Chapter 6

Continuous Dynamic Patterns

The first chapter of this book gave an overview from a considerable altitude, with all the lack of detail that compels. The second chapter got stuck into practical issues of measurement of images as a sort of antidote, since in real life much hangs on the details and they need to be treated squarely. The third chapter was so theoretical as to alarm many a simple soul, but was necessary for the thinking man or woman in order to prepare them for a healthy degree of doubt about what follows, and some later discussion on neural models. The fourth chapter simply gave some convenient recipes for actually doing the job of recognising standard (static) patterns by statistical methods, and the last chapter explained how the standard neural nets work and how they accomplish pattern classification, when they do.

In the present chapter we start to deal with the problem of dynamic patterns, of things that change in time. Examples are speech, on-line character recognition, and such things as telling two missiles or two people apart, using knowledge of the way they move. I shall defer such issues as looking at video-images of moving objects later, until we have addressed the issue of how to work out what objects are in an image. In the present chapter I shall, in proper generality, discuss the issue of classifying trajectories in $\mathbb{R}^n$ which arise from sampling some continuous process. In the next chapter I shall discuss issues arising from trajectories which occur in a discrete space of symbols known as an alphabet[1].

I shall start by discussing a simple and direct approach to Automatic Speech Recognition, the method being a staple of practical single word recognisers. This practical discussion will be followed by a short survey of the issues involved in continuous speech recognition. This problem is too hard to do much with at the present time, but the reasons it is so hard are interesting and worth thinking about.

The issue of *noise* in dynamic pattern recognition requires special treatment, it is something to which any measurement system is liable, and needs to be confronted. A great deal of engineering literature is concerned with this. I shall go briefly into the issues of stochastic processes or random time series and filters. I shall approach this from the engineering end rather than the statisticians end, although both have made contributions.

Since filtering theory and Automatic Speech Recognition (ASR) are both huge subjects in their own right, and quite impossible to cover in this book, the treatment will be confined to a survey of the basic ideas and pointers to the literature. The diligent reader will then, be in a position to write simple word recognition programs, will be in a position to understand and use samples of such programs on the accompanying disk, and will begin to understand the profound difficulties of the area.

Finally, I shall outline alternative approaches to Speech Recognition and discuss related problems.

---

[1]If this looks like a concession to the Artificial Intelligentsia, and a descent into rather crude types of representational machinery, this is an error, and I shall put you straight in later chapters, but it has to be done, if only because continuous trajectories are reduced to discrete trajectories a lot of the time. There are other and better reasons, as the persistent reader will discover.

6.1 Automatic Speech Recognition

6.1.1 Talking into a microphone

If you say a word, perhaps the name of a digit, for example 'one', into a microphone, then it is straightforward to sample and digitise the resulting signal, and feed it into a computer as a longish sequence of numbers measuring the voltage generated by the microphone and your voice. Typically, a word may take one third to half a second to enunciate, and the signal is sampled perhaps twenty thousand times a second, giving around seven thousand numbers. Each number will be quantised to perhaps 12 or 16 bit precision. Thus we may be looking at a data rate of around 30 to 40 kilobytes per second. This present paragraph, would, if spoken at a reasonable reading rate, occupy over two megabytes of disk space. If printed, it would occupy around a kilobyte. There is therefore a considerable amount of compression involved in ASR.

There are various methods of proceeding from this point, but the most fundamental and conceptually simplest is to take a Discrete Fourier Transform (DFT) of a short chunk of the signal, referred to as the part of the signal inside a *window*. The FFT or Fast Fourier Transform of Cooley and Tukey accomplishes this with admirable efficiency and is much used for these purposes. I have already discussed the ideas involved in taking a Fourier Transform: there are two ways of thinking about it which are of value. The first is to imagine that we have a sound and something like a harp, the strings of which can resonate to particular frequencies. For any sound whatever, each string of the harp will resonate to some extent, as it absorbs energy at the resonant frequency from the input sound. So we can represent the input sound by giving the amount of energy in each frequency which the harp extracts, the so-called energy spectrum. This explanation serves to keep the very young happy, but raises questions in the mind of those with a predisposition to thought: what if the sound changes this spectrum in time? If it sweeps through the frequency range fast enough, there may not be enough time to say it has *got* a frequency, for example. And what kind of frequency resolution is possible in principle? These considerations raise all sorts of questions of how filters work that need more thought than many writers consider worthwhile.

The second way to look at the DFT and FFT is to go back to the old Fourier Series, where we are simply expanding one function from $[-\pi, \pi]$ to $\mathbb{R}$ in terms of an orthogonal set of functions, the sine and cosine functions. This is just linear algebra, although admittedly in infinite dimensions. Now we have to reckon with two issues: one is that the signal is sampled discretely, so in fact we have not an algebraically expressed function but a finite set of values, and the other is that the time window on which we do our expansion is constantly changing as we slide it down the signal. What sense does it make to take out a chunk of a continuously changing signal, pretend it is periodic, analyse it on that assumption, and then slide the window down to a different signal and do the same again? Of course, these two ways of looking at the problem give equivalent results, principally some honest doubt as to whether this is the way to go. Alternative transforms such as the Wigner transform and various wavelet families are available for those who want to follow this line of thought. They are beyond the scope of this book, but the reader who is of a reflective disposition will be ready for them when he meets them. I shall skip these interesting isues on the grounds that they are somewhat too technical for the present work, which is going to concern itself with more mundane matters, but the reader needs to know that there are problems about fixing up the window so that the FFT gives acceptable answers reasonably free of artefacts of the analysis process. See the standard works on Speech Recognition and many, many issues of the IEEE transactions on Signal Processing, and the IEE Proceedings part I for relevant papers.

We take, then, some time interval, compute the FFT and then obtain the power spectrum of the wave form of the speech signal in that time interval of, perhaps, 32 msec. Then we slide the time interval, the window, down the signal, leaving some overlap in general, and repeat. We do this for the entire length of the signal, thus getting a sequence of perhaps ninety vectors, each vector in dimension perhaps 256, each of the 256 components being an estimate of the energy in some frequency interval between, say, 80 Hertz and ten KHz. The FFT follows a divide and conquer strategy, and when it divides it divides by two, so it works best when the number of frequency bands is a power of two, and 256 or 512 are common. Instead of dealing with the raw signal values in the window, we may first multiply them by a window shaping function such as a Hamming Window, which is there to accomodate the inevitable discontinuity of the function when it is regarded as periodic. Usually this just squashes the part of the signal near the edge of the window down to zero while leaving the bit in the middle essentially unchanged.

Practical problems arise from trying to sample a signal having one frequency with a sampling rate at another; this is called 'aliasing' in the trade, and is most commonly detected when the waggon wheels on the Deadwood Stage go backwards, or a news program cameraman points his camera at somebody's computer terminal and gets that infuriating black band drifting across the screen and the flickering that makes the thing unwatchable. There is a risk that high frequencies in the speech signal will be sampled at a lower frequency and will manifest themselves as a sort of flicker. So it is usual to kill off all frequencies not being explicitly looked for, by passing the signal through a filter which will not pass very high or very low frequencies. Very high usually means more than half the sampling frequency, and very low means little more than the mains frequency[2].

The 256 numbers may usefully be 'binned' into some smaller number of frequency bands, perhaps sixteen of them, also covering the acoustic frequency range[3]. The frequency bands may have their centres selected

[2]The same effect can often be obtained by using a cheap microphone, but this approach is thought unprofessional by most engineers. They prefer to get an expensive microphone and then put a filter in to make it behave like a cheap one.

[3]By comparison, the human ear has a cochlea, a rubbery tube filled with fluid, and a membrane down the middle of it, which vibrates when sound strikes the ear, and which because of varying properties of the membrane, responds in different places to different frequencies. The vibration of the basilar membrane is monitored by around 30,000 hair cells. It is thought that this performs something like a fourier transform on the speech signal, but it is known that much more happens than this.

judiciously in a more or less logarithmic division of the frequency range, their widths also adjusted accordingly, and the result referred to as a simulated filterbank of sixteen filters covering the audio spectrum. Alternatively, you could have a bank of 16 bandpass filters, each passing a different part of the audio spectrum, made up from hardware. This would be rather old fashioned of you, but it would be faster and produce smoother results. The hardware option would be more popular were it not for the tendency of hardare to evolve, or just as often devolve, in time. The so called 'Bark' scale, slightly different from logarithmic and popularly supposed to correspond more closely to perceptual differences, is used by the more sophisticated, and, since speech has been studied since Helmholtz, there is an extensive literature on these matters. Most of the literature, it must be confessed, appears to have had minimal effect on the quality of Speech Recognition Systems.

Either of these approaches turns the utterance into a longish sequence of vectors in $\mathbb{R}^{16}$ representing the time development of the utterance, or more productively as a *trajectory*, discretely sampled, in $\mathbb{R}^{16}$. Many repetitions of the same word by the same speaker might reasonably be expected to be described as trajectories which are fairly close together in $\mathbb{R}^{16}$. If I have a family of trajectories corresponding to one person saying 'yes' and another family corresponding to the same person saying 'no', then if I have an utterance of one of those words by the same speaker and wish to know which it is, then some comparison between the new trajectory and the two families I already have, should allow us to make some sort of decision as to which of the two words we think most likely to have been uttered.

Put in this form, we have opened up a variant of traditional pattern recognition which consists of distinguishing not between different categories of point in a space, but different categories of *trajectory* in the space. Everything has become time dependent; we deal with changing states.

Note that when I say 'trajectory' I mean the entire map of the time (discretely sampled) into the space, and not just the set of points in the

The difference between 16 and 30,000 is a little disturbing.

path. These are very different things. If you walk across the highway and your path intersects that of a bus, this may not be important as the bus may have long since gone by. On the other hand if your *trajectory* intersects that of the bus, then unless it does so when you are both at rest, you are most unlikely to come back to reading this book or, indeed, any other. I am prepared to reconstruct the time origin of any two utterances so that they both start from time zero, but one may travel through the same set of points twice as fast as another, and the trajectory information will record this, while the *image* of the two trajectories will be the same. A trajectory is a function of time, the path is the image of this function[4]. If you want to think of the path as having clock ticks marked on it in order to specify the trajectory, that's alright with me. Now it might be the case, and many argue that in the case of speech it *is* the case, that it is the path and not the trajectory that matters. People seldom mean it in these terms, since the words /we/ and /you/ are almost the same path, but the direction is reversed. Still, if two trajectories differ only in the speed along the path, it can be argued that they must sound near enough the same. This may or may not be the case; attempts to compare two trajectories so as to allow for differences in rate which are not significant are commonly implemented in what is known as *Dynamic Time Warping* (DTW). I shall not describe this method, because even if it applies to speech, it may not hold for everything, and DTW is more commonly replaced by Hidden Markov Modelling these days.

On first inspection, one might argue that a trajectory in one space is simply a point in a function space. This is true, but not immediately helpful, as different trajectories may not be in the same space, as function spaces are traditionally defined. It is rather hard to put a sensible metric structure on the set of maps from any interval of the real numbers into $\mathbb{R}^n$ without any other considerations. So the abstract extension from points to trajectories needs some extra thought, which may depend upon the nature of the data.

It would be a mistake to think that binning the power spectrum into

[4]If you can't tell the difference, don't cross any roads until you have thought about it.

some number n of intervals is the only way of turning speech into a trajectory in $\mathbb{R}^n$, there are others involving so called cepstra or LPC coefficients which are believed in some quarters to be intrinsically superior.

6.1.2 Traditional methods: VQ and HMM

It appears to be an article of faith with practitioners of Automatic Speech Recognition (ASR) that an utterance is a string of phonemes and a phoneme is a region of the space of speech sounds. This is not quite what a phonetician means by a *phoneme*, but phoneticians are people who know, roughly, how to tell each other how to hold their faces so as to make noises like foreigners talking, and are generally quite unable to say much about the speech space. The anthropologists who feel the clash of disparate cultures have nothing on a speech engineer going to a meeting of phoneticians or vice-versa. The more courageous find it exhilarating, the less, terrifying.

