each cluster and a new line segment, we could use standard methods to decide to which family it belonged. Moreover, it is a simple extension to the program to have it presented with squares and triangles, so that having first found the edges, it then UpWrites each square to four points in $\mathbb{R}^6$, each triangle likewise to three points in $\mathbb{R}^6$, and then performs exactly the same trick again to obtain a cluster of square points and a cluster of triangle points. These might be taken to be in $\mathbb{R}^{28}$. This would give us a natural system for classifying triangles and squares. It is easy to see that the same process can be applied again to higher level objects such as cubes and pyramids, and even

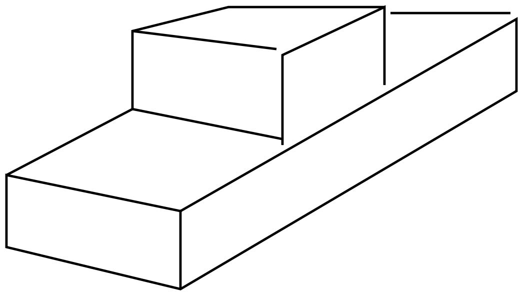

Figure 8.9: And one requiring even more.

higher order objects built up out of boxes. And the only limitations on this process are computational.

8.3 Curved Elements

The images containing only blocks, and things made from blocks, are but a small and unnatural subset of the set of all images. It is essential to be able to treat of more general objects before we can feel warranted in taking ourselves seriously. So the question arises, can we decompose more complicated objects than linear images? Happily the answer is that we can. If we have an image built up out of curves, at some suitable level of resolution, between where the noise intrudes and the global structure intrudes, we may use the same process to perform chunking, and the same process to perform the UpWrite of each chunk. Higher order moments than two will generally be necessary in order for the general shape of the curved chunk to be described with sufficient precision to allow a convincing DownWrite. There is plainly no intrinsic obstruction however to dealing with images composed of piecewise curved pixel segments. Moreover, the resolution radius is a parameter which can be adjusted in some cases to distinguish noise from signal, a point which will be enlarged on later.

The significance of curved elements is, as the thoughtful reader will have noted, that they arise rather naturally from physical objects coming in front of other physical objects. Boundaries can often be taken to be piecewise curved objects with noise.

8.4 Regions

Images of such things as characters, which are usually long thin things stuck together, can either be thinned or can be treated as virtually thin. For example, Japanese Characters can be, for the most part, treated as built up from *strokes*, which can be found by the above method. Note that what we are doing is modelling the data as locally describable by positive definite quadratic forms. Now we have not assigned a *pdf* to the pixels, but we might have done: it would be easy to use a gaussian distribution derived from the form. Not that the data looks remotely gaussian, but then we are working only to a crude second order approximation. And that means that we are modelling the image as a mixture of gaussians. We might reasonably give up on our arbitrary procedures and use the EM algorithm to fit the gaussians and hence the quadratic forms to the data. This is a useful way of getting some uniformity in the disposition of the quadratic forms, but it has problems associated with it. It is usually necessary to find a sensible way of initialising the forms, or EM can produce quite grotesque decompositions.

Having found strokes by chunking the forms, we then need to UpWrite the strokes. Some characters feature long curves, and if these can reasonably claim to be strokes, we may need decriptors that go to higher than second order, but this is of course merely a computational incon-

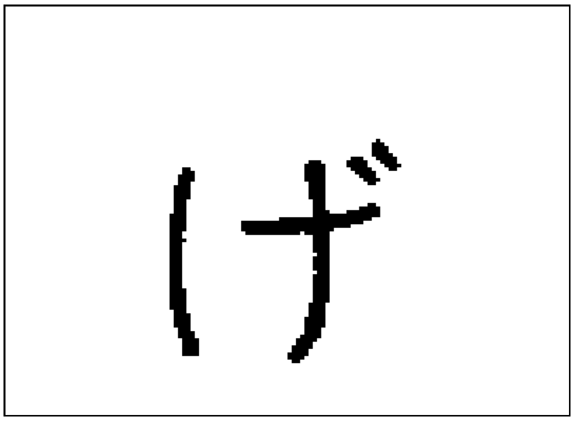

Figure 8.10: Japanese handwritten character.

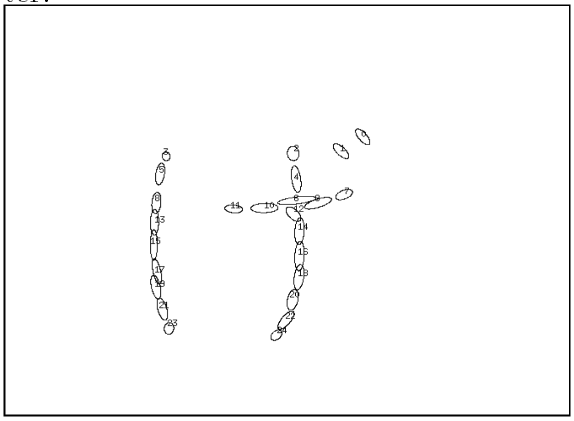

Figure 8.11: Finding chunks.

venience. In principle, we should go from pixels to strokes and from strokes to radicals, then from radicals to characters. There will be variation at each stage of the process, and noise at the pixel level will be quite different from noise at the stroke level.

A system of this sort has been trained quite successfully on handwritten Kanji and Hiragana characters, and appears to give pleasing results which do not depend strongly on the vocabulary size. A similar system has been applied to boundaries of differenced images obtained from tracking moving objects in a domestic environment, and two levels of

UpWrite suffice to pick out heads of human beings moving before the camera.

8.5 Parameter Regimes

There are two parameters associated with the chunking UpWrite as described so far, one is the resolution radius and the other is the order of the moments used to compute the UpWrite symbol assigned to each chunk. It is natural to wonder how one can pick the best values for these parameters.

