Data Driven Production Models for Speech Processing

Thesis by

Sam T. Roweis

In Partial Fulfillment of the Requirements

for the Degree of

Doctor of Philosophy

California Institute of Technology

Pasadena, California

1999

(Defended January 13, 1999)

Dedication

To my father and my mother (if only she could have been here to see this day) for teaching me by example to pursue the highest standards of excellence, perseverance and self-confidence in all things.

Acknowledgements

Many thanks to John Hopfield and his group at Caltech (and now Princeton) for providing an outstanding intellectual home for me over the last four years. Very special thanks to Carlos Brody, Sanjoy Mahajan and Erik Winfree without whom I never would have made it through the last months. Thanks also to Abeer Alwan and her group at UCLA, and especially to Brian Strope for generously sharing their speech expertise and providing advice on many aspects of my project. Pat Keating has also generously allowed me to use the resources of her laboratory at UCLA and has given me a patient and sympathetic ear on several occasions. Simon Blackburn, Dan Fain and John Hogden have all worked on or are still working on very similar endeavors to this one. They have been invaluable and unselfish colleagues from the start. Larry Saul and Mazim Rahim at AT&T, and Malcolm Slaney at Interval research stood up for my work even before it warranted such defending. Yaser Abu-Mostafa and Pietro Perona welcomed me into their research groups at Caltech and provided welcome advice about machine learning and pattern recognition. Dick Lyon agreed to sit on my committee and went far beyond the call of duty by patiently reviewing the thesis in wonderful detail and making numerous suggestions which greatly improved its scientific content and quality of writing. Finally, my great thanks to John Westbury and his team at Wisconsin, especially Carl Johnson for providing me with the database on which this work is based and for promptly answering my many questions over the past many months.

Abstract

When difficult computations are to be performed on sensory data it is often advantageous to employ a model of the underlying process which produced the observations. Because such *generative models* capture information about the set of possible observations, they can help to explain complex variability naturally present in the data and are useful in separating signal from noise. In the case of neural and artificial sensory processing systems generative models are learned directly from environmental input although they are often rooted in the underlying physics of the modality involved. One effective use of learned models is made by performing *model inversion* or *state inference* on incoming observation sequences to discover the underlying state or control parameter trajectories which could have produced them. These inferred states can then be used as inputs to a pattern recognition or pattern completion module.

In the case of human speech perception and production, the models in question are called *articulatory models* and relate the movements of a talker's mouth to the sequence of sounds produced. Linguistic theories and substantial psychophysical evidence argue strongly that articulatory model inversion plays an important role in speech perception and recognition in the brain. Unfortunately, despite potential engineering advantages and evidence for being part of the human strategy, such inversion of speech production models is absent in almost all artificial speech processing systems.

This dissertation presents a series of experiments which investigate articulatory speech processing using real speech production data from a database containing simultaneous audio and mouth movement recordings. I show that it is possible to learn simple low dimensionality models which accurately capture the structure observed in such real production data. I discuss how these models can be used to learn a forward synthesis system which generates spectral sequences from articulatory movements. I also describe an inversion algorithm which estimates movements from an acoustic signal. Finally, I demonstrate the use of articulatory movements, both true and recovered, in a simple speech recognition task, showing the possibility of doing true articulatory speech recognition in artificial systems.

Contents

List of Figures

List of Tables

Chapter 1 Introduction

1.1 Speech processing by machine

The dream of talking machines has long been emblematic of the quest for artificial intelligence. From early science fiction stories to the celebrated HAL-9000 computer in Arthur C. Clarke's *2001* [35] to the tritely named "computer" in the Star Trek series, the sentient machines we envision for the future have always had voices and ears. Conversant technology seems somehow much smarter to us than technology which merely performs complex calculations or superhuman feats of memory. But, as the saying goes, the future is now. As computers become ubiquitous the question of how to design a good interface between humans and machines becomes more pressing. A potential answer to this question which is rapidly moving from dream to reality is: what about the spoken word?