If we follow, temporarily and without prejudice, the ASR engineers, then we solve the problem of recognition of trajectories by first chopping up the space into regions, and labelling each region with a symbol. Ideally, the region of the space would correspond to what a phonetician would call a phoneme. In practice we may have the region rather arbitrary in position and shape. A trajectory, complete with clock ticks, then gets turned into a sequence of symbols, the labels of the region it is in at each clock tick. If we have a large number of utterances, we obtain from our trajectories, and way of chopping the space up into regions, a corresponding number of symbol strings. We then try to model the strings corresponding to a word such as 'yes' by means of a *stochastic grammar*, the strings corresponding to the word 'no' by a different stochastic grammar, and a new word has its probabilities computed by applying each grammar in turn. Then we use Bayesian arguments to get back from the probability of each grammar having generated the string, to the probability, given the string, of one grammar or another having been the source. A *stochastic grammar*, to anticipate the next chapter somewhat, is an algorithm which assigns a probability to a string of symbols to say how likely it is to occur in a sample from a particular language, and a (stochastic) *language* is a set of strings of symbols with a frequency attached to each of them. It is clear that a certain collection of assumptions about the nature of speech have been employed to come to this approach.

Chopping up the space into lumps is known in the trade as *Vector Quantisation*, which no doubt sounds a little classier put that way[5], and may be accomplished in a variety of ways. Some of these are so arbitrary and whimsical as to be embarrassing. One of the better ways is actually to take the speech data as just a big stack of points in $\mathbb{R}^{16}$ or whatever, and use a gaussian mixture model with some judiciously chosen number of gausians. Then each gaussian may be used to compute the likelihood of a given piece of speech sound having been obtained from it, and the largest likelihood yields the name of that particular gaussian as the symbol. Alternatively we can use the Mahalanobis distance from each of the centres. This gives somewhat better results than the conventionally used LBG algorithm, a variant of *k-means clustering* which I shall now describe.

Suppose we have a set of points in $\mathbb{R}^n$ and we wish to chop them up into clusters. We first decide how many clusters we propose to have, say k, and choose ourselves k different kinds of points, call them centres. Now we throw the centres down at random in the space, or perhaps uniformly, and then partition the points so as to assign each point to the closest centre, thus constructing subclusters. Then we move the centres to the centroids of the subclusters, and recompute the assignment of points to subclusters, and the shifting of centres to centroids of subclusters, until the process stabilises. We may have to get rid of centres which coincide, in extreme cases. This may be said in algebra:[6]

Given $D \subset \mathbb{R}^n$, D finite and $K = \{x \in \mathbb{Z} : 1 \leq x \leq k\}$, let $f_0 : K \longrightarrow \mathbb{R}^n$ be an initial assignment of centres; this might be uniform on the region

[5] So much so that one of my more ebullient colleagues has been found threatening to vector quantise people who argue with her.

[6] Watch out for the next footnote.

containing D or random. Now let $P_m = p_1, p_2, \cdots p_k$, for $m = 0$ in the first instance, be the partition of D which has

$$\mathbf{x} \in p_i \Leftrightarrow \forall j \neq i ||\mathbf{x} - f_m(i)|| \leq ||\mathbf{x} - f_m(j)||$$

Now obtain, in general, f_{m+1} from f_m by putting

$$\forall i \in K, f_{m+1}(i) = \frac{1}{|p_i|} \sum_{\mathbf{x} \in p_i} \mathbf{x}$$

and similarly obtain another partition P_{m+1}.

It turns out to be rather too dependent on the initial assignment of centres to leave the thoughtful reader altogether happy.

In The LBG (Linde, Buzo, Gray, q.v.) algorithm, the k means are obtained by a splitting process: first we go to the centroid of the whole data set D. Then we choose a random line and split the centre into two centres, each moved apart a little distance along the line. Then we partition the data set by assigning each point of it to the closest centre, just as before. Then we move to the centroids of the partitioned bit, as before. And we proceed to replicate the splitting process until we get bored or run out of points. This leads to k being always a power of two. The method again is dependent on the orientation of the lines used and often leads to bad clusters, which is to say clusters that don't look like natural clusters to the naked eye. If we generate the data by a gaussian mixture, we can get some exceedingly unappealing results for the final LBG assignment of points to clusters, given that we know how we in fact did it. In abstract terms, this method places more significance on the metric than is normally warranted. It is true that many metrics will agree about where the centroid of a set of points should be, but they usually have a depressing indifference to the distribution of the data. Fitting gaussian clusters and computing Mahalanobis distances or likelihoods actually gives somewhat better performance in recognition systems, which suggests that reality doesn't love the standard metrics all that much. We can think of the quadratic form as being a local estimate of the metric tensor for the generating process, or if you prefer, the data is trying to tell you something about what 'close' means, and it would be advisable to listen to it.

As a Vector Quantisation (VQ) system, gaussian mixture modelling by fairly crude means is as quick and easy and more defensible than most. The simplest method is to replace the k-means approach by computing not just the mean of the subcluster assigned to the centre, but also its covariance matrix, and then to use *that* in order to compute a Mahalanobis distance when doing the next partitioning of the data points. We can think of this as taking a local estimate of the metric, or a second order approximation to the distribution of data points. You will note that this is a sort of quick and dirty version of the EM algorithm.

The labels for the different parts of the space are known as *codewords* and the list of them all is known as a *codebook*, for reasons which derive from coding theory. The general procedure of replacing points in a space by some set of central approximating points is used in many areas, for example in image compression, and it is the main idea in coding against noise, where the hope is that if you send the centres and they get corrupted, the points will be moved a short distance, but providing we choose to replace each point by the closest centre, we usually get back to where we started from. In this case a point is usually in a finite space of bytes.

It is common to try to estimate the extent to which replacing the actual point by the closest centre has deformed the trajectory, by adding up the squares of the distances of the actual points from the closest centres.This is called the *distortion measure* of the codebook. Of course, this presupposes that we know the right way to measure distances in the space, which we don't, so there is a certain fatuity in this approach. In image compression, for example, the Discrete Cosine Transform or DCT, a relative of the DFT which expands in terms of cosine functions only, is used on the two dimensional signal that is an image, and then the resulting function is quantised. Now the DCT produces a spectrum of spatial frequencies in the original image, and the eye is relatively insensitive to high spatial frequencies and low spatial frequencies, and is most sensitive somewhere in the middle. This is telling you that the euclidean metric on the space of DCT transforms is a particularly useless one to use if you want to say when two images look similar to

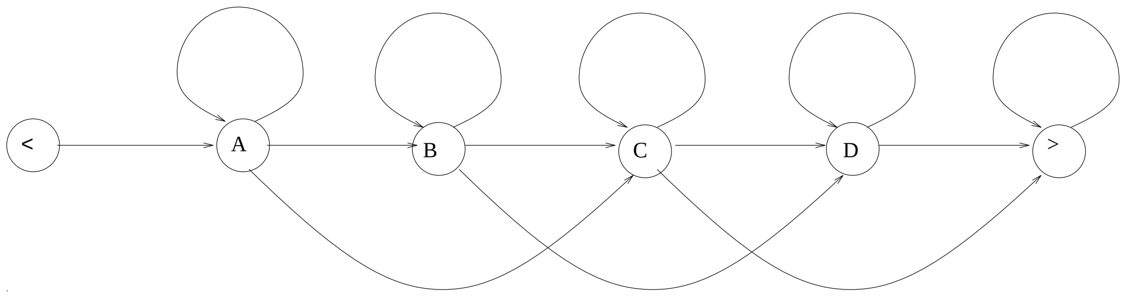

Figure 6.1: A Markov Process for spoken word recognition.

human beings. Nor is there any reason to believe that the euclidean metric on a space of power spectra is of much use as a measure of acoustic similarity as perceived by people. So attempts to obtain distortion measurements by this method betray a certain naivete on the part of those making them. There's a lot of it about.

Having now chopped the space up into lumps and replaced the original trajectory by the sequence of codewords to which each point of the trajectory is closest (using, of course, a Mahalanobis distance in the case of the gaussian mixture modelling) we have the problem of representing the resulting strings by means of *stochastic grammars.* The stochastic grammars most commonly employed are known as *Hidden Markov Models* or HMMs for short. A hidden Markov model is offered here, heuristically, as a description of the vocal tract going from one state to another. Typically ASR workers use a state diagram such as $Fig.$6.1.

Here the states have been labelled for ease of talking about the diagram.

Variants having more states or additional arcs (such as from A to D) are common.

We turn this diagram into a stochastic grammar for a set of strings, a language sample from a stochastic language, over the alphabet of labels of the regions into which the space has been chopped up into lumps, sorry, vector quantised. The stochastic grammar is obtained as follows.

Imagine someone in the next room to you stepping over the arcs represented by arrowed lines, between discs represented by the circles of Fig.6.1. Let them be supposed to have a handful of coins or dice or other means of generating random numbers. Suppose the diagram over which they trip their light fantastic, has been augmented in two ways. The first is that for each arc out of a node or disk, there has been a probability assigned. In going from node $<$ to node A there is no choice, so we can put a probability of 1 on this arrow, but once at node A we have three choices, jumping up in the air and coming back down on node A, hopping to node B, or skipping to node C. In order to decide which to do, your colleague throws a die or dice, and makes a decision depending on the outcome. If the three probabilities (AA), (AB), (AC) were the same, then your friend might hop if the die showed one or two, skip if it showed three or four, and jump if it showed five or six. The probabilities out of each node all add up to one, and we shall suppose that every node with an arc leading out of it has had probabilities assigned to it.

The other augmentation is to suppose that for each arc there is a probability distribution over the alphabet which assigns a probability of your friend shouting out each symbol. Again, having chosen probabilistically an arc to move along, your friend now has, while moving along it, to make another throw of coins or dice in order to decide which symbol to call out. Tricky, but not impossible. You may recognise this as a first order markov process for the grammar.

As your friend next door starts from the $<$ node and proceeds through the diagram, you hear the names of symbols being shouted out (along with the clatter of falling dice and coins), with a pause as she comes to the end node, >, and marches back to the beginning to do it again.

The system may be simplified, at modest cost, by ensuring that your friend calls out a symbol on arriving at a node, which means that the probability distributions over the symbols of the alphabet are the same for any two arcs terminating in the same node.

It would be easy to write a program to do the same, and less taxing for your friend.

The thing that is *hidden* about this markov model, is that you hear the string of symbols, but you don't know by which particular path through the network it was obtained. It may be possible that each string has a unique *parse*; that is to say, for any string there is either no way it could be obtained, or there is a unique sequence of hops, skips and jumps which was responsible. On the other hand there might be a large number of different possible routes which could have yielded the string.

To set the model up by choosing probabilities assigned to arcs or symbols according to one's whim is easy enough but unprofitable. The question of interest is, suppose we have a diagram such as the last figure and a set of strings, how do we select the probabilities of arcs and symbols so as to get a good stochastic grammar for our sample? And other questions of interest are, where did that diagram come from, and is there a better choice? Avoiding the embarrassment of the last two questions *pro tem.*, let us suppose that we have obtained the diagram itself by inspecting the backs of the tablets on which the ten commandments were inscribed, but that the creator has neglected to insert the probability distributions (stored usually as tables and called the A-matrix and B-matrix in the literature). We may obtain some suitable numbers by the following procedure[7] which the astute reader

[7] Described in English rather than algebra. Algebra has the merit of precision and the drawback that it can take a long time to work out what the underlying ideas are, sometimes rather more than a lifetime. It is generally rather easier to turn an idea from English into algebra or a suitable programming language than to go the other way round, but you may memorise algebra and get the illusion that you understand it when you have merely learnt it. Engineers sometimes use algebra

may recognise.

The Baum-Welch and Viterbi Algorithms for Hidden Markov Models

Start off with an arbitrarily chosen set of probabilities of choosing an arc from each node except the last, and another set of arbitarily chosen probabilities for each symbol being emitted on arriving at a node. Suppose you have to hand your sample of strings allegedly produced by this markov model. Now select the first string from the set of samples, and trace it through the diagram. If this cannot be done, some of your probabilities must have been made zero, which was naughty of you, go back and make them all positive. If it could be done in a lot of different ways, make a list of all of them. Since it is a finite length string and a finite diagram, there can only be a finite set of possibilities.