A choice of resolution radius amounts to a decision that smaller scale variation will be treated as noise and averaged over. The decision as to what constitutes noise and what signal is seldom clear cut in the real world.[4]

The problem is not well-posed: in order to decide what is the optimum resolution one must know what one is trying to perceive in the image, and there may be structure at many levels. The following example was suggested to me by Richard Thomas of the University of Western Australia:

Look at a page of handwritten text, as in fig8.12. If we choose a resolution radius bigger than the mean distance between lines of text, we get an image, after chunking, which looks like the region in fig.8.13.

If we had used a smaller radius, we should have still got the same outcome, until the resolution falls below a certain value, whereupon we should have got fig.8.14, briefly, before getting more details of the word level.

It is clear that there are stable regimes for the parameter values, and that between them there are fairly fast transitions. It depends rather

[4] As when one's progeny want one to switch off the Bach so they can concentrate on the Heavy Metal, or *vice versa*.

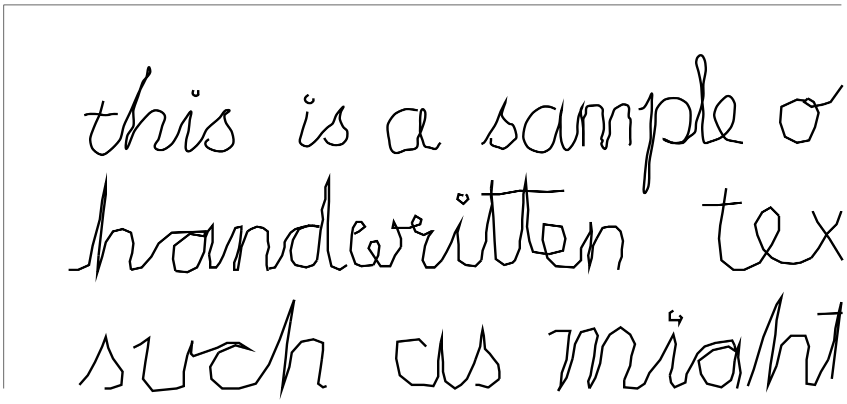

Figure 8.12: A bit of a page of handwritten text.

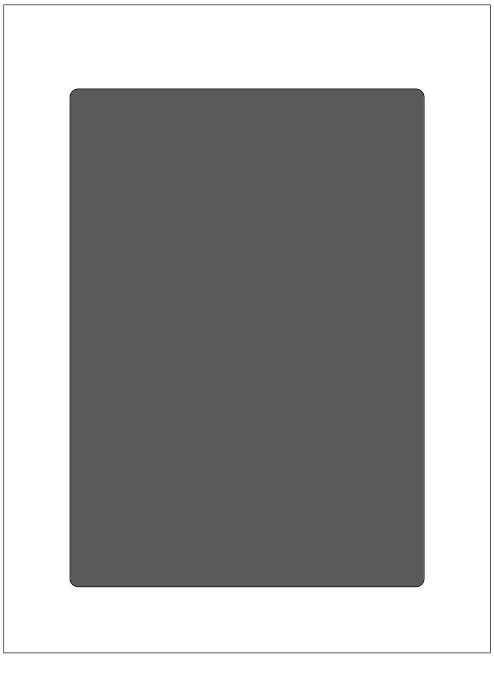

Figure 8.13: The page at low resolution.

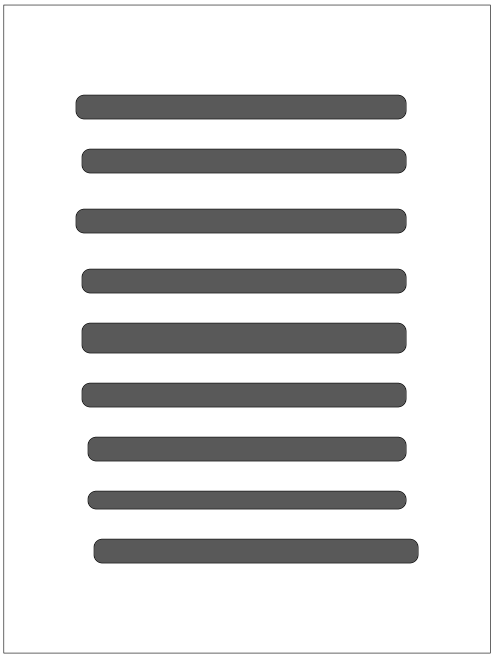

Figure 8.14: The page at intermediate resolution.

on what we are trying to do which of these regimes matter to us most.

I shall often observe that a problem is solvable by the general means outlined in this chapter, by simply noting that there are stable parameter regimes which solve them. A purist might grumble that I do not give procedures for finding them which are practicable on present day computing equipment, and that investigating a large sample of parameter values concurrently requires specialised hardware of colossal power. This is correct. In a later section I shall discuss what we are here regarding simply as a more sophisticated pattern recognition system, in the different and significant context of a new class of neural models, and the suggestion will be made that neural implementations would proceed at many different scales or parameter values concurrently, and would also infer the regime structure, and correlations between regimes.

In particular applications, we may cheat and use the human eye to decide on a suitable scale value. This is generally more convenient than trying a range of values, when it is possible to do it at all.

The situation with respect to order of UpWrite is, I suspect somewhat different, but I shall go into that below when I discuss neural implementations of the UpWrite process.

8.6 Invariance: Classifying Transformations

An important case of families of images occurs where there is essentially one object, which generates a set of images by the application of a Lie group of transformations, or some neighbourhood of the identity in such a group. [5]

In order to focus ideas, I shall consider again the simple case of triangles which are all equilateral and which differ only in the scale. I shall suppose that we have a standard such triangle with the centroid at the origin, and that by taking a range of scalars between 0.5 and 1.5 I have shrunk or enlarged the basic triangle to give a set of images, one for each scalar.