Speaking is the most natural method of communication between humans. It would be a great advance if our machines could also learn to speak and listen. For decades engineers have promised us that this technology is just around the corner. But so far, truly useful devices remain merely a promise. What would we want these speaking machines to do? Ideally they should be able to read aloud (convert text into an acoustic waveform), transcribe conversations (convert an acoustic waveform into text) and recognize the sex, identity, or even mood of a speaker (voiceprint analysis). These problems are called *speech synthesis*, *speech recognition*, and *speaker identification* respectively. Along with *speech compression* (which attempts to minimize the number of bits needed to store or communicate a speech signal without compromising the way it naturally sounds to humans) they are the central problems of speech processing by machine. Figure 1.1 illustrates these problems, showing which pieces of information are used as inputs and outputs in each case.

How do we know that these tasks are possible? The fact that parents can read stories to children, court stenographers can capture judicial proceedings and that friends and family members can identify each other on the telephone are all proofs by existence. The human brain is the repository of at least one computational platform and set of successful algorithms for speech processing problems. We may one day create machines that exceed our own abilities in speech processing,

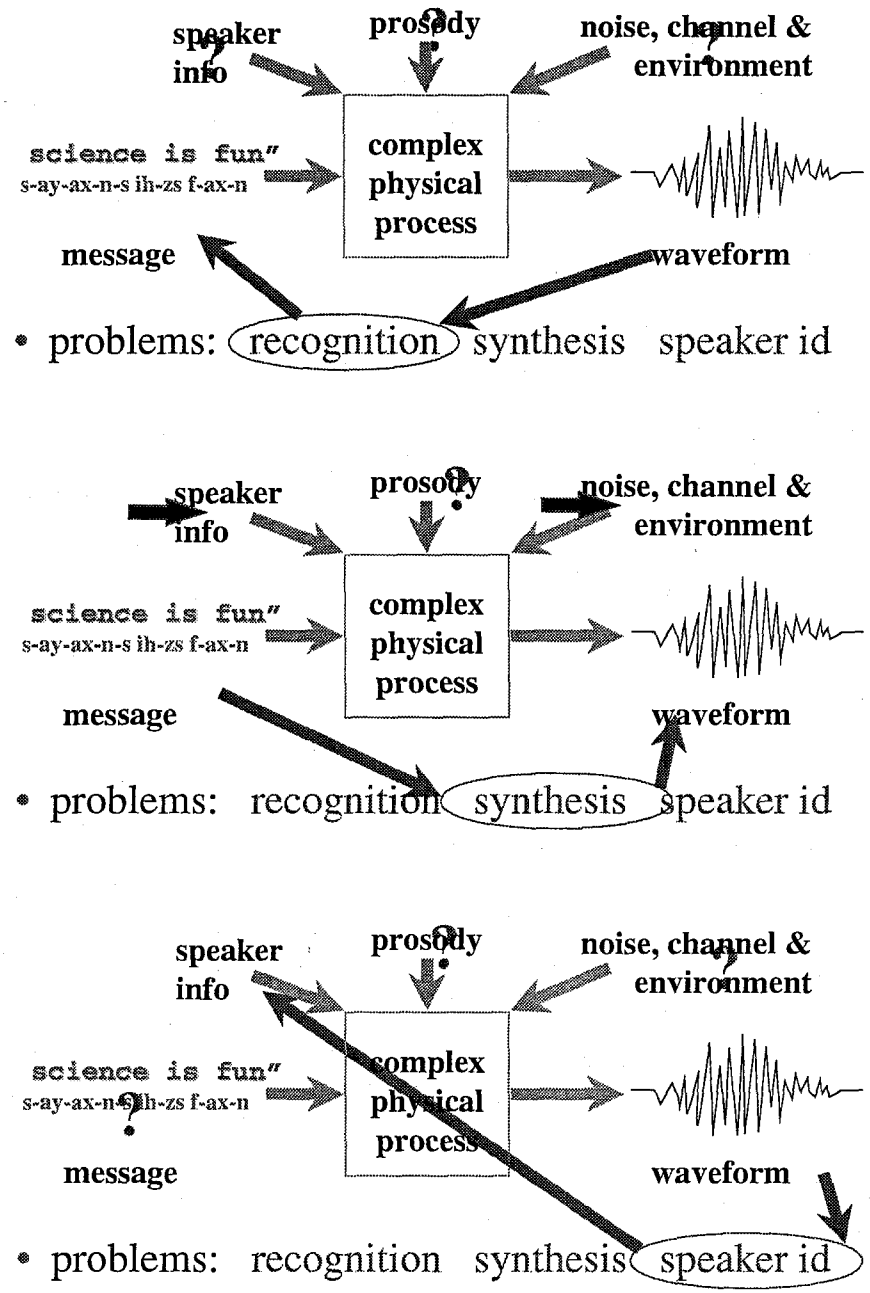


Figure 1.1: Three important speech processing problems. Diagrams show a schematic representation of the information available and sought (arrows) as well as information that is unknown (question marks) in the *recognition*, *synthesis* and *speaker identification* problems. In speech recognition (left) the goal is to recover a sequence of phonemes or words given an acoustic signal. Information about the speaker and acoustic environment are unknown. In the synthesis problem (middle) the goal is to produce an acoustic waveform given a text. Timing and prosodic information are unknown. In the speaker identification problem (right) the goal is to recover the sex, mood or even identity of a talker given an acoustic waveform. The text of what was said is often unknown.

but until that time, merely equaling human performance provides an excellent metre stick for evaluating systems.

Unfortunately, even the best speech synthesizers today sound extremely un-natural (the "talking moose" effect—hear audio A.4.1 below) and state-of-the-art speech recognizers get about half the words wrong in natural speech (see table A.4 below). In addition, when these tasks are made more difficult by adding noise and distortions, machine performance degrades dramatically whilst human performance remains almost unaffected [120, 2]. Our brains must be using a very different strategy than current machine algorithms to solve these problems. It is this persistent gap between the abilities of computers and of people that motivates speech researchers to search for new approaches and ideas.

1.1.1 Why study speech?