Each way of laying down the string from $<$ to $>$ through the diagram can have a probability computed for it, based on the arbitrarily assigned probabilities for transitions and emissions of symbols. The probabilities of choosing the sequences of arcs are readily computed by multiplying all the arc probabilities together, and the probability of having emitted a symbol at each arc, given that the arc was selected, can also be multiplied in at each symbol. So there is a net probability that the string was generated by any given parse, and we can add these together for all the different possible parses of the string. There are two approaches next, the *Viterbi* approach is to select the most probable parse and use that. The other approach due to Baum and Welch is to split up the string into fractional strings according to the probability of each parse, and lay down fractional strings over the diagram. If one imagines a diagram such as the above figure, and a string with symbols as beads on a piece of thread, the symbols have to get allocated to arcs or nodes in a sequence which lays out the thread over the diagram. In the case of the Baum-Welch method, we have lots of copies of the same threaded

---

as Lawyers and Medical men use Latin: to impress the peasantry and justify their salaries. Also, it conceals the fact that they can't spell and have awful handwriting.

necklace of symbols, each with a weight attached to it, the sum of the weights being one. Each of them gets layed out on the diagram, making for lots of fractional threads everywhere in the Baum-Welch case, and just one in the Viterbi case. We maybe ought to make the threads we lay down on the diagram a nice striking colour, say scarlet, to contrast with the sober black of the network connections.

This is repeated for each string of the sample. This gives a diagram covered with a lot of scarlet threads. We then proceed to reestimate the probabilities using these threads. To reestimate the probability of travelling from A to C, we look at all the bits of string we have laid down, and count the total number leaving A. Of this total number (which for the Baum-Welch algorithm may not be an integer) we count the fraction going to C, and this fraction becomes the new probability assigned to the arc from A to C. Similarly for all the other arcs. Given that we go from A to C, we now count up the (possibly fractional) occurrences of each of the symbols that we have laid down along that path. The ratio of each symbol count to the total number of counts gives the probability reestimate for the symbol.

From the initial estimate and the sample of strings, we have a reestimate of the probabilities on the diagram. From the first, possibly ill chosen selection of probabilities, we have obtained a more plausible set which have done a better job of accounting for the actual data. We now do it again with the reestimate as our starting point. We repeat the process until the numbers don't change and claim that this gives a maximum likelihood model for the data given the network topology.

The Baum-Welch algorithm is clearly a particular case of the EM algorithm, which can be used for much more than just mixture modelling. The description given in English may be replaced either with some code or some algebra, but if you write your own code, making up a few little threaded necklaces of strings and working out where to put them will be much the quickest way. People of inferior intellects who have trouble with such arcana as reading and writing in natural languages may prefer algebra or C; such folk may find the above description confusing in which case they may find alternative formulations via the bibliography.

The process of effectively calculating all the probabilities may be organised, and the way of organising it is known as the *forward-backward algorithm.*

6.1.3 Network Topology and Initialisation

There are, as anybody who has married, had children or been in politics is aware, some questions it is better not to ask. One of these is, where does the diagram of Fig.6.1 come from? Another is, is a random initialisation of EM, or Baum-Welch, really a good idea? I shall ask them anyway.

If we suppose that the vocal tract has some fixed number of states and that any word consists of some definite number of them and we can decide which, but that the time spent in each state varies, and that sometimes states are skipped, then a diagram such as Fig.6.1 becomes almost defensible against a weak and incoherent attack. If we take the view that it is a simple model which compresses strings and works, sort of, then the only useful attack upon it is to improve it, which has been done successfully.

It is found in the case of Hidden Markov Models, just as for Gaussian Mixture Modelling, that EM tends to be sensitive to initialisation, and for the former case, good initialisations for different words are passed between nervous looking individuals in seedy looking bars in exchange for used notes of small denomination. At least, nobody publishes them, which one might naively think people would. But this is based on the assumption that people publish papers and books in order to inform readers about useful techniques, and not to impress with one's supernal cleverness. To be fairer, initialisations are word and sample dependent, so you may just have to try a lot of different random ones and see which work best.

6.1.4 Invariance

Although not as up to date as possible, it is as well to remember that VQ and HMM is the staple of single word recognition systems and rather more. It works reasonably well provided that the system is trained on the speakers it is proposed to use it on; this makes it a dead loss for selling to banks to allow people to tell an automatic system their mastercard number. Even a modest amount of speaker independence has to be purchased by training on a whole raft of people and hoping that your speaker is in there somewhere, or at least a clone or close relative of the same sex. The standard *el cheapo* boxes for single word recognition which you buy for your PC or MAC work this way. They often solemnly advertise that the system will only respond to your voice, as though this is a great virtue, but in fact it doesn't even do that particularly well.

I have mentioned one way of improving the VQ side of this approach. Another is to try to explicitly recognise the gaussian mixture model is an estimate for a continuous *pdf* over the states. The only difference in practice is that instead of choosing the closest gausian distribution, we may want to take account of the weights associated with it and choose likelihoods. This yields a semi-continuous Hidden Markov Model. There are some simple ways to improve the HMM side of it too: one can go further and try to model the probability density function for trajectories in the space by directly training a gaussian mixture model on the data consisting of graphs of repetitions of utterances of a single word, and then computing likelihoods for each new trajectory relative to the family of models. After all, it is the set of trajectories with which one is concerned, this is the data. The natural desire to model the *pdf*s for the different words is thwarted in practice by the fact that there are a number of transformations that are done to the data, and this means we never have enough data to get sensible estimates. It is rather as if you wanted to tell a letter A from a letter B when they had been scribbled on a wall, by storing information about which bricks were painted on. You don't get much help from the replication of A's if they occur in different places, when represented in this form, whereas if you had been able to get a description which was shift invariant, it would be possible to pool the data provided by the different A's and B's. But that requires knowing the transformation we want to factor out, and in general we don't. The only thing the prudent human being can do to make life easier for his automatic system is to build in all the invariance he expects to be useful.

One thing, then, that ought to make the ASR practitioners nervous is the big, fat, fundamental assumption that phonemic elements are regions of the space.

To see what other possibilities there are, consider the problem of asking a small girl and a large man to say the words 'yes' and 'no'. They have different sized and shaped vocal tracts and it is clear as day that their utterances will occupy very different regions of the speech space. Had we given them a piece of chalk apiece and asked them to write the words upon a wall instead of speaking them, we could reasonably have expected that the two dimensional trajectories of the movement of the chalk would also have occupied different parts of the space. Trying to distinguish the words 'yes' and 'no', when written down, by looking to see which bricks they are written upon would not, on the face of things, be a good way to go.

It is clear that the problem arises because the fundamental assumption is clearly wrong in the case of handwritten words, and it might be just as wrong for spoken words too. It isn't the regions of the space you pass through, it's the shape of the trajectory, in some scale invariant way, that determines what a written word is. The HMM if used on the bricks of the wall to determine written words might more or less work for one little girl and one large man, but would flip when confronted with an intermediate size writer.

The problem is that the written word is invariant under both shifts and scaling. And the spoken word is also invariant under some part of a group of transformations: simply making the sound louder throughout will not change the word. Lowering or raising the pitch, within reason, will not change the word. And it is easy to believe that continuous

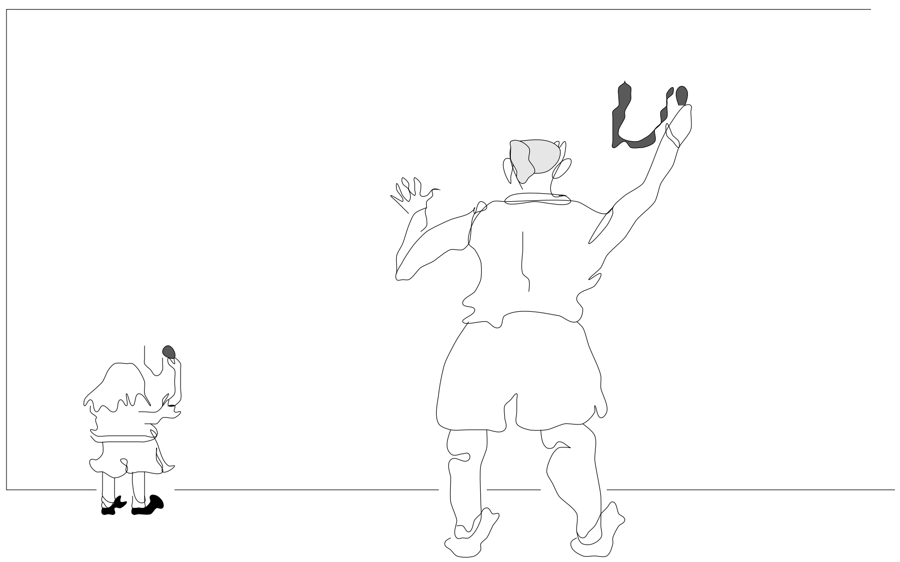

Figure 6.2: Translation and scale invariance.

enlargement of the vocal tract won't change the word, either. Adding white noise by breathing heavily all over the microphone, as favoured by certain pop-stars, might also be said to constitute a direction of transformation of a word which leaves the category invariant.

If one excises vowels from the speech of many different speakers, plots the short segments of trajectory which one then obtains as short sequences in $\mathbb{R}^{16}$, and then projects down onto the screen of a computer, one sees what looks like a good approximation to a seperate gaussian cluster for each vowel. But the cluster has quite a large extension, close vowels sounds appear to overlap, and the distance between centres is not as big as the extension along the major axes. It is tempting to try to split the space into a part which differs in what vowel is being articulated, and an orthogonal part which tells you who is speaking, how loudly, and how hoarse they are, along with much other data which seems to be largely irrelevant to working out what is being said. It would be nice to try to decompose the space in this way, and it is possible to attack this problem, if not cheap.

Consider, as another basis for doubting the fundamental assumption when applied to speech, the various sign languages used by deaf people and others. These consist of stylised gestures of various sorts, performed in succession, at rates comparable with ordinary speech as far as the rate of transfer of ideas is concerned. The question is, are the target gestures actually made in Sign? It can be plausibly argued that they are hardly ever actually 'stated' except in teaching the language, and that fluent signers only sketch out enough of a gesture for the signee to be able to decode it, after which they move on rapidly to the next gesture. So signed 'words' are idealisations never actually attained. Moreover, the space of gestures references such things as heads and faces, and is thus not readily reducible to trajectories in an absolute space.

Speech has been described as 'gestures of the vocal tract', and we have to consider the possibility that it resembles sign languages in its trajectory structure rather more than meets the eye. If so, the crudity of the VQ/HMM model becomes embarrassing. One would have to be desperate to suggest this approach to sign languages. In fact the problem of automatic reading of sign via a camera is an interesting if little explored area for research which might well illuminate other languages. Perhaps the difficulties are simply more apparent for sign languages than for spoken ones.

In the case of speech, it is generally argued that the path is what is wanted, not the trajectory, since going over the same part of the space at a diferent rate will not change what is said, only how fast you say it. This gives another family of transformations of the space of trajectories which would, if this view is correct, leave the word itself invariant.

In general the problem of speaker independent speech recognition is a profoundly difficult one and one which still needs critical thought, as well as more creative approaches.

For confirmation of some of the pessimistic observations of this section, see the program *fview* by Gareth Lee on the ftp site ciips.ee.uwa.edu.au, referenced in the bibliography. Pull it back and play with it: it is the best free gift of the decade.

6.1.5 Other HMM applications

A Hidden Markov Model is merely a device for compressing strings, otherwise a stochastic grammar. It does not behave well when there is much noise in the strings, symbols which don't occur very often and which have no significance, and it does not do a particularly good job of extracting the brief and conceivably critical transitions which may well characterise, say, stop consonants.