If we UpWrite the triangles to points in $\mathbb{R}^{28}$ in the standard way, we see that the set will constitute an embedding of the interval $[0.5, 1.5]$ in $\mathbb{R}^{28}$. The line will be curved rather than straight, and in practice we will get a finite point set on or close to the idealised embedding.

Doing the same thing with squares and hexagons will yield two more such curves. If the size of the standard object in all three cases is similar, then the curves will lie approximately on a space of 'scaled planar objects'. Other objects, such as pentagons, heptagons, octagons

[5] A *Topological Group* is a collection of invertible transformations such that the composite (do one then another) of any two is a third, and also having the property that the multiplication operation and the inversion operation are continuous. A *Lie Group* (pronounced 'Lee') is a topological group where the operations are also differentiable and the set forms a manifold, which is a higher dimensional generalisation of a curve or surface. This definition is sloppy and intended to convey a vague and intuitive idea which is adequate for present purposes.

and circles will also lie on such a space. This space is foliated by the curves representing the Lie group action. Given a new object, it is possible in principle to estimate the scaling curve upon which it lies, and to be able to recognise a scaled version of the object which has never been seen before. In other words, we may learn the transformation.

Less ambitious undertakings have been found to work quite satisfactorily: in an application involving the recognition of aircraft sillhouettes, the space of each curve was found for two aircraft, and a hyperplane fitted to each of them. Scalings from 60% to 140% of the original size were used. A new silhouette of one of the aircraft was then presented to the system, and it was classified according to which of the two hyperplanes was closest. By this means it was found possible to obtain correct classification down to about 20% of the original size, at which point resolution issues came into play. The system was capable of learning the space and performing creditable extrapolations in order to achieve a classification.

Other Lie Groups occurring naturally are the group of shifts or translations, and the group of rotations. The cartesian product of these spaces with a scaling space gives, for each planar object, a four dimensional manifold embedded in the top level space.

It is of course also possible to use invariant moments, but this presupposes that we know the appropriate group. If we contemplate the general situation, we see that this may be infeasible. For example, an image of a bar code printed on a can of beans is deformed in a way which is rather harder to specify, and the transformations to a trajectory in the speech space involved in (a) breathing over the microphone, (b) enlarging the vocal tract, (c) changing sex, we see that although we have group actions on the data, and although factoring out these group actions is highly desirable, an analytic description of the action is at least difficult and generally impossible to obtain.

A symmetric object will not generally give an embedding, but will give an *immersion* of the group[6]. If, for example, we rotate a square, the

[6]The term is from differential topology, and may be found in any of the standard

result of UpWriting will not embed the circle group SO(2) in the top level space, because the path traced out will recur after one quarter of a circuit. We get a circle mapped into the space, but we trace it out four times. Given noise, quantization and finite sampling, the result may be four very close lobes, each close to what is topologically a circle. This gives us a means of identifying symmetries in the object.

If we consider three dimensional rotations of an object such as an aircraft which is then projected onto a plane, as when we take a video image, we get, via the UpWrite process, a space which may not be a manifold, but which is likely to be one for almost all of the points. If the reader thinks of something like a circle with a diameter included, he will observe that the object is pretty much one dimensional, in that for most points, a sufficiently small neighbourhood of the point will give a set which looks like an interval in $\mathbb{R}$. The two points where the diameter meets the circle are singular, but there are only two of them. For most points of the UpWritten image, a small perturbation derived from a rotation in three dimensions, will give another point which if DownWritten to the image set will give something qualitatively the same. Imagine an aeroplane or a cube 'in general position'. For some views of the cube or the aeroplane however, the image degenerates into something different from neighbouring views, as for example when the cube is seen along an axis orthogonal to a face and we see a square. There may be discontinuities in the UpWrite at such locations; nevertheless, the space is mainly a manifold, it may be UpWritten as an entity in its own right. When this is done with distinct classes of objects, we recover at this level separated objects which may conveniently have the symbol designating them replaced by an alphabetic term, since the toplogy has become trivial. Thus we see that the case of strings and symbols in the usual sense may be recovered as a degenerate case of the more general topological one.

texts. It is not necessary to expand upon its precise meaning here, and an intuitive sense may be extracted from the context.

8.7 Intrinsic and Extrisic Chunking (Binding)

I was guilty of glossing over a point in drawing an analogy between $k$gram predictors in natural language text, and the use of forms or gaussian distributions to predict close pixels in an image. The point merits some attention. In the case of the $k$gram predictor, we actually do not simply use one $k$gram to predict the next symbol, we run along a lot of text and generate a probabilty distribution conditioned by the preceding $(k-1)$gram. Thus we apply a knowledge of what other $k$grams consist of. In the case of the forms on arrays of pixels (or higher level objects), we used only the distribution of points within one region. I shall call the latter *intrinsic* and the former *extrinsic* chunking.

It is possible to perform both in each category, the text or the images. If, for example, we used $k$grams on a text consisting of long strings of one letter followed by long strings of another, then in the middle of one string we should predict another letter the same, and the chunks (words) would all be strings of one letter. Dually, if we took sequences of forms and learnt that one form tended to predict another in a particular position reasonably close to it and in a similar orientation, we should be able to use extrinsic chunking of forms. We see also that storing the possibilities and their counts allows us to extract straight line segments and also 'corners' as significant 'features', by what is a *clustering UpWrite* instead of the chunking UpWrite we have used exclusively thus far in our discussion of images. Here, we do not use the entropy of the conditional probability distribution, we use the information supplied to the predictor by the data.