Speech processing by machine seems to be a rare example of a problem which is **intellectually very interesting** while at the same time being **extremely well defined**. It can be **easily worked on by an individual researcher with modest resources**, its solution will be **genuinely useful**, and there is a **real need for new ideas** in its research field.

The miracle of speech and audio processing in the human brain draws a confluence of admirers from an unlikely mix of disciplines. Engineers such as myself, eager to come up with algorithms to make sense of the elusively simple looking waveforms we digitize from our microphones, look to the brain for motivation because it is by far the most successful solution known. Neuroscientists, evolutionary biologists, linguists and phoneticians see speech as a uniquely human extension of sensory processing and communication strategies. They believe that studying how we speak and listen can teach us a lot about how our brains compute in general, how they developed and how that development has affected where we find ourselves today.

However, speech research is unlike engineering programs attempting to endow machines with human-like abilities to see (machine vision) or control movements (robotics). It is blessed – and cursed – by having **no major obstacles to success other than a profound lack of really clever new ideas**. Vision inherently involves an enormous amount of data which until recently could not be effectively processed by computers. In addition, although it is often assumed that the **input** should be static two-dimensional images or movies, it is not clear what the **output** of a vision system should be. Similarly disadvantaged, robotics drags the researcher into the frustrating and expensive realm of hardware headaches. And although the **output** is certainly a set of joint torque

commands over time, it is difficult to cleanly formulate the **input** requirements of the problem.

Speech, on the other hand, is a one-dimensional time series—like a single monochrome pixel of a vision problem. We have an enormous amount of high quality and virtually free data[1] which is furthermore standardized across all research groups. The inputs and outputs of speech recognition, speech synthesis, speech compression and speaker identification problems are all well defined. Furthermore, in the case of speech recognition and speaker identification (and increasingly so in the case of synthesis and compression), we have quantitative error measures that can be used to directly compare different systems on standard databases. From a computational point of view, the rapid advance of technology has meant that researchers have, at least recently, been able to try out essentially any algorithm they entertained. The current research machines used at AT&T labs, for example, have **ten gigabytes** of physical memory allowing them to store literally hundreds of hours of speech training and testing data in core. In other words, we have everything we need except the algorithms. And to make things more exciting, we know such algorithms do exist because the human brain exhibits them daily.