Alternatives, such as simply finding substrings which recur, up to some matching criterion, can be employed and are much faster. Local grammars and variants of them designed to accommodate noise conveniently may be used with only a small amount of ingenuity, and these will be discussed in the next chapter.

HMMs have been used in handwritten character and word recognition; one might be forgiven for suspecting that the choice is dictated by the crudest forms of research practice which requires absolutely no understanding of what you are doing: you copy out an algorithm used for tackling one problem and use it for tackling one which is sufficiently similar to offer hope.[8] There are more intelligent ways which will be

[8] I suppose that a methodology which allows morons to publish research papers is a fine and egalitarian thing, but it does clog up the journals somewhat. Again, I incline to blame the Mathematicians for a disgracefully inadequate job of teaching Mathematics in Universities particularly to Engineers and Physicists. Of course, the Mathematicians blame the Engineers and Physicists for demanding technical skills without any idea of why the methods work. This leads to Engineering journals keeping the standards high by ensuring that there is a significant mathematical content to each paper. They decide this by counting the number of integral signs, partial derivatives and half the summation signs and divide by the number of pages. This leads to the most banal of ideas, which could easily be explained in English in half the space with twice the clarity, being expressed in algebra. The other part of the problem is an insane ambition which leads folk of limited intellect to imagine that they can do Science and Mathematics. Let them do the so called Social Sciences. That's what they were invented for. Real Science and Engineering and Mathematics are difficult. Ah, I feel lots better now, thank you. You might ask why diatribes like this are found in a simple, academic book. Well, if nobody bitches about incompetence, dogmatism and stupidity, we shall get a lot more of it, and there's already plenty. It's a dirty job, but someone has to do it; it clearly takes

discussed later.

6.1.6 Connected and Continuous Speech

Recognising one spoken word is a start, but it is a long way from being able to dictate to your computer and have it type out what you thought you said, or as any good secretary will, what he or she thinks you think you ought to have said. Systems such as the IBM *Tangora* can recognise a sentence of words from a fifty thousand word vocabulary providing each is spoken with a short space between them. [9] This is because there are a lot of ways of trying to decode continuous speech. The reader should try saying each of the four sentences following out loud a few times:

> Ken ewe won ah star net.
> Car knew one a stair nit.
> Care new oner sten at.
> Can you understand it?

The last is the most likely fit to an actual utterance in English, but the earlier versions are likely to be better from an acoustic point of view. In order to have good grounds for preferring the last, one needs to know something of the allowable syntax of the spoken language. The constraints, given by a stochastic grammar of some sort, a kind of statistical model for the strings of the language, are needed in order to control the horrendous number of possibilities. And even working out what the grammar for spoken language *is* can be rather trying[10].

---

a highly developed contempt for the whole concept of political correctness, making the author rather well qualified. Actually, I'm joking. Academic life is really just one big happy family.

[9]What this does to the French language shouldn't be done to bushflies.

[10]There are two activities traditionally practised by grammarians. The first is to regulate how other people ought to talk and write. Grammars promulgated for this reason are called *normative*. The second is to try to describe what is actually said or written by some group of people. Such grammars are called *descriptive*. We are only concerned with the second sort.

For instance, the most common word in the spoken English language is 'uh' or 'ah' or 'um' or 'er'. This is a transactional word which means, roughly, 'shut up, I haven't finished talking yet'. It does not occur, for obvious reasons, in the written language at all. So when a system has had the constraints of the written language put in and then somebody says to the system 'Tell me about, uh, China', the system may well start telling him about America, since there is no country called Uhchina and the system 'knows' it has to be a proper name in its data base beginning and ending with a schwah. There is an apocryphal tale about such a system which asked the user which day he wanted to come back and he sneezed. After thirty seconds of agonising the system replied: 'We're closed Saturdays'. So the business of using constraints to determine which word string was used can run into troubles.

The basic problem of sentence recognition is that people don't speak in words. There is no acoustic cue corresponding to the white space in text. This leads the reflective person to wonder if we could cope were all the white spaces and punctuation to be removed from English text, something which invites a little experiment.

In general, the attempt to use the constraints of natural language to improve recognition systems have succeeded only because the systems that didn't use them were unbelievably bad. There are many different kinds of information which human beings are known to use, from syntactic at the phonemic level(knowing what strings of noises can occur) to syntactic at the lexical level (knowing what strings of words can occur) to semantic (knowing what the words and sentences mean), pragmatic (knowing what the speaker is talking about) and prosodic (stress: pitch and loudness variation). Workers are still exploring the issues involved in using these sources of information intelligently. They have been since the seventies; see the many reports on the HARPY and HEARSAY projects from Carnegie-Mellon University. A discussion on the subject of inferring (stochastic) grammars for a language from a sample will be a feature of the next chapter. You have already seen it done once, rather unconvincingly, using the EM algorithm for Hidden Markov Models.

It should be plain to the thoughtful person that we are a long way from telling how to build the ultimate speech recognition system, and that this discussion merely outlines some of the difficulties. The situation is not actually hopeless, but the problem is difficult, and it would be as well not to underestimate its complexity. The AI fraternity did a magnificent job of underestimating the complexity of just about all the problems they have looked at since the sixties: see the reference to Herbert Simon in the bibliography to chapter one. Engineering builds wonderful devices once the basic science has been done. Without it, engineers are apt to want to plunge swords into the bodies of slaves at the last stage of manufacture so as to improve the quality of the blade: sea water or suet pudding would work at least as well and be much cheaper. The business of discovering this and similar things is called 'metallurgy' and was poorly funded at the time in question. Similarly, the basic science of what goes on when we hear some spoken language at the most primitive levels is in its infancy. Well, not even that. It's positively foetal and barely post-conception. It's an exciting area, as are most of the areas involved in serious understanding of human information processing, and progress is being made. But for the man or woman who wants to make a robot behave intelligently, abandon now the thought of having a discussion with it and hope, in the immediate future, for it to be able to handle only single words, or sentences with artificial and restricted syntax.

6.2 Filters

There is a fairly well developed theory of trajectories in $\mathbb{R}^n$, which encompasses both the case of continuous and discrete time, and which is aimed not at classifying the trajectories but at predicting, interpolating and smoothing them. This entails some kind of modelling process for the production of the function. It is reasonable to care about this sort of thing even if we are interested in classifying trajectories and not predicting, interpolating or smoothing them, for two excellent reasons. The first is that we may reasonably expect to be plagued with noise in whatever dynamic pattern we are faced with, and the theory of filters gives some advice on what to do about this. The second is that any recognition or classification will surely involve us in some kind of modelling of the process which generated the trajectories, and so it is worth looking to see what kinds of models other workers concerned with dynamic patterns have thought worth investigating. We may be able to pinch their ideas.

In the case $n = 1$ we are talking about predicting, interpolating or smoothing the graph of a function of a single real variable, as they used to call it (and still do in some places). The function is usually referred to as a *signal* or a *time series*, depending on whether your source of information is an engineer or a statistician. The business of finding a function which looks to be a good fit to the given graph is called *filtering* if you are an engineer, *regression* if you are a statistician, and *function-fitting* or *function approximation* if you are just one of the boys.

Conceptually, you are confronted with a sequence of dots, or possibly a wiggly line, on a piece of paper which invites you to take a thick, black brush and to draw a smooth curve through the dots or the wiggly line, not necessarily through each dot, maybe passing in between dots or wiggles. Then you can extend your smooth curve with a flourish (extrapolation or prediction), fill in between points (interpolation) or decide you would prefer your curve to the actual data at some particular x value (smoothing). To do this by eye is to invite confrontation with anybody else who did the same thing, since the two smooth curves will surely differ, at least a bit. This leaves us with the problem of automating the process of getting the smooth curve, of getting the *right* smooth curve, or at least one which can be agreed upon as acceptable and also computed automatically. After chapter three, it probably won't come as much of a shock to discover that there is no one 'right' solution, and doing it by hand and eye and some dash may be performed with a clear conscience. This method is no less rational and defensible than any other; the rationales for the other methods are more complicated but ultimately hinge on you buying some untestable assumption about the data. The advantage of automating the process is not that you can feel any more confidence in its rightness, but that it can be then done to a lot more data than your arm could cope with.

There are two traditions in the area, one is the statistical tradition (time series) going back to Yule and Kolmogorov which talks of AutoRegressive Moving Average (ARMA) models for time series, and the engineering tradition which talks of Finite Impulse Response and Infinite Impulse Response (FIR and IIR) filters for signals. These are the same thing. I have heard tell of a statistician who asked an engineer if he wanted ARMA modelling done in his course on stochastic time series for engineers, and was told 'no, we don't need that, just see if you can fit in a bit on FIR and IIR filters, if you have time.' Not knowing what these were, the statistician simply shortened the course.

The engineering terminology confuses the objectives with the methods and details of the source of the problem, that is to say it is cluttered with massive irrelevancies and makes it hard to see the wood for the trees. The statistical terminology by contrast has been made so abstract by some (the French school) that you need a strong background in analysis before you can understand what the subject area is, by which time you may have lost interest.

The basic ideas then, have been obscured by the language in that traditional manner so widely believed to be the main contribution of Mathematics to Science and Engineering. It will sadden the reader to learn that a whole machinery of analysis of speech, Linear Predictive Coding, depends upon it. The reader is advised that if she lacks the training in either of the areas mentioned, she will find her patience sorely tried by the literature. I shall try in this section to explicate the main ideas while avoiding details. As you were warned in the introduction to this chapter, the area is a large one, and the most useful contribution I can make in this book is to sketch out the central ideas in my inimitable way, so as to allow you to decide if you want or need to go further, and to suggest where to look if you do.

6.2.1 Linear Systems

Definition 6.1 *A* discrete real time series *is a map from $\mathbb{Z}^+$ or $\mathbb{Z}$ into $\mathbb{R}$; we think of the integers as consecutive ticks of a clock, and the real values u_n or $u(n)$ as being the value of some physical variable at the n^{th} clock tick.*

For example, the clock ticks might be once a day and the values might be the Dow-Jones averages on those days, or the clock ticks might be once every twentieth of a millisecond, and the values the voltages output by a microphone as you talk into it. Some notations have $u(n)$ for the value and hence u as the time series, which is evidently a function; others write the sequence $\{u_1, u_2, \cdots, u_n, \cdots\}$ which is clumsier but more familiar to the older generation. I shall use the function language because it is neater and cuts down on the number of different kinds of things you have to think about.

Definition 6.2 *A* continuous real time series *is a continuous map from $\mathbb{R}$ or $\mathbb{R}^+$ into $\mathbb{R}$. We think of the input as being continuous time and the output as being the value of some changing measurement at that time.*

Since we are going to be dealing with real life, where it is impractical to take a measurment at every conceivable instant of time, we have no immediate interest in such things except that we may like to believe for philosophical reasons that a discrete time series came from sampling a continuous one. Of course, you have to be pretty committed to metaphysics to believe that using $\mathbb{R}$ for time or space is anything more than a linguistic convenience. Maybe space-time comes in very, very small lumps.

It is common to look at stock exchange figures and to feel that there is altogether too much wild variation in consecutive values. A standard way of *smoothing* the stock exchange time series is to replace the central value on a day by the average of the three days consisting of the original day, the preceding day, and the succeeding day. This cannot be done in real time, you get to be a day late with your figures, but predictions on smoothed data might be more useful, so it could be worth it. This is an example of a *moving average filter*; it takes in one time series and outputs another. In fact this is the definition of a filter.

Definition 6.3 *A* filter *is a map from a space of time series to a space of time series.*

The usual application is to filter out random noise, as in the moving average example. In the old days, when life was hard and *real* engineers had to do three impossible things before breakfast, the time series was likely to be continuous, was almost invariably a voltage, and a filter was made out of capacitors, inductances, resistors and similar knobbly things you could pick up and lose. The signal went in, and emerged purified, like water having the dirt filtered out by pouring it through well, through filters. The mental image of a dirty, insanitary time series coming in, and a somewhat improved one leaving, still pervades the language. In these latter days, the time series is more likely to be a sequence of numbers input to a computer by some data acquisition system or output from it to some mechanism, while the filter is likely to be a software construct. In this case they become known as *digital filters.* I shall consider only digital filters here, with a brief aside now and again to indicate the directions for the continuous case.