This is of particular significance when we contemplate alternative training procedures for generating recognition of such things as cubes. It was tacitly supposed that the system was trained first on line segments, then on faces consisting of parallelograms, and finally on cubes, so that it would have some regions of the UpWrite spaces at intervening levels already containing points which could be recognised. It is hardly feasible however to train a human face recogniser on noses, eyes and mouths separately, and we confront some problems when we try to train exclusively on cubes. The question is, what particular set of points representing line segments should be selected so as to find faces? Why not select the 'Y' shaped triple of line segments which are a distinctive feature of a cube seen in general position instead? Indeed, why not? It seems likely that such 'features' are indeed used by the human eye.

In order to extract a face of a cube, as a distinctive feature (i.e. as a point in a suitable UpWrite space), in a set of images of cubes, we would seem to need to perform a rather painful combinatorial search through the space of line segments, choosing some subset which recur often enough to be worth grouping into an entity or feature. This problem of entity formation has troubled psychologists and also neurophysiologists, by whom it is referred to as *binding*. It is an important issue, and I shall try to persuade the reader that the method outlined here reduces the matter to a formal problem in geometry and statistics.

Suppose our data consists entirely of binary images of linedrawings of cubes, each in general position, differing in respect of location, size, and also orientation with respect to the plane on which they are projected. Suppose, moreover, that each image is perfect, with only quantisation noise occurring. Then there is nothing to be gained from a syntactic decomposition at all, and it would be sensible to chunk the whole object on each occasion into one set of pixels, and to UpWrite this using higher order moments to obtain an embedding of the transformation group. The UpWrite would be all done in one step, and would hardly be justified in being called an UpWrite at all. The presumption is that new data will also look like a cube in general position, so there's nothing to classify. Similarly, if there are images of pyramids in general position *and* images of cubes in general position, then we could simply UpWrite each directly to a point in a moment space and obtain two manifolds, one for each object, with little difficulty. Recognition could be accomplished with another level of UpWrite or in other words by modelling each manifold crudely and computing distances.

Now let us suppose that the data set of cubes and pyramid line drawings is modified to contain noise at the level of the line segments, that is to say, some of the lines are too short, too long, slightly slanted, or slightly displaced from the ideal position. Now a direct moment description of the sets leads to very considerable variation at the UpWritten level. If, in addition, some of the lines are missing altogether, something that the human eye will recognise as a cube with a bit missing will not usually be correctly classified. The situation arises naturally in the case of handprinted characters, where variation at the stroke level can reduce recognition rates considerably.

In this case, there is a plain evolutionary advantage in finding intermediate structure, and it can be done by intrinsic chunking, or by extrinsic chunking. The former is simpler, but the latter is more powerful, since it tells a program trained on the data described, that at the lowest level of pixels, there is a structure which can be described locally, and which extends either linearly or so as to produce corners. We collect sets of quadratic forms and discover that they are locally correlated in one or the other of two simple ways. In general, we take a small neighbourhood of a point in the space, and describe it by UpWriting it. Now we take a ball in the UpWrite space and UpWrite the points in that. Now we look for clusters in this space. In the case of the cubes and pyramids, we discover that in the case where we are looking at, say, pairs or triples of ellipses, each describing a small region of the image, we get a cluster of ellipses in straight lines. We may get a smaller cluster of ellipses in right angled bends for cube images, and similarly for 120^o bends if there are pyramids. The existence of the clusters tells us that there is structure and that it should be exploited. It also tells us how to exploit it.

We select then a prediction system extracted from the clustering, and use it to predict where the next ellipse should be in each image, and of course use the information supplied by the image to partition the pixels into line segments.

This procedure works at every level. If there is noise at the level of faces of a cube or pyramid, we look at the sets of points in $\mathbb{R}^6$ corresponding to, say, a cube. There will be nine of them. Taking any neighbourhood of one big enough to include at least one other such point, and taking note of where it is by computing moments of the pair, we repeat on a lot of images of cubes and discover a cluster. This cluster might denote the angled bends between adjacent edges, the 'Y' shaped feature in the middle, or the parallelism of opposite faces. All that is necessary is that the property consitute a cluster when data is accumulated from a set of images, in other words that it recur with sufficient frequency. If the neighbourhoods chosen are small, then we may find ourselves going through intervening stages where we group together edges into corners or opposite faces. If they are larger we shall find faces, and if even larger, we shall find only the whole cube.

It is plain that there is a case for some kind of feedback from one level to its predecessor saying what is a sensible size of resolution radius. If the data is too surprising, maybe a smaller choice of radius at a preceding level would be a good idea.

Note that the combinatorial issue of which points to group together is solved by the following strategy:

- take a point and some size neighbourhood in the space.
- Describe the shape of the cluster of points in the neighbourhood in an UpWrite space.
- Repeat for lots of different points in the same set and also in different data sets.
- Look for clusters. If the clustering is poor, change the size of the neighbourhood in the first step and try again, or if you have a brain on which to run things, do lots of different choices in parallel and select out all the cases where there is clustering.

Such an approach has been used in finding strokes making up the boundaries of moving solid objects such as heads.

8.8 Backtrack

This chapter started with the observation that there is a fundamental issue of segmentation, which should properly be part of the classification process: a little thought shows that we have indeed the possibility of incorporating the two matters in a coherent way. If we proceed to extract structure from a postcode which has the stroke of the five distinct from the body, then we can easily see that there will be a stroke cluster corresponding to the digit '5' provided only that there have been lots of 5's offered to the system. Moreover, cases where the connection is made and where it is not quite closed will simply be variants of the same cluster of three strokes. Since the stroke triple consisting of all the strokes for the digit '5' will occur with relatively high frequency, we will get a cluster of these triples.

In effect then, the system under discussion will do automatically the aggregating into what sets should become segments, hierarchically, under the accumulation of data. We have disposed of the segmentation problem.

There were two simple data sets which all existing classification systems get wrong. It was asked if we could find a generic 'fix' which would do better than the naive clustering methods discussed thus far under the heading of static pattern classification.