Once machine intelligences which solve speech processing tasks as well as humans or better can be developed, an amazing array of new technologies—some of which are now in their infancy—will become available to society. The most obvious are blind reading and hands free typing. The ability to effectively receive a text without using one's eyes or to create one without using one's hands will clearly be of enormous benefit to those with handicaps or disabilities that hinder conventional reading and writing. But there are other exciting applications: imagine sending and receiving faxes or email from anywhere by using a telephone to connect to a speech processing computer. Perhaps one would like to have a computer in the car search for interesting parts of the New York Times and then read them aloud, transcribing comments made along the way by the listener. Taking notes at group meetings, luncheon appointments and public lectures will be a thing of the past. Another important application of speech technology comes from the relentless miniaturization of electronic devices. Soon there will be wristwatches which are full fledged computers. How will we get input to them? The famous comment on this point is that, "you can't type with toothpicks".

Truly amazing things can also be envisioned. For example if machine translation continues to progress and speech processing is ready for it, the day will soon come when it is possible to

[1]The Linguistic Data Consortium at the University of Pennsylvania distributes many CD-ROMs of speech databases annually to its members for only a few thousand dollars per laboratory. See `http://www.ldc.upenn.edu/`.

build a real-time interactive translation system that allows people who do not speak each other's languages to converse. When one party utters a phrase, a computer does speech recognition to convert the audio into words in their native language. A machine translation system converts these words into words in the language of the other party which are then synthesized into speech.

Robust speech recognition is really only the first step in the larger goal of **speech understanding**. For spoken dialogue systems to be effective, machines need to correctly interpret and respond to spoken input. Such "intelligence" is indeed the ultimate goal of much speech processing research. Success at this level depends crucially on advances in natural language processing. However, the problems of understanding and recognition are not decoupled – there is predictive feedback from understanding which helps us in recognition. While it is clear that human recognition abilities are greatly aided by such high level contextual feedback, our performance still far exceeds that of machines even on nonsense sentence and nonsense word tasks, especially in the presence of noise or unknown filtering [2].

An excellent example of such a speech understanding system is the JUPITER weather information server at MIT. A user calls the telephone number 1-888-LSD-TALK and is connected to a real-time speaker independent application that provides information about climate and weather all over the world. Audio example 1.1.1 is a recording of the author interacting with an early version of JUPITER.

Audio file 1.1.1 (`track 1`) *A telephone call to the* JUPITER *interactive weather information system.*

1.1.2 Why articulatory (speech production based) models?

A potent objection to speech recognition techniques as they exist today is the lack of speech-specific knowledge in them. The existing models could be (and are) equally well used to predict financial time series, analyze PET data or to classify protein and DNA sequences (see for example [52, 38, 196]). By incorporating prior information about the nature of speech, the problem can be heavily constrained. Such regularization is essential to achieving more robust and accurate performance than current approaches permit.

One way to view automatic speech recognition is as a code-breaking problem [96, 95]. There is an **unknown message** (say a sequence of phonemes) which has been **encoded** into a pressure versus time waveform by the evil wizard of speech production. The goal is to **decode** the wave-

form and recover the message (see figure 1.2). Three important sources of prior information may be used to aid in this decoding. First, a considerable amount is know about the **set of possible messages** because languages have very strong syntactic structure; this motivates work on *language modeling* for use in speech processing systems. Second, there is information about the **receiver of the code** from studies of human perception; this allows one to pre-process the waveform by emphasizing perceptually relevant features.[2] Such studies motivate short-time spectral analysis as an almost universal pre-processing step in recognizers, speaker identification systems and compression algorithms. Finally, much is known about the **producer of the code** from studies of speech production; this enables one to assert that the message cannot be embedded in the audio signal in particular ways because the producer could not possibly have performed such an encoding. The ultimate aim of the present work is to use speech production models to improve speech processing systems.