6.2.2 Moving Average Filters

The moving average filter as described can be modified in a number of ways: we can take more than three numbers, we can insist that the filter be *causal* which means that the output at time k cannot require knowledge of the input at a later time, and the averages can be weighted. If I give you the numbers one quarter, one half, one quarter, with the instruction to add one quarter of yesterdays Dow Jones to half of today's to one quarter of tomorrows, we output (a day late) a *weighted moving average.* The histogram of three numbers, one quarter, half, one quarter, is a sort of blurring mask which we run over the time series to modify it. we can regard it as a function defined from $\mathbb{Z}$ to $\mathbb{R}$ which is zero except at -1,0 and 1, where it takes the given values. You may be reminded of the blurring done by morphology techniques.

Definition 6.4 *If f and g are functions from $\mathbb{Z}$ to $\mathbb{R}$, the* convolution *of f with g, written $f * g$, is a new function from $\mathbb{Z}$ to $\mathbb{R}$ defined by*

$$f * g(n) = \sum_{i=-\infty}^{\infty} f(n-i)g(i)$$

It is clear that I smooth f by performing a convolution with a suitable g. What I do is to compute the new, smoothed function $f * g$ a a point n by centring g on n (that is move the 0 value of g over n) and then multiply the corresponding values of f and g and sum them. You might think I have written g backwards, but in this form it works if both functions are defined for the non-negative numbers only. A variant with an integral sign for the case of continuous functions defined on $\mathbb{R}$ or $\mathbb{R}^+$ is left as an exercise for the student.

If f is not defined for all values of $\mathbb{Z}$, then I need to do some obvious fiddling, like making $f(n)$ zero when n is negative or changing the limits on my summation. Since filters, to be implementable must have only a finite number of terms, in practice one is rather restricted to smoothing envelopes g which are non-zero for only a finite (and usually pretty small) number of values.

If the values of g are allowed to be negative some rather surprising things can happen, and it is useful to experiment by obtaining some time series, writing down a Moving Average filter g, and performing the convolution in a program to see what comes out. In particular, it is possible to input white noise and get out a time series which manages to look deeply significant. White noise is, for those who don't know, noise which has equal spectral representation at all frequencies, and is hence entirely mythical. Approximations to it over the audio spectrum sound like surf on a beach and are said to be very soothing to listen to. Well, more soothing than most things one hears on the radio these days, at least. One can produce quite reasonable simulations of gaussian white noise by choosing, for each clock tick, quite independently, a random number according to a gaussian or normal distribution with mean zero and variance one. The resulting graph or signal looks a mess. If you haven't got a gaussian die to throw, fiddling your random number generator to produce an approximation to one is an exercise which is almost as soothing as listening to the result played back through a speaker, and gives a somewhat greater sense of accomplishment.

If we take it that g, the function with which we propose to accomplish a smoothing, consists essentially of a small number of non-zero numbers, we can refer to them as the *coefficients* in the Moving Average filter, and then a filter of order k has just that number of coefficients. More accurately, we can take the set of k numbers

$$b_0, b_1, \cdots, b_{k-1}$$

and filter u to v by

$$\forall n \in \mathbb{Z}, v(n) = b_0 u_n + b_1 u_{n-1} + \cdots + b_{k-1} u_{n-k+1}$$

so as to get a causal MA filter determined by the filter-vector of coefficients. Moreover, we could extend the idea by having a Moving Average filter which changed as we move along the time axis. It had better not change too fast, or the process is unlikely to be intelligible in terms of changing filters, but if we take a moving average family of filter vectors which change in time, we can use this to model speech. Instead of using white noise as input, we can use a pulse at the frequency of the pitch of a human voice, and argue that what our filter family is doing is modelling the way the vocal tract acts on the vocal chords to produce a sound. This is conceptually fairly straightforward, and gives an idea of a class of models, even if it lacks a certain amount of necessary detail.

6.2.3 Autoregressive Time Series

A time series f might have the property that the value of f at time n is related to the value at an earlier time. In particular if f were periodic with period T then we would have $f(n) = f(n-T)$ or perhaps $f(n) = -f(n-T/2)$. If f were built up as a finite sum of periodic functions with a variety of periods and phases, we could write

$$f(n+1) = a_0 f(n) + a_1 f(n-1) + \cdots a_{r-1} f(n-r+1)$$

for some suitable collection of r coefficients.

Definition 6.5 *A time series $f : \mathbb{Z} \longrightarrow \mathbb{R}$ is said to be* autoregressive of order r *iff*

$$\exists a_0, a_1, \cdots a_{r-1} \in \mathbb{R} : \forall n \in \mathbb{Z}, f(n+1) = \sum_{i=0}^{r-1} a_i f(n-i)$$

As I have defined it, some series are autoregressive, while most are not. You choose some starting sequence of r values for f, and then operate upon them to produce the next value of f. Now you slide up one value and do the same again. A little thought shows that if the coefficients were of the wrong sort, the series could easily blow up; for example if $r = 1$ and we have $f(n+1) = 2f(n)$, we get a geometric progression going bigger, while $f(n+1) = 1/2f(n)$ has it sinking to zero fast. The situation with more coefficients is more complicated and is considered by engineers under the title of stability of the filter.

To the engineer, a set of coefficients like the b_i above is just asking to be made into a filter. Is such a thing a filter? If I give you a starting segment you can go ahead and generate a time series, but what do I do if I have a whole time series as input? Well, nothing subtle. I can MA filter it and add it in, but that doesn't make for a separate kind of filter called Autoregressive (AR) filtering, although you might naively think it did.

What I can do is to consider a class of filters, called IIR filters , where if you input a time series, say u, you get output time series v and where

$$\begin{aligned} v(n+1) &= a_0 v(n) + a_1 v(n-1) + \cdots a_{k-1} v(n-k+1) \\ &+ b_0 u(n) + b_1 u(n-1) + \cdots + b_{r-1} u(n-r+1). \end{aligned}$$

The IIR is short for *Infinite Impulse Response* and comes from the observation that if you choose r to be 1, input a time series u which is zero everywhere except at one time where it takes the value 1, make b_0 any non-zero value you wish, then unless the a_i are rather unlikely, the output time series will never become permanently zero (Although it can get pretty close to it). If you put such an input into a MA filter, then of course it rapidly goes to the average of the zero inputs, so is zero except for some finite period of time. Hence the term *FIR filter*, F for Finite, for what I have called a Moving Average filter, because my name tells you a lot more about what it is.

Much of the history of Filtering theory dominates the older treatments in the Engineering texts. Given a sequence of real or complex numbers, some engineers feel an irresistible urge to make them the coefficients of a complex power series or Laurent series, and use the resulting complex function instead of the series. It is possible to satisfy oneself that there is a one to one correspondence between convergent infinite sequences and complex analytic functions. This leads to turning problems about time series into problems about properties of complex analytic or more generally meromorphic functions. (Meromorphic functions may go off to infinity at some discrete set of points while analytic ones don't.) Having suffered the agonies of mastering complex function theory, they don't see why you shouldn't suffer too; they may firmly believe that such suffering enobles and strengthens the character, although others may doubt this. I should undoubtedly be drummed out of the Mathematician's Union were I to cast doubts on this view. Besides I like Complex Function theory. So I won't.

6.2.4 Linear Predictive Coding or ARMA modelling

For reasons which are not altogether convincing, it is not uncommon to model speech as if it were produced by IIR filtering a glottal pulse input (for voiced sounds) or white noise (for unvoiced sounds). Then if we know what is supposed to have gone in, we know what came out, we can calculate the coefficients which give the best fit to the output over some period of time. As the vocal tract changes, these coefficients are also supposed to change in time, but relatively slowly. So we change a fast varying quasi-periodic time series into a vector valued time series, or a bit of one, which I have called the trajectory of an utterance. The argument for Autoregressive modelling suggested above hints at relationship with the Fourier Transform, which emerges with more clarity after some algebra. This approach is called *Linear Predictive Coding* in the Speech Recognition literature.

ARMA modelling with its assumptions, implausible or not, is done extensively.[11] A variant is to take a time series and 'difference' it by deriving the time series of consecutive differences, $v(n) = u(n) - u(n-1)$. This may be repeated several times. Having modeled the differenced time series, one can get back a model for the original time series, given some data on initial conditions. This is known as ARIMA modeling, with the I short for Integration.

The modelling of a stationary time series by supposing it arrives by filtering a white noise input is a staple of filtering theory. The method is perhaps surprising to the innocent, who are inclined to want to know why this rather unlikely class of models is taken seriously. Would you expect, say, the stock exchange price of pork belly futures to be a linear combination of its past values added to some white noise which has been autocorrelated? The model proposes that there is a random driv-

[11] It is important to understand that we don't judge a model by its plausibility in science, we judge it by whether it works or not. Thus we replace woolly arguments between proponents and detractors by careful investigation of the world. This saves a lot of time, and sometimes blood.

ing process which has short term autocorrelations of a linear sort and arises from the driving process by more autocorrelations, that is dependencies on its own past. Would you believe it for pork bellies? For pork belly futures? Electroencepghalograms? As a model of what is happening to determine prices or anything much else, it seems to fall short of Newtonian dynamics, but do you have a better idea? Much modelling of a statistical sort is done the way it is simply because nobody has a better idea. This approach, because it entails linear combinations of things, can be written out concisely in matrix formulation, and matrix operations can be computed and understood, more or less, by engineers. So something can be done, if not always the right thing. Which beats scratching your head until you get splinters. Once the reader understands that this is desperation city, and that things are done this way because they can be rather than because there is a solid rationale, he or she may feel much more cheerful about things.[12] For speech, there is a theory which regards the vocal tract as a sequence of resonators made up out of something deformable, and which can, in consequence, present some sort of justification for Linear Predictive Coding. In general, the innocent beginner finds an extraordinary emphasis on linear models throughout physics, engineering and statistics, and may innocently believe that this is because life is generally linear. It is actually because we know how to do the sums in these cases. Sometimes, it more or less works.

6.2.5 Into $\mathbb{R}^n$

Given a time series $u : \mathbb{Z} \longrightarrow \mathbb{R}$, it is sometimes convenient and feasible to turn the time series into a map from $\mathbb{R}^n$ to $\mathbb{R}^n$ which is iterated, with some noise added. All we do is to take a sequence of n consecutive time values of the series as input, and the shift to one time unit later

[12] Or perhaps not.

as output. That is, we look at the map

$$\begin{pmatrix} u(t) \\ u(t-1) \\ \vdots \\ u(t-n+1) \end{pmatrix} \longmapsto \begin{pmatrix} u(t+1) \\ u(t) \\ \vdots \\ u(t-n+2) \end{pmatrix}$$

This I shall call the *delay map*.

Similarly if we have a MA filter of order k from a time series u to a time series v we can write this as a linear map from $\mathbb{R}^{k+1}$ to $\mathbb{R}^k$. For example, if we have that

$$v(n) = b_0 u(n) + b_1 u(n-1)$$

we can write

$$\begin{pmatrix} v(n) \\ v(n-1) \end{pmatrix} = \begin{pmatrix} b_o & b_1 & 0 \\ 0 & b_0 & b_1 \end{pmatrix} \begin{pmatrix} u(n) \\ u(n-1) \\ u(n-2) \end{pmatrix}$$

Writing such a map as f, it is not hard to see that ARMA modeling of v becomes a modeling of f by matrix operations. This approach has the additional merit that if we have a time series of vectors, we can simply amalgamate the vectors in the same way, and the general theory doesn't trouble to distinguish between the cases where a time series is of vectors having dimension one and where it has any other dimension.

It is not particularly reasonable to expect the delay map to be linear or affine, with or without noise, but we can keep our fingers crossed and hope that it is. Chaotic systems arise rather easily when it isn't. Our position is not so much that linear and affine systems are the ones which arise in practice a great deal, it's that those are the ones we can figure out what to do something about.[13]

[13]Definitely desperation city. When Einstein remarked that the most amazing thing about the universe was that we could understand it, he was looking at a biased sample. Mostly we can't. It's rather less amazing that there are bits of the universe simple enough to be understood even by thickos like us, and quite typical of human beings to congratulate themselves excessively when they find one.