The answer is gratifyingly positive. Consider first the chess-board pattern. At the lowest level of aggregation, we find neighbourhoods in a small ball correlate with other adjacent small balls provided the diameter of each ball is small compared with the size of a square. The points are technically in $\mathbb{R}^3$, with the last component either $+1$ or -1 depending on the colour or type of point. The entropic chunking will thus collect all the points of a square together for the next level of UpWrite.

The next level now represents the alternating pattern. In a suitable space of square regions, perhaps crudely represented by fourth order moments of a square in $\mathbb{R}^2$, we get alternating blocks of points in the base level. Taking any one such point and looking at a neighbourhood, we get, as we go across the image, very tight clustering: we keep getting a point which is a cross being surrounded by opposite category points along ranks and files, and diagonals which are of the same type if the neighbourhood is slightly bigger. This structure is in effect learnt from other parts of the same data set. Using such a neighbourhood descriptor, the entire image is decomposed into iterates of this primitive. The system learns the process by which it appears to have been generated in the same way as the bag structure ABCDABCDABCD was learnt in the last chapter.

Classifying a point in a region is done by downwriting from the higher level description which will fill the empty square with points which although not present in fact are deduced from the higher level structure. Now we can determine the category of the unknown point by looking at its neighbours- not the actual neighbours but the virtual neighbours generated by the DownWrite. This is analogous to identifying the word *profes5or* as being close to the word *professor* which has been seen before, DownWriting the word to its spelling (a rather trivial operation), and then identifying the symbol between s and o

We are using the same machinery in both cases, just doing it with different interpretations of the terms 'symbol' and 'concatenate' in fact. This point requires some mathematical sophistication not required of the reader of this book.[7]

In the same way, the double spiral data set is a set of points in $\mathbb{R}^3$, the first two coordinates telling you where the data is and the last digit telling you if it is a nought or a cross. The local structure is quite different from the case of the chess-board pattern, but the same machinery works. The set of noughts is locally described by a set of ellipses, and local groups of, say three ellipses may be represented by giving their location relatively, or this information may be inferred from the canonical UpWrite via moments of the points in $\mathbb{R}^6$. The relationship will in fact change slightly, but predictably, as the data gets further from the centre. Extrapolation is therefore possible. This is done automatically

[7] But watch out for the sequel.

by simply inferring the structure 'rule' and DownWriting to get the distribution of points which the top level coding will generate. Again, the classification is accomplished by the same machinery, we look at the virtual neighbouring data obtained from the DownWrite.

8.9 Occlusion and other metric matters

An important part of the problem of binding or chunking is the following: Suppose I have seen lots of squares and triangles separately, and I see three sides of a square: how do I arrange to look harder for the fourth side? After all, I should have some expectation that there is a fourth side, so would you, and so, one suspects would a moderately bright cockroach. Does this emerge from the formalism? How does one deal with the situation whereby we have two objects, one partially in front of the other and we recognise them both? This is, if you reflect upon it, a clever trick.

The case of three sides of a square where we have formerly seen all four will do to start with, since the general case of missing data is essentially the same. It will by now be apparent that these issues do not make sense except relative to a body of data, so we shall suppose that many squares and triangles of different sizes and orientations and locations have been provided to the system, and also a similar set of triangles. Now we present the system with a three sided square, that is, one with a single edge absent. The system has to decide if this is more like a square or a triangle. If it concludes that it is a square, it has to supply the missing side. A related problem is that we give the system a noisy square and require the system to clean it up.

Now the UpWrite of the three-sided square will have a zeroth component saying it has three elements in the set, a centroid consisting of six numbers, and a covariance matrix of 21 numbers. This supposes that we have a resolution radius big enough to encompass all three edges. The six centroid entries will contain first the average of the number of pixels in each edge; these numbers will be very close, squares being what thery are. The corresponding variance entry will therefore be small and the centroid entry will be the same as the length of each side in pixels. The next entry in the centroid wil be the centroid of the centres of the edges. This will not be in the centre of the square because one side is missing. The centroids of the covariance matrix terms in $\mathbb{R}^6$ will likewise be distorted from what we might have expected had the fourth term been added in. Likewise, the covariance terms will be affected by the missing edge.

If we look at the (four dimensional) manifold of squares in $\mathbb{R}^{28}$, the six dimensional manifold of all triangles in the same space, and the single point representing the UpWrite of the three-sided square, we can ask, to which manifold is the new point closest? And with this comes the associated question, how do we measure distances? This is clearly vital here: if we give high weight to the first component, we observe that all triangles have three sides, this object has three sides, therefore this object is a triangle. If we giver high weight to the centroids as well, we shall conclude that the triangle it 'is' must be equilateral. But if we give high weight to the covariance terms, we might complete the square instead of bending two sides to join the remote ends. Similarly, if we had used smaller neighbourhoods to get more levels of UpWrite, we should have been heavily influenced by the fact that we had two right angles in the three-sided square, and that right angles correlate with squares and not triangles. Which is the right way to go?

It depends on the data. There is no answer to the dilemma of which is right, either might be reasonable, it would depend largely on what kind of data has been seen before. If there were cases of triangles which had gaps in them at one or more vertices, the case for thinking it a deformed triangle would be strengthened. If there were squares with holes in one or more of the sides, the case for a square would be strengthened. In general there is no way to tell, and both are possible.