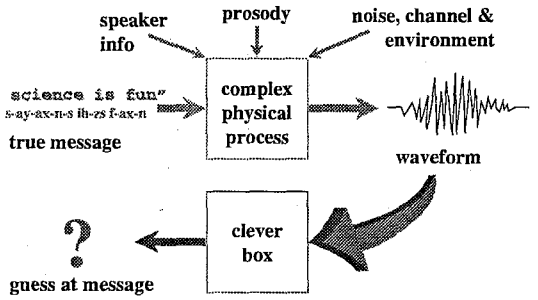


Figure 1.2: A view of speech recognition as a code-breaking problem. There is an unknown message (say a sequence of phonemes) which has been encoded into a pressure versus time waveform by a complex physical system. The goal is to decode the waveform and recover the message.

What does speech production have to offer engineering? For an answer one has only to look at the approach taken in any other engineering problem that involves estimating certain unknown quantities from noisy observations or measurements. Universally in such cases, signal processing practitioners employ a **model** of the system under investigation. Typically the model has certain internal *state variables* which control the output being measured or observed. Armed with a mathematical description of the process that generates the data, an informed analyst can attempt

[2]Many pre-processors **discard** certain information about the waveform. Continued intelligibility by humans after such discarding is one (but by no means the only) way to ensure that such pre-processing does not render the unknown message unrecoverable. In other words if we know that a certain manipulation of the acoustic signal leaves it still intelligible to humans, then we can be sure that it has not destroyed **essential** information. However, it may certainly have destroyed **useful** information. Humans are quite amazing recognizers, and so it is not a very strong statement to say something is still intelligible to us.

to *infer* the internal states of the model from the observations. This inference-through-a-model step is an extremely common first stage in analysis of especially time series data. For example, the celebrated Kalman filter [103, 104] does exactly this sort of state estimation for linear dynamical systems. A judicious pattern recognition approach might then be: (1) pre-process and extract features from the incoming data; (2) using this sequence of features and the process model, do state inference; (3) using the inferred states (possibly along with the original feature sequences) do pattern recognition.

This usage of a model is nowhere present in most current speech processing systems. Instead, the current approach seems to be: (1) pre-process and extract features from the incoming data; (2) try to do pattern recognition directly on these feature sequences. At this point a reader somewhat familiar with current speech processing techniques (see Appendix A) may object. "But a hidden Markov model *is* a model for generating the data," one might argue. While it is true that HMMs do have a generative model interpretation, they are the pattern recognition technology portion of the system algorithm and not a true model of data generation. Recognition systems universally have hundreds of separate HMMs, one for each token (e.g. , triphone) to be recognized. Yet there is only one speech production process at work. In a system that is trying to model this generation, only one model of the process should be used for **all** the data. Furthermore, state inference is of no interest in conventional HMM speech analysis[3] because it is always known that the model will progress from left to right. What is of interest is which of the models will assign the highest likelihood to the incoming sequence. This is already a pattern recognition question.

Why is using a model so important? A good model allows us to encode in a general way those few pieces of prior knowledge that we have about the problem. For example, we know that human speech is invariant to a certain amount of time warp in the rate at which the basic sounds are produced or perceived. It also allows one to use the rich dynamics of a system to advantage rather than being confounded by them. For example, the effect of *co-articulation* in speech processing, long viewed as a curse of unwanted variability, is in fact, in the present view, a blessing because it provides important information about what is happening to the states of the underlying production system. This point has been made by Church and others [34, 135] with reference to machine algorithms and by Liberman and others with reference to human perception. Consider the following excerpt from Liberman:

For a motor theory, on the other hand, systematic stimulus variation is not an

[3]Except for forced alignment in which case the timing of the state transitions is of interest.