If we take a point in the domain of the delay map and apply the map to this point, and then to the result of applying the map, and thus iterate the map on this point, we get a set of points in $\mathbb{R}^n$. We may compute the mean and covariance matrix for this set in order to get a second order description of it, just as for a set of pixels in $\mathbb{R}^2$. It is called the *autocovariance matrix*. It may turn out that the set of points lies on a subspace of dimension significantly less than n. Anything that can be said about the set of points obtained by parcelling up the time series in this way is a statement about the process which generated the time series, so we have a distinct interest in describing point sets in $\mathbb{R}^n$. The mean and covariance matrix are rather inadequate descriptors for complicated shapes, and so it is possible to go to higher order. It is also possible to go to local descriptions which are locally low order and to piece them together. We shall have more to say about this later.

The simple minded trick of taking a time sequence of numbers and turning it into a time series of vectors by putting a moving window over the time series in the hope that scrutiny of the resulting point sets will be informative is also done in the old piecewise affine neural nets, where a simple perceptron then becomes a MA filter with a non-linear threshold stuck on the end. These are called *time-delay* neural nets. We can also feed the output back into the input through a subnet which procedure yields a *recurrent neural net*. The range of basic ideas here is really fairly limited.[14]

6.2.6 States

The assumption that we are using at present is that it suffices to study time series with linear dependencies so that if we take vector values or time blocks or both, we have an output vector valued time series v produced by an input vector valued times series u (which might be uncorrelated gaussian white noise or might not) and with an autore-

[14]This is the *good* news.

gressive component. Formally,

$$v(n+1) = Av(n) + Bu(n)$$

for vectors $u(n), v(n), n \in \mathbb{Z}$, and matrices A, B. It is usual to introduce another complication by supposing that there is an internal state which is produced by such a process and that this is observed by some means which may introduce more noise. Thus we may have two equations:

$$\begin{aligned} z(n+1) &= Az(n) + Bu(n) \\ v(n+1) &= Cz(n+1) + Dw(n+1) \end{aligned}$$

Here, z is some internal state which is not observed, it is 'hidden', and which gives rise to v which is observed. The output vectors lie in some affine subspace of the output space in general, and finding it and its dimension is a part of the problem of filtering the process given by the above equations. In even more generality, we may have the case where the matrices A,B,C,D are allowed to change in time, although not, one hopes, too quickly.

6.2.7 Wiener Filters

To take the simplest example of filtering, suppose $\{u = u(n) : n \in \mathbb{Z}\}$ is an input series and $\{v = v(n) : n \in \mathbb{Z}\}$ is an output series. I want to have a convolution of u with some finite filter so that u looks like v. Let us suppose that I have some finite amount of history of the past of u, say for $j \in [1 \ldots n]$. Then I want to choose some coefficients $b_0, b_1, \cdots, b_k$ so that

$$\sum_{j \in [1 \ldots n]} (v_j - b_0 u_j - b_1 u_{j-1} - \cdots b_k u_{j-k})^2$$

is a minimum.

If I take k to be 1, for purposes of exposition, then I am trying to minimise

$$\sum_{j\in[1\ldots n]} (v_j - b_0 u_j - b_1 u_{j-1})^2$$

The expression is best seen as a distance in $\mathbb{R}^n$ from the point $\begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix}$ to the plane $b_0 \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} + b_1 \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{n-1} \end{pmatrix}$ determined by all possible choices of b_0 and b_1. In general, it will be a k-dimensional subspace determined by all possible choices of the b_i.

The problem of finding the closest point on a plane, P to a given point v in $\mathbb{R}^n$ but not on the plane, is a simple piece of linear algebra. We observe that if the closest point to v on P is called q, then the line joining the point v to the point q is orthogonal to the plane. Hence it is orthogonal to the vectors which span the plane. This follows immediately from Pythagoras' theorem, although it is called rather pompously the 'orthogonality principle' in many text books. If you draw a picture for $n = 3$ you should have no particular difficulty with this: see $Fig.$6.3.

For orthogonal vectors the dot product is zero, so we obtain the two equations

$$\left(b_0 \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} + b_1 \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{n-1} \end{pmatrix} - \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} \right) \bullet \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} = 0$$

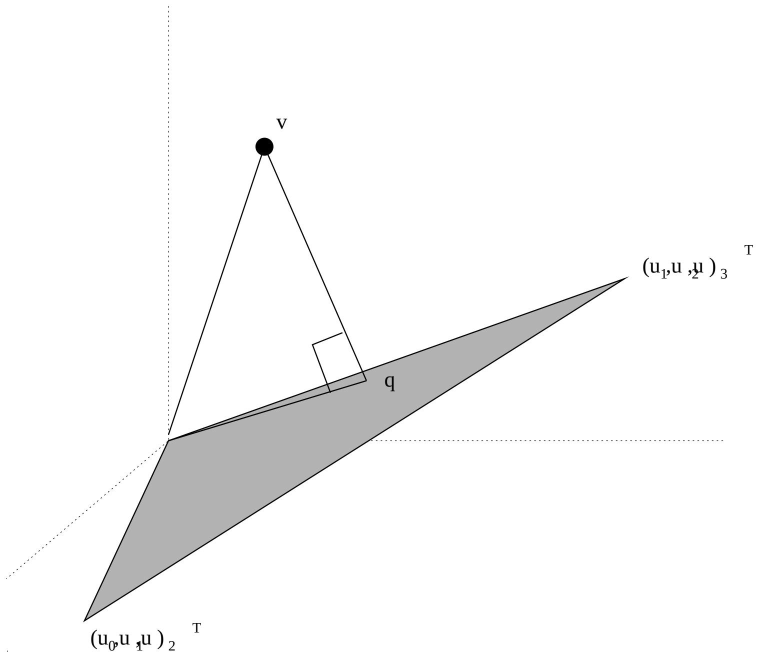

Figure 6.3: Closest point to a plane through the origin: line **v q** orthogonal to plane.

and

$$\left(b_0 \begin{pmatrix} u_1 \\ u_2 \\ \vdots \\ u_n \end{pmatrix} + b_1 \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{n-1} \end{pmatrix} - \begin{pmatrix} v_1 \\ v_2 \\ \vdots \\ v_n \end{pmatrix} \right) \bullet \begin{pmatrix} u_0 \\ u_1 \\ \vdots \\ u_{n-1} \end{pmatrix} = 0$$

If instead of having a filter of order 2 we had used k coefficients, we should have had the same orthogonality condition (still by Pythagoras' Theorem!) but this time there would have been k of them. $\bullet$ denotes the dot product in $\mathbb{R}^n$.

The k equations are called the *normal equations* and solving them gives the point q. They are simply k linear equations in k unknowns. We leave it to the reader to decide when there is a unique solution.

In the filtering case, the equations are also known as the *Yule-Walker* equations, and the corresponding equations in the continuous case are called the *Wiener-Hopf* equations. [15] The solution gives a so-called *optimal filter* but it should be noted that we are simply trying to get as close as we can to a known point in $\mathbb{R}^n$ while staying in a linear subspace.

An obvious practical objection to the process as described above, is that the filter pays as much attention to the current values as it does to the remote past ones. In other words we have chosen a particular (Euclidean) metric on the space $\mathbb{R}^n$ given by

$$\left(d\left(\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}\right)\right)^2 = (x_n - y_n)^2 + (x_{n-1} - y_{n-1})^2 + \cdots + (x_1 - y_1)^2$$

where it might be more realistic to discount the past by weighting it less. This can be achieved by changing the metric to:

$$\left(d\left(\begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix}\right)\right)^2 = (x_n - y_n)^2 + \frac{1}{2}(x_{n-1} - y_{n-1})^2 + \cdots + \frac{1}{2^{n-1}}(x_1 - y_1)^2$$

This can be done by defining a new inner product. The choice of $\frac{1}{2}$ raised to increasing powers can be varied to taste. A new inner product can be conveniently represented by a symmetric, positive definite matrix. If $\mathbf{x}$ and $\mathbf{y}$ are vectors and A is such a matrix, the new inner product will be $\mathbf{x}^T A\mathbf{y}$ and the good old 'dot' product is simply the special case where A is the identity matrix. The above equations may look slightly different in this form.

It is common to rewrite the $(k+1)$ normal equations as follows:

$$\forall j \in [0 \cdots k], (b_0\mathbf{u}_n + b_1\mathbf{u}_{n-1} + \cdots b_k\mathbf{u}_{n-k}) \bullet \mathbf{u}_{n-j} = \mathbf{v} \bullet \mathbf{u}_{n-j}$$

[15] The power of Linear Algebra is shown here: No less than four blokes got their names attached to a geometrically trivial observation just because the dimension was higher than three.

or alternatively as:

$$\forall j \in [0 \cdots k], \sum_{i=0}^{k} b_i(\mathbf{u}_{n-i} \bullet \mathbf{u}_{n-j}) = \mathbf{v} \bullet \mathbf{u}_{n-j}$$

If you reflect on what n ought to be in practice, you can see that it should be as long a sample as you have from starting time to finishing time, except for the small problem of shifting outside the region for which we have data. A certain amount of wastage will occur at the ends of a finite sample. Note that we do not in fact need the time series $\mathbf{v}$ we need only its dot product with different shifts of the time series $\mathbf{u}$.

A slightly different formulation can be obtained by dividing the above equations on both sides by n; in this case we can define the *Auto-Correlation Function (ACF)* for a time series u as the function which assigns to an integer j the average or expected value of $u(n)u(n-j)$. We can define the *Cross-Correlation Function (CCF)* for u and v to be the average or expected value of $v(n)u(n-j)$. If the time series u is *stationary* then

$$\frac{1}{n}(\mathbf{u}_{n-i} \bullet \mathbf{u}_{n-j}) = r_{11}(i-j)$$

where $r_{xx}(t)$ is the ACF of u at t. Similarly

$$\frac{1}{n}\mathbf{v} \bullet \mathbf{u}_{n-j} = r_{12}(j)$$

where $r_{12}(t)$ is the CCF of v and u. Whereupon the Normal-Yule-Walker-Wiener-Hopf equations become:

$$\forall j \in [0 \cdots k], \ \sum_{i=0}^{k} b_i r_{11}(i,j) = r_{12}(j)$$

with the obvious changes made for the case when everything is continuous.

The expected value of the CCF of two functions may be zero; if v is a signal and u is random noise, then this should be the case, and if u is uncorrelated white gaussian noise with zero mean, then the ACF of u (the CCF of u with itself) ought to be zero except at 0 where it should be the variance. This is the sort of thing which is dead obvious, but when said in algebra can leave you wondering whether it is profound or not.

Not.

If we write the time series as a time series of vectors obtained by storing consecutive windows of n of them and moving the windows along to turn the time series of real numbers into a time series of vectors in $\mathbb{R}^n$, as described earlier, then we can compute the autocorrelation matrix of a piece of the time series regarded as a set of vectors. The autocorrelation matrix is the covariance matrix with a shift, because the mean is not subtracted from the values when you compute the autocorrelation matrix. It is easy to verify that the matrix equations relating the covariance matrix with the autocorrelation matrix are

$$Cov(Data) = AutCor(Data) - \frac{N}{N-1} M M^T$$

where $Data$ is the matrix formed by listing N vectors in $\mathbb{R}^n$ to give an $N \times n$ matrix, M is the $n \times 1$ matrix of the mean of the data vectors, M^T is its transpose, and $Cov(Data)$ and $AutCor(Data)$ are self explanatory.

6.2.8 Adaptive Filters, Kalman Filters

This brief discussion of adaptive filtering is intended to place the general ideas of what they do and how they do it in some sort of context relevant to pattern recognition. The details of how to do the sums are easily obtained from the sources given in the bibliography, should that look to be a good idea.