To see how to use the extra data, say lots of squares with bits missing, note that the manifold of squares is now made rather blurred along the outlines by the noisy data of partial squares. Local descriptions of the structure of this noisy manifold will give, to second order, local

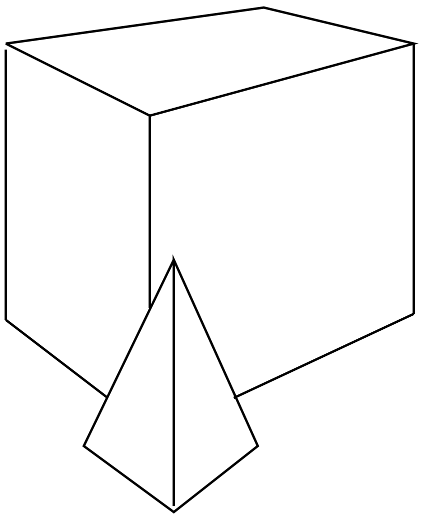

Figure 8.15: A simple occlusion problem.

quadratic forms near the centrally fitting manifold. We may use these to obtain an estimate of the metric in the region of the three-sided square. If deformations of this sort occur commonly, then the quadratic forms will have large extensions in the appropriate direction. We can think of them as giving estimates of the Riemannian Metric Tensor for a suitable metric. This solves, at least in principle, the issue of the best completion of the three-sided square. All we have to do is to find the closest and DownWrite it. Refinements using different levels of the UpWrite will occur to the reflective reader. It will be immediately apparent to such a reader, that excluding adventitious line segments to an existing square is essentially the same problem.

The occlusion problem extends this quite directly. If we have a pyramid occluding part of a cube,as in fig.8.15, first we find the pyramid.

This is a matter of deciding to exclude some elements in a neighbourhood, as referred to dismissively in the last sentence of the preceding paragraph. (It may be a little longer to compute it than to describe it.)

Next we remove the pixels belonging to the pyramid which we have by this time classified. We then take the object which is left and classify that. We discover that it may be completed into a cube by consistent and relatively small extensions at lower levels. This means we have to use the appropriate metric at the topmost level. The metric is learnt, either intrinsically from other parts of the same image or extrinsically by comparison with other images. Alternatively, we use a hierarchical probabilistic model to generate relative likelihoods of the two models given the data.

8.10 Neural Modelling

8.10.1 Self-Tuning Neurons

There is an interesting and suggestive experiment which has been carried out by Hubel and Wiesel, Hirsch and Spinelli, and extended by Blakemore. See the Bibliography for some sources, particularly the Blakemore paper. In the experiment, a cat has its head clamped and is kept fully conscious looking at a white disk free to rotate about an axis which is approximately the same as the optical axis of the cat's eye. A bar is drawn in black across a diameter of the disk. Its orientation may therefore be altered to any value.

A microelectrode is inserted into a carefully selected area of the visual cortex of the cat, an electrode fine enough to record the response of a single neuron. The neuron is monitored via the electrode as the disk is rotated. It is found that the firing rate of the neuron will suddenly increase at some particular orientation, going swiftly with angle from the idling rate to a value substantially higher.

We can plausibly argue that the neuron has somehow got a particular sensitivity to edges of the orientation of the bar. Random selection of neurons obtained by simply pushing the microelectrode through, gives a random set of orientations to which the neurons are tuned.

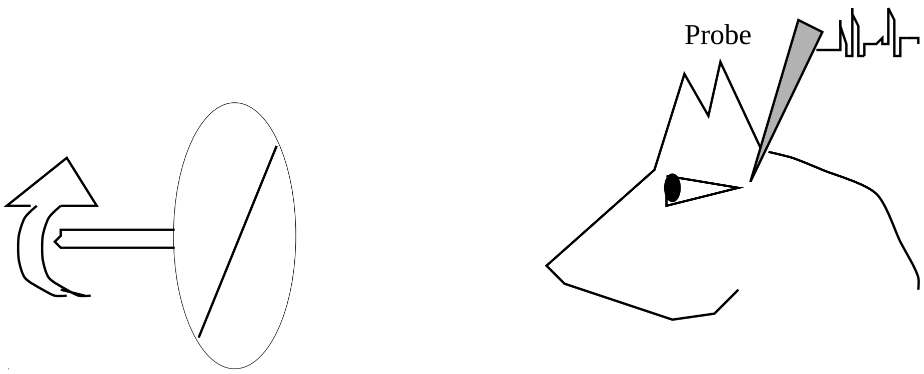

Figure 8.16: A cat looking at an oriented bar

Are the neurons in a ready trained state from the time when the cat is born, or do they acquire a particular orientation at some later date? And what makes them pick one orientation rather than another? These questions have been addressed, the 'cat in the hat-box' experiment described by Blakemore in the paper cited below, shows that when a cat opens its eyes in an environent of vertical stripes, neurons in this area become tuned to vertical edges to a large extent, when the cat is brought up in a horizontally striped environment, it has large numbers of neurons dedicated to responding to horizontal edges, and when it is swapped between boxes in its early life, it acquires neurons which are tuned either to horizontal edges or to vertical edges, but not both.

8.10.2 Geometry and Dynamics

Since this happens over fairly short times, it is natural to conjecture that the self-tuning phenomenon observed in the visual cortex of the cat occurs quiet generally and is related to learning and cognition.[8] The situation may be described more colourfully in terms of the neurons being attracted to the data; if we describe the state of the neuron by the input to which it responds most strongly, we can imagine it as a point sitting in the same space as the input data. And when an input datum is presented to the sensorium, the neuron 'moves' towards the datum, much like a dog being drawn irresistibly towards a rabbit when the latter sticks his head out of his hole. The neuron doesn't move physically inside the head of the cat, it changes its state so as to move in the state space. In the case of the edge detecting neurons, we draw a line and mark in a point to denote an angle of zero, another to denote an angle of 90 degrees. Then we put a datum in to the system and in the form of a vertical edge by lighting up the corresponding orientation, a point near the 90^o location. The neurons are not initially on the line, since there is no strongly tuned response to any orientation when the kitten first opens its eyes. So we may put the neurons initially at locations off the line representing the input states. The repeated flashing of data points at or near the 90^o location draws in the neurons however, until they are all close to the data presented. Similarly, if data points are presented near the 0^o location on the line, neurons respond by being attracted to the corresponding location. Neurons are attracted to data just as dogs are to rabbits.