The last subsection should have made it clear that computing the values of an optimal MA filter was essentially the same idea as fitting a straight line through a set of points. After all, if I give you data consisting of

$\{(x-1, y_1), (x_2, y_2), (x_3, y_3)\}$ and ask for the line $y = mx + c$ that fits the data best, then you follow Gauss (by about two hundred years; my how time flies when you're having fun) and simply take the sum of squares:

$$(y_1 - (mx_1 + c))^2 + y_2 - (mx_2 + c))^2 + (y_3 - (mx_3 + c))^2$$

and choose m and c so as to make this a minimum. This sum of squares we regard as the distance from the point $\begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$ to the plane $m \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} + c \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$. This is a minimum at a point $\mathbf{u}$ on the plane when the line joining $\mathbf{y} = \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix}$ to the point $\mathbf{u}$ is orthogonal to the plane and hence to its basis vectors. This is precisely what we did with a filter having two coefficients. Gauss was confronted with the problem of continuously updating his estimate; he had the batch mode least squares worked out by 1795, and devised an online least squares system quarter of a century later. He also did rather a lot of other things in the same period.

If I give you a stream of (x_i, y_i) and leave you to find the best fit straight line to the data, you can wait until the stream stops and then start working or you can take a block of some number, do it for the block, and then work out how to pool your results with those for the next block of data. A block of data can of course consist of just one datum. If you have reason to suspect the best fit line may be changing in time, you may want to discount old data somewhat. When the model does not change its parameters in time we say it is *stationary*, a definition which raises as many questions as it answers, and a recursive least squares best fit algorithm which can discount the remote past can track a non-stationary process.

The various methods of doing adaptive filtering, are, in effect doing just this. The details can be found in some of the books given in the references.

In Kalman filtering the process is taken to a high level. In the simplest case, the equation $y = mx + c$ is generalised so that y,x and c are vectors and m is a matrix, and c is often taken to be gaussian white noise that has been filtered by a MA filter to autocorrelate the noise. It remains true however that except for the complication of an inner product other than the old dot product to describe the varying degree of credibility of past data and the autocorrelations of the noise with time, what we have is basically a least squares best fit problem done sequentially rather than in batch mode. The choice of a modified least squares can either be justified by placing restrictions on the model class (which are never in practice confirmed to be correct), or not justified at all and regarded as a simple and workable heuristic (as Gauss did) for getting an answer of some sort which is justified *post hoc* and traded in for a better method when one turns up. Statisticians seem to be tempted by the first approach; they feel the comforts of philosophy keenly. Engineers, as is well known, incline to be brutal, insensitive oafs[16] who can't distinguish clearly between rationales and rationalisations and don't trust either.

Kalman filtering is of considerable interest partly because instead of the vector c, above, being a noise vector to be got rid of, it may also be interpreted as a control function to be used to force the system from one state to another, so there is a theory of control which is dual to the theory of filters. This halves the amount of work you have to put in to understand a given amount of material, which is a *good thing*.

6.3 Fundamentals of dynamic patterns

Let us suppose that, as with Speech, we have decided what n things to measure, and have made a sequence of measurements for each of

[16]Well, the ones who can put up with me tend to be. No doubt it's a biased sample. Of course, *they* say they are merely healthy and have good digestions and clear consciences as a result of doing honest work.

a number of events, such as will certainly occur if we are attempting to recognise words. We may suppose that these have been labelled appropriately, and next wish to take a new, unlabelled such trajectory in $\mathbb{R}^n$ and classify it. What basic issues occur? What considerations need to be investigated, whether the problem is in speech or in any other form of dynamic pattern recognition?

The first consideration is that of invariance: if there are known to be groups or parts of groups of transformations under which the trajectory classification is invariant, this information needs to be taken into account. If, for example, the trajectory classification is invariant under a monotone reparametrisation of the path, then we can reduce each trajectory to a canonical time and rate by reparametrising by path length. If there are operations on the space $\mathbb{R}^n$ under which the classification remains invariant, then the transformations should be factored out. If, for example, adding any value to one co-ordinate does not change the classification, then simply project down onto the other co-ordinates and ignore the one that doesn't make a difference. It can only complicate the recognition. Of course, there ought to be some good grounds for believing that the invariance is as claimed, if it is not, you could be making big trouble for yourself.

The main reason for wanting to factor out invariance of as many sorts as possible is, as has been mentioned above, to get more data into the recognition system. To make this clear, if you have a hundred printed digits in each category from 0 to 9 *but they can be anywhere on the page* and if you didn't register them in some way, you would be obliged to learn the (shift) invariance, and this would be wasting data. Similarly, if you had them in different sizes but otherwise identical, it would be sensible to normalise them. This way, you increase the effective size of your training set by reducing the size of your parameter space.

In practice we may be obliged to learn the invariance because we don't know what it is in advance. This may be done more or less efficiently. Starting out with the intention of finding out what the invariants are as a conscious aim is hard to beat as a prelude to an efficient way of finding them; relying on chance or the kindness of the gods is not

Figure 6.4: An aeroplane silhouette.

recommended.

To focus ideas, suppose we have measured some properties of an aeroplane silhouette which describe the shape in some way. For the present we could suppose that it was done by computing moments of the set of pixels shown in $Fig.$6.4.

Let us suppose we have the plane coded as a point in $\mathbb{R}^n$ for some reasonably large $n \gtrsim 20$. I shall assume that no invariance has been built into the measurement process. Now suppose we rotate the figure about its centroid by some small angle and measure the resulting object as a new point in $\mathbb{R}^n$. If the change is sufficiently small then we may expect that the resulting point will be close to the original point. Continue in this way with a sequence of very small rotations, and the measurement process will embed the Lie group of rotations, $SO(2,\mathbb{R}) \cong S^1$ in $\mathbb{R}^n$. If the aeroplane had some symmetry under rotation, for example if it were a square cross, then the map of the circle group S^1 into $\mathbb{R}^n$ would not be an embedding, the result of rotating by $\frac{\pi}{2}$ and integer multiples thereof would take the object to the same point in $\mathbb{R}^n$. For objects not having rotational symmetry, the map embeds the one dimensional manifold of rotations, otherwise, as has been said already, known as the Lie group SO(2) and looking uncommonly like the circle $S^1 = \{z \in \mathbb{C} : ||z|| = 1\}$, into $\mathbb{R}^n$. Similarly, if we scaled the object by different amounts between, say, 60% and 140%, we would be embedding a copy of the interval $[0.6, 1.4]$ in $\mathbb{R}^n$. And if we generated some shifts it would embed a neighbourhood of the origin in $\mathbb{R}^2$ in $\mathbb{R}^n$, and if we did the lot we should have a four dimensional manifold embedded in $\mathbb{R}^n$ by the measurement process. The embedding would not generally be flat, so the dimension of the embedded set might be four, but it could normally be expected to go outside a four dimensional affine subspace; after all, the dimension of the circle is one, but it would be quite normal to find an elastic band taking up a two dimensional piece of space, usually a bit of the carpet.

Now if we have a lot of different aeroplanes rotated, shifted and scaled, we might want to work out which aeroplane we were looking at, or we might want to say something about the shape of the manifold of transformations. How could we detect the circle embedded in the enclosing space $\mathbb{R}^n$ by the rotations? Remember, all we have is a finite sample of points. Well, the dimension is one, which means that locally we would find that the set of points would lie along a line. If we were to take the set of points obtained by rotating the aeroplane, choose one of them, take a small ball in the space and compute the covariance matrix for the set of points in the ball, then provided there are enough points in the ball and the embedding is reasonably smooth, we would find that all the eigenvalues except for the largest were very close to zero. *And this would be true for any initial such starting point.* The number of non-negligible eigenvalues would tell us something about the dimension of the transformations in general.

This procedure is generally used for dimension reduction, not usually locally, and is known, *inter alia*, as *Principal Component analysis.* It is essential to do it locally if we want an honest estimate of the dimension of a non-flatly embedded manifold. If the dimension is not the same in different parts, it isn't a manifold. Many transformations may be expected to arise from manifolds, (Lie groups, or neighbourhoods thereof) although not all.

If vowel sounds are excised and embedded in a space of simulated filterbank coefficients, it is found that the set of trajectory fragments corresponding to a single vowel is pretty much a gaussian cluster in the space, the dimension of which is less than the enclosing space. The centres of the vowels lie in a lower dimensional region- almost a plane in fact, and the locations of the centres projected onto this plane look just like the diagrams of the vowel space which phoneticians draw. This suggests that the space of vowels is two dimensional, that it is nearly but not quite embedded flatly in the filterbank space, and that the variation between vocal tracts and patterns of enunciation occurs in a low dimensional complement of this space. Confirming this would be of some potential value in Speech Recognition, because one could normalise trajectories by shifting them and projecting them onto the appropriate manifold; the projection would not in general be a simple matter of finding the closest distance however, which is all that has been tried so far.

This method allows us to estimate the dimension of the transformations under which the classification is invariant, provided we have enough data. Indeed, if a new vocal tract comes along which is a transformed version of one we have heard before, and if we hear enough identifiable phonemic elements, we might learn the transformation and then apply a learned transformation. For example, by learning part of a scaling transformation on aeroplane images, it has been found possible to extrapolate from a learnt range of 60% to 140% of the size of an image, down to about 20%, at which level resolution becomes an issue. The sudden transition from aeroplane images with different amounts of scaling to squeaky voices with different amounts of squeak will not alarm the reflective reader. [17] We are talking about the same thing abstractly. I shall discuss this at greater length when we come to syntactic pattern recognition of which dynamic pattern recognition is a special case. In the case where the embedding of the transformation group is

[17] And if it burns out the brain of the unreflective reader, serve him right.

flat, we may not only compute its dimension, we can factor it out of the measurement space altogether by taking a linear transformation of the variables we use to measure the state of the system, and then neglect the last r, where r is the dimension of the transformation group. If the transformation group is embedded in a non-flat way, which is regrettably the normal situation, we may seek for a non-linear transformation of the space which will accomplish the same thing. This requires that we understand a little geometry. There are some exercises at the end of the chapter which will enable you to see how to proceed in special cases.

The assumption that the transformations will derive from the action of a Lie group, or some neighbourhood of the identity in a Lie group, is overly optimistic. It is reasonable to suppose that this is largely the case however, and that many transformations can be approximated adequately in this way.

As well as factoring out the transformations under which the trajectory classification is invariant, or at least giving it a shot, there is another desideratum: stability under noise. The measurements of the state of the system at different times will generally be, in this imperfect world, contaminated with noise. This will have an effect on the job of estimating the dimension and structure of any manifold of transformations. It would be as well therefore to smooth the data first. Unfortunately, the smoothing process might easily eliminate the significant parts of the signal or trajectory. For example, in handwriting, cusps and regions of high curvature are usually highly informative, often being separators between different regimes. Having a curve blunted just when it is starting to point somewhere interesting would not be good for recognition. The assumptions behind most smoothing operations, that we have a stationary or at least slowly changing process which is afflicted by gaussian noise, may be grotesquely inappropriate. It may be necessary to divide the trajectory up into segments, each of which is relatively homogeneous, but where there are plainly distinct regimes. If one does this with cursive writing, one gets what might reasonably be called *strokes*, and one can also find such things in the speech signal. I shall discuss this at some length in the chapter on Syntactic Pattern

Recognition.

6.4 Exercises

1. A time series consists of the numbers 1,0,-1,0,1,0,-1,0, repeated many times. Writing it as a set of points in $\mathbb{R}^2$ we get the vectors:
$$\begin{pmatrix} 0 \\ 1 \end{pmatrix}, \begin{pmatrix} -1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ -1 \end{pmatrix}, \begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}, \cdots$$
which we can see is the set of iterations of the matrix
$$\begin{pmatrix} 0 & -1 \\ 1 & 0 \end{pmatrix}$$
on an initial vector $\begin{pmatrix} 0 \\ 1 \end{pmatrix}$. Now suppose some noise is added to this time series, with a zero mean gaussian with variance about 0.1 generating a random number which is added to the original series. This means that iterating from the given (noise free) initial state will produce four clusters in $\mathbb{R}^2$. The fact that there are four tells you that the original time series is periodic with period four, and the fact that each cluster is a spherical gaussian tells you that you have additive gaussian noise which is uncorrelated. Filtering the series can be done by replacing each gaussian cluster with its mean.