There is some scope for enquiry concerning the dynamical laws governing the attraction of neurons to data or dogs to rabbits. If we have two rabbits alternately springing up in a field at different locations A and B, then if a pack of dogs is set loose at random initial positions in the field, what assumptions can we plausibly make concerning the motion? Well, one assumption is that the dogs can see the rabbits but not the other dogs. Presumably neurons can share a field of inputs, but neurons aren't data, and it seems unreasonable that they should be able to respond to each others states. On the other hand, there is a mechanism, lateral inhibition, whereby they can *hear* each other. If the strength of the output of one neuron is interpreted whimsically as the

[8]Blakemore conjectured it in the paper cited, but it probably struck a lot of people that this was a variant of a Hebbian learning rule.

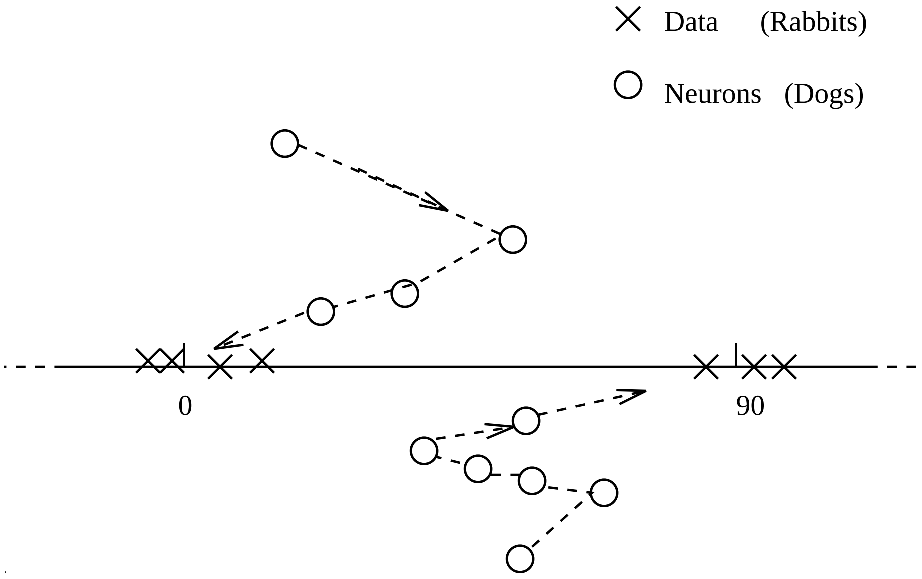

Figure 8.17: Neurons being attracted to data

dog barking, then many groups of neurons have output passed between them, so that the output of one is allowed to inhibit the effects of the common input. This allows for more precise tuning, and it also may be interpreted in the geometrical for which I have a decided preference.[9]

In dynamical terms then, the dogs move towards a rabbit when it stands up and presumably stop shortly after it goes down again. They cannot see each other, but they can see the rabbits, bark more frenziedly as they get closer, and can hear each others barking.

It makes sense for the dogs to move larger distances when they are closer to the rabbits and smaller distances when they are further away, for the reasons discussed in chapter five. If one contemplates a dog

[9]There is a belief in some quarters that God is a mathematician, or why did He make Physics so mathematical? And in particular, He must be a geometer, because most of Physics uses geometry. This is like wanting the Sun to shine at night when we need it and not in the daytime when it's light anyway. The fact is that the most powerful languages allow you to say more interesting and important things. *That* is why I use geometry.

which is attracted to each rabbit alternately, with the two rabbits, one at A and one at B, then the dog which simply jumps half way to the closest rabbit, winds up at neither but at the $\frac{1}{3}$ or $\frac{2}{3}$ point along the line joining the rabbits. Moreover a pack of dogs does the same thing. The pack coalesces to a single dog very rapidly, all in the wrong place. Whether discussing dogs in a field or neurons in a state space modelled on a linear space, we are faced with arguments against letting the law of attraction be what is known in informed circles as a contraction mapping. (See any good book on linear analysis.)

It is natural to suppose that the inhibition from surrounding dogs operates by reducing the size of any jump they might make towards a rabbit. We therefore obtain a fairly simple set of functions determining the motion of a dog towards a rabbit or a neuron towards a datum. The options for the qualitative behaviour are not great, as a more sophisticated mathematical analysis will demonstrate. Here I argue only at the intuitive level, which carries reasonable conviction with most.

Other effects which may be expected to affect the dynamics of neurons when attracted to data, are found in biological neurons: habituation dims the response of neurons to repeated data eventually, lability of neurons, their propensity to respond by change of state is believed to be reduced after some time. I suppose in the more formal model that the neuron jump size towards the datum is reduced by a 'fatigue' factor which is proportional to the number of data seen (or repeated) close to the present state of the neuron. This has the desirable effect of locking the neurons on close data and making them immune from the disturbing influence of remote data. A description of the formal system complete with rational function descriptions of the dynamics may be found in the paper by McKenzie and Alder cited in the Bibliography below.

8.10.3 Extensions to Higher Order Statistics

The proposition that a neuron might be in the business of finding the centres of clusters, in dynamical terms, has implications for what we have suggested is the mechanism for UpWriting, as well as general force in the context of unsupervised learning. This is rather different from the model of a neuron as binary classifier or adaptive threshold logic unit. It has far more similarity to the Kohonen network, but that is matter of which I shall not speak, as Gandalf puts it. But can this model be carried further? There are arguments for thinking it can.

A neuron in the visual cortex of a cat is presented with a flood of data concerning the edge orientations painted on the wall of its box: it can do temporal integration and the scene changes continuously over times shorter than the saccadic jumps of the eye. The inputs to a single neuron will almost invariably be correlated, and the neuron may be able to learn this fact. If the modification of synaptic transmitter substance or synaptic geometry, whatever records the effect of the datum on the neuron, is accomplished by a process which admits of interactions between synapses, then the correlations may also be learnt. To put it in geometric terms, the neuron sees not a single datum but a cluster of points in the input space, and the neurone may adapt its state to respond to their distribution. In simple terms, instead of responding to merely the location of the data points, it may be modelling higher order moments than the zeroth and first.

We know that the neuron has a response which is not infinitely sharp, and instead of representing it as a point or dog sitting in the space, trying to sit over a rabbit or cluster of rabbits, we might usefully think of it as conveying shape information. If for some reason we wanted to stop at seond order instead of first, we might visualise it as an ellipse, trying to do a local modelling, and the response to a subset of the data might be modelled by something not too unlike a gaussian distribution specified by the form represented by the ellipse. Thus a family of neurons may be dynamically achieving something rather like a gaussian mixture model of a data set. The complications avoided by stopping at second order make this quite pleasing, but of course there is no particular reason to think that real neurons do anything so simple In general we may however model them as computing moments up to some order less than n, for data in $\mathbb{R}^n$, where n is of order the number of afferent synapses of the neuron. This may be typically around 50,000, according to current estimates.

8.10.4 Layering

If a family of neurons sharing the same input field from some part of the sensorium may be crudely modelled as a dynamical and sequential form of a gaussian mixture model finding clusters in the data, we have some interesting grounds for speculation about statistical models. I shall avoid this and go straight to the propensity for neurons to form layers, and for the output of one layer to be the input to the succeeding layer. If each layer is accomplishing a model for the local clustering of its input, is a sequence of layers of such families implementing Syntactic Pattern Recognition? It is easy to see that this is not too hard a question to answer in the affirmative. The process of chunking I have described is readily seen to be easily implementable in layers of cluster finding entitities. The details need rather more formalism than I allow myself in this present book, but I claim it can be done. Moreover, the evolutionary process of layer formation can easily be accounted for along the lines discussed earlier, when I pointed out the variation in data that could be accommodated more effectively by having intermediate levels of UpWrite. We can essentially identify the levels of UpWrite with the layers of processing elements.

8.11 Summary of Chapter

The vision of the brain as a machine for, cognitively speaking, finding correlations in its inputs so as to be able to create entities from the sensorial chaos has its attractions. We are, it must be confessed, talking about what has been called *concept formation* here, although the term must fill any mathematician, or even anyone with a decent feel for language, with fury and loathing. The point of 'extracting features', to use another abominable contortion of the inarticulate, in this way, is of course to do more than classify, it is to predict, contingently, it is to survive. It would seem however, that the process of extracting and symbolising entities via layers of neurons has to be a fairly basic piece of machinery with which to accomplish survival in this present Universe.

In this chapter I indicated initially the drawbacks of the kind of static recognition I had used at the start of the book, and which are the commonplace methods of the times. I then drew attention to the inspiration of King Sun Fu that there was some similarity between the structure of natural language and that of certain images, and pointed out certain drawbacks in his attempt to provide algorithms for using this insight to understand images. I assured the reader that there was a formal theory couched in Category Theoretic terminology which allowed one to exploit the structures Fu had observed, and I undertook to convey the elements of this theory by practical exampole rather than formal mathematics. The remainder of the chapter consisted of my trying to fulfil that promise, with what success the reader is in a position to pass judgement, and in a final section I indicated that there was a link between the statistical methods of mixture modelling and what neurons might do in the Central Nervous System - in other words, paradoxically, statistical pattern recognition may be closer to what neurons do than neural net pattern recognition. I then suggested that there was a link between the UpWrite processes and the layering of neurons in the Nervous Systems of the higher animals.

The theory of cognitive processes and their implementation which this implies, may be tested empirically by building syntactic pattern recognition systems. There is some prospect of some of the work which has applied these ideas, actually making a few bucks for the present writer in the not too distant future. Very, very few bucks are likely to actually get past the tax man and his emulators. Of course, they would only be spent on Havana cigars, or even less respectable forms of self-indulgence, so perhaps it is no bad thing that the customers are all close to broke, or claim to be.

8.12 Exercises

I think you are entitled to think of your own applications and see if you can make them work using these methods. Some of the papers in the Bibliography may inspire you.

8.13 Bibliography

1. R. Gonzalez, M. Thomason *Syntactic pattern recognition: an introduction* Addison-Wesley Pub. Co., Advanced Book Program, 1978.

2. K. S. Fu *Syntactic pattern recognition and applications* Prentice-Hall, 1982.

3. T. Pavlidis *Structural pattern recognition* Springer-Verlag, 1977.

4. C. Blakemore *Developmental Factors in the Formation of Feature Extracting Neurons* , pp105-114, in: *Feature Extraction by Neurons and Behaviour* Ed. Gerhard Werner, MIT Press, 1975.

5. P.McKenzie, M.D. Alder, *Initialising the EM Algorithm for use in Gaussian Mixture Modelling.* Pattern Recognition in Practice IV, Vlieland, 1994. To Appear.

6. M. Alder *Topological Stochastic Grammars* in the book *Syntactic Pattern Recognition* Ed.Simon Lucas. To Appear.

7. G. Lim, M.Alder, C.deSilva *Syntactic Pattern Classification of Moving Objects in a Domestic Environment.* Pattern Recognition in Practive IV, Vlieland, 1994. To Appear.

8. R. McLaughlinn, M. Alder *Recognising Cubes in Images.* Pattern Recognition in Practice IV, Vlieland, 1994. To Appear.

9. M. Alder *Inference of Syntax for Point Sets* Pattern Recognition in Practive IV, Vlieland, 1994. To Appear.