   Experiment by modifying the example so as to (a) increase the period, (b) take longer time delay vectors, (c) change the noise so that it is MA filtered gaussian white noise. Investigate the kinds of point set which you get and decide how you might filter the noise out.

   Experiment by taking some matrix map, some initial point, and some noise process. Iterate the map plus the noise and examine the point set you get. If you choose low dimensions you can easily project the sets onto the screen of a computer and gloat over them. What common sense suggestions do you have for filtering out the noise?

2. Use the *fview* program (see the bibliography) to obtain samples of the digits zero to nine spoken by, say, five different people. Now get a program to select a sixth speaker, extract the same words and rename the files so as to conceal the words spoken but to store the encoding in case of fights later. Get all your friends to try to work out from looking at the sample trajectories what words the mystery speaker is speaking. Provide a bottle of wine for the person who gets most right. Get the people who got them right to explain how they did it, and try to write a program to use the same methods.

3. You are given a set of points in the plane in two categories, Noughts and Crosses. The Noughts are represented by the data points:

$$\begin{pmatrix} -1 \\ 1 \end{pmatrix}, \begin{pmatrix} -0.8 \\ 0.64 \end{pmatrix}, \begin{pmatrix} -0.6 \\ 0.36 \end{pmatrix}, \begin{pmatrix} -0.4 \\ 0.16 \end{pmatrix}, \begin{pmatrix} -0.2 \\ 0.04 \end{pmatrix}, \begin{pmatrix} -0.1 \\ 0.01 \end{pmatrix}$$

$$\begin{pmatrix} 0.0 \\ 0.0 \end{pmatrix}, \begin{pmatrix} 0.3 \\ 0.09 \end{pmatrix} \begin{pmatrix} 0.5 \\ 0.25 \end{pmatrix}, \begin{pmatrix} 0.7 \\ 0.49 \end{pmatrix}, \begin{pmatrix} 0.9 \\ 0.81 \end{pmatrix}$$

and the Crosses by the data points

$$\begin{pmatrix} -1 \\ 1.5 \end{pmatrix}, \begin{pmatrix} -0.8 \\ 1.14 \end{pmatrix}, \begin{pmatrix} -0.6 \\ 0.86 \end{pmatrix}, \begin{pmatrix} -0.4 \\ 0.66 \end{pmatrix}, \begin{pmatrix} -0.2 \\ 0.54 \end{pmatrix}, \begin{pmatrix} -0.1 \\ 0.51 \end{pmatrix},$$

$$\begin{pmatrix} 0.0 \\ 0.5 \end{pmatrix}, \begin{pmatrix} 0.3 \\ 0.59 \end{pmatrix}, \begin{pmatrix} 0.5 \\ 0.75 \end{pmatrix}, \begin{pmatrix} 0.7 \\ 0.99 \end{pmatrix}, \begin{pmatrix} 0.9 \\ 1.31 \end{pmatrix}$$

First view the data. This should lead you to suspect that using one gaussian for each category would not be a good idea, and you should also rule out the use of a single unit Perceptron. You then have to choose between the hypotheses that the data shows signs of a group action of transformations which might be learnable, and that it doesn't. In order to test this, you take each data set, a radius of about 0.4, select points and compute for all the points within a radius of this resolution the covariance matrix. You test to see if the results indicate a dimension of one. (Actually, one is the only one worth trying for in the circumstances, so you could give this a miss.) Having found that this is the case, you decide to try to fit a polynomial function to each category.

Compute a low order polynomial to fit each of the categories. This is your first pass at an attempt at obtaining a transform under which the category is invariant. Proceed as follows: linearise the points of one category by taking the polynomial for the Noughts $y = f(x)$ and sending each point $\begin{pmatrix} x \\ y \end{pmatrix}$ to the point $\begin{pmatrix} x \\ y - f(x) \end{pmatrix}$. Now forget about the first co-ordinate and simply store each point by its residue. The results should make pattern recognition rather simpler. Test the legitimacy of using the same polynomial on both data sets by computing the variance of the residues $y - f(x)$ and seeing if the variance is different for each category, or by curve fitting the residue and testing to see how non-linear it is. If the two functions f_0 and f_X for the two categories had been very diffferent, what might you have tried?

4. Given the simplicity of the last exercise, find an embedding of a k-dimensional cube in $\mathbb{R}^n$ for $k \approx 4$ and $n \approx 12$ which gives, when some values are selected randomly and when small amounts of noise are added, a finite data set lying close to a $k-$manifold This is going to be a data set obtained by applying a $k-$dimensional space of transformations to a single point. Now change a few parameters in your embedding to get a different embedding of a different data set. Again generate some more points. Note that I took $k = 1, n = 4; x = t, y = t^2; x = t, y = t^2 + 0.5$ when I did this in the last exercise. I had zero noise.

   Now can you get back again to recover the embeddings given the data points? If you can, describe what happens in the space of embeddings as you go from one category to another. Is there any reason for trying to linearise the space so as to be able to factor out the transformation? After all, it would be simplest to decide what category a datum is in by measuring its distance from the best fitting manifold, why should you go to any more trouble than this? (Hint: There may be more problems in the same space coming down the pipeline!) You might consider the low dimensional case where one category has $y = x2$ as a fitting manifold and the other has $y = x^4 + x^2 + 0.3$ as its. Find a non-linear transform of $\mathbb{R}^2$ which allows you to factor out one component in order to get rid of the transformation group action.

5. **Learning Transformations** You are given a set of points of type Nought which lie on the graph of $y = x^2$ and points of type Cross which lie along the graph of $y = x^4 + x^2 + 0.3$. All such points have X-values uniformly between -1 and $+1$. We shall suppose you have plenty of points so as to be able to fit the curves with high precision. You are also given a point of a third category Square, at location $\begin{pmatrix} 0.5 \\ 0.4 \end{pmatrix}$. Where else might you find points of type Square?

6. The last three exercises had nothing much dynamic about them, and you might feel that they should have been treated earlier with static recognition. Quite so. Now I ask you to produce some pseudo-speech data corresponding to utterances by a range of speakers.

   You take a set of measurements of different speakers saying the vowel /AA/, as in English Cat and Bag, and discover, after binning them into 12 filterbank values that the sounds occupy a region of the space $\mathbb{R}^{12}$ which is approximated by a single gaussian distribution, and has dimension about six, i.e. the first six eigenvalues are reasonably large, and the last six are all pretty much the same, small, and can be interpreted as noise. You repeat for seven other vowel sounds and discover that the same is essentially true for each of them, that the centres of the gaussians occupy a plane in the space, and that the gaussians are more or less shifts of each other in this plane. The principal axes of the gaussians all stand out of the plane at an angle of about 30^o, like six dimensional whales leaping out of a flat sea, all about half out and all pointing in the same direction. An impressive sight.

   You entertain the hypothesis that the distributions are telling you that there is a six dimensional space of transformations which can be performed on a vowel utterance in order to encode information about how loudly it is spoken, the age and sex of the speaker, and so on.

(a) How might you test this hypothesis?

(b) If you were satisfied that the hypothesis is tenable, how would you use the data in order to normalise a new trajectory in the vowel space?

(c) What about a trajectory representing a word such as " wish" or "whiff"?

(d) Construct some data satisfying the conditions described and examine it carefully, comparing it with real speech data.

7. Consider the double spiral data set of the last chapter. Although somewhat implausible as a model for some transforms of a basic category, it is plain that there is a certain similarity between the case of an embedding of an interval arising from some family of transformations and the spiral structure of the data sets. In fact one *could* represent the spirals as the embedding of an interval. The question arises, can one extend the arguments used to deal with embeddings of manifolds which are not too far from being flat, to cases such as the double spiral in which there is plainly some kind of structure to the data? It is plain that we could perform the same trick as before: first choose a suitable radius, then compute covariance matrices for points lying inside balls of the radius, use the coincidence of all of them having one eigenvalue much bigger than the other to argue that the data is essentially one dimensional, and then try to fit a curve. It is the last part which will give some troubles to an automatic system; doing it with polynomials does not usually give good extrapolation properties. Try it and see. [18]

[18]You were warned about function fitting. Don't say you weren't.

6.5 Bibliography

1. Peter Ladefoged, *Elements of acoustic phonetics*, Oliver and Boyd 1962.

2. Flanagan, James Loton *Speech analysis, synthesis, and perception*, Springer, 1972.

3. Norman J. Lass.(Ed) *Contemporary issues in experimental phonetics*, Academic Press, 1976.

4. Paul A Lynn, *An Introduction to the Analysis and Processing of Signals*, Macmillan, 1985.

5. Lawrence R.Rabiner, Bernard Gold, *Theory and application of digital signal processing* , Prentice-hall,1975.

6. Lawrence R. Rabiner, Ronald W. Schafer *Digital processing of speech signals* , Prentice-Hall, 1978.

7. Lawrence R. Rabiner *Fundamentals of speech recognition*, Prentice-Hall, 1993.

8. Wayne A. Lea, editor *Trends in speech recognition*, Prentice-Hall, 1980

9. D. R. Reddy, editor, *Speech recognition : invited papers presented at the 1974 IEEE symposium*, Academic Press, 1975.

10. Renato De Mori and Ching Y. Suen (Eds) *New systems and architectures for automatic speech recognition and synthesis*, (Proceedings of the NATO Advanced Study Institute on New Systems and Architectures for Automatic Speech recognition and Synthesis held at Bonas, Gers, France, 2-14 July 1984) Springer-Verlag, 1985.

11. Kai-Fu Lee (Foreword by Raj Reddy) *Automatic speech recognition : the development of the SPHINX system*, Kluwer Academic Publishers, 1989.

12. Y Linde, A Buzo and R.M. Gray *An algorithm for Vector Quantizer Design*, IEEE Trans.Comm. COM-28(1)pp84-95 1980.

13. Fred Jelinek, *Principles of Lexical Language Modeling for Speech Recognition*, IBM Report: Continuous Speech Recognition Group, IBM Thomas J. Watson Research Center, Yorktown Heights, NY 10598, USA.

14. Fred Jelinek *Self-Organized Language Modeling for Speech Recognition*, IBM Report: Continuous Speech Recognition Group, IBM Thomas J. Watson Research Center, Yorktown Heights,NY 10598. USA.

15. Lalit Bahl, Frederick Jelinek and Robert Mercer *A Maximum Likelihood Approach to Continuous Speech Recognition*, IEEE Trans Paattern analysis and Machine Intelligence Vol PAMI-5 No. 2pp179-190 March 1983.

16. L.R. Bahl, P.F. Brown, P.V. deSouza, R.L. Mercer and M.A. Picheny *A method for the Construction of Acoustic Markov Models for Words*, IEEE Trans. Speech and Audio Processing Vol.1 No. 4. pp443-452, October 1993.

17. Robert Linggard *Electronic synthesis of speech*, Cambridge University Press, 1985.

18. Gareth Lee *fview*, ftp site: ciips.ee.uwa.edu.au , IP no. 130.95.72.69, in directory pub/fview. 1993.

19. S. M. Bozic *Digital and Kalman filtering : an introduction to discrete-time filtering and optimum linear estimation*, Wiley, 1980.

20. T. Kailath *Lectures on Wiener and Kalman filtering*, Springer-Verlag, c1981.

21. Karl Brammer and Gerhard Siffling *Kalman-Bucy filters* , Artech House, c1989.

22. Walter Vandaele *Applied time series and Box-Jenkins models*, Academic Press, c1983.

23. Norbert Wiener *Extrapolation, interpolation, and smoothing of stationary time series : with engineering applications*, M.I.T. Press, 1949.

24. George E. P. Box and Gwilym M. Jenkins *Time series analysis : forecasting and control* , Holden-Day, 1976.

25. Maurice G. Kendall *Time-series* , Griffin, 1973.

26. Joseph L. Doob *Stochastic processes*, Wiley, [1953].