> *obstacle to be circumvented or overcome in some arbitrary way; it is, rather, a source of information about articulation that provides important guidance to the perceptual process in determining a representation of the distal gesture.* [118] pp.14–15

In particular this thesis draws upon the **continuity**, **smoothness** and **low dimensionality** of the underlying states as essential constraints from speech production models. Because humans speak using mouth movements that are for the most part slow and smooth,[4] the complicated acoustic signals which we observe should ultimately be explainable by slowly varying underlying state variables (the articulator movements). Furthermore, since the articulatory system has a limited number of degrees of freedom (estimates vary from 3 to 10), the number of such independent variables required should be small. Such explicability constraints are an important feature of speech signals and are not obviously true of, for example, protein sequences, PET data, or financial time series.

1.1.3 Psychophysical and linguistic motivations

The engineering advantages presented above motivate the use of speech production models from a pattern recognition point of view. In addition there are several pieces of evidence which suggest that inversion of speech production models plays an important role in the way humans perceive and recognize speech. This idea has a long history in the speech science literature. The most visible recent proponents are Liberman and colleagues from Haskins Laboratories in New Haven (affiliated with Yale University). Their *motor theory* of speech perception [115, 118] posits that speech perception explicitly involves recovery of articulatory movements and that recognition involves comparing deduced articulation with memorized gestural templates. Carol Fowler's *direct realist* theory of speech perception ([64, 65]) makes the stronger[5] claim that listeners perceive vocal tract gestures "directly, that is unmediated by processes of hypothesis testing or inference making" ([64] p.1731). (Fowler is also at Haskins, which has contributed to the somewhat unfair coupling, in the minds of many speech scientists, of these theories as almost identical.) While these two theories are currently the most influential, earlier authors shared this gestural view of speech perception [49, 97, 180].

There are also several pieces of direct psychophysical evidence showing that articulatory re-

[4] While individual articulators may at times make abrupt movements, in general there is little power above 20 Hz in measured traces. See for example [201, 132, 137].

[5] And in the view of the present author, poorly defined.

covery is involved in human perception. (The various theories mentioned above merely hypothesize this to be true. The theories are, of course, motivated by the following and other pieces of direct evidence.) The strongest and most well known of these is the *McGurk-MacDonald effect* [130, 126], discovered at the University of Surrey, Guildford, by Harry McGurk and John MacDonald, two developmental psychologists interested in how infants reconcile conflicting information in different sensory modalities. The effect occurs when subjects perceive conflicting auditory and visual information from a speaker. It was originally discovered by creating a video tape on which the movie of a talker's face corresponded to one sound while the audio track corresponded to another.[6] Listeners perceived sounds which were strongly influenced by both the auditory and visual information. Furthermore, the effect is not lessened by subjects' knowledge of the underlying trick. The percept is one of a unified sound though a sound which is different from either the pure auditory or pure visual stimulus. For example a visual `/ga/` and an auditory `/ba/` are together perceived as `/da/` or `/za/`. Figure 1.3 shows a cartoon of this effect.

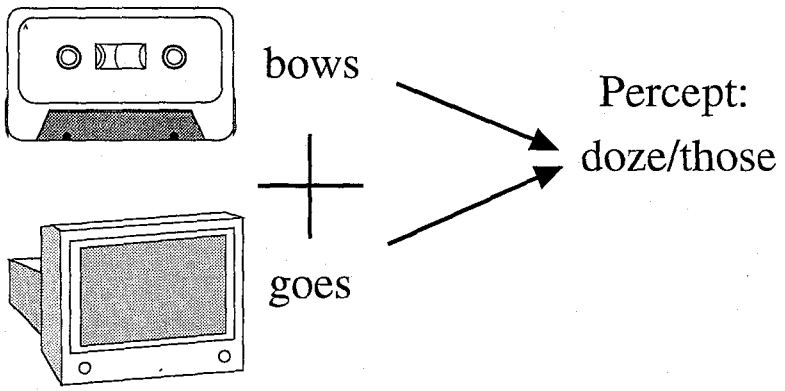


Figure 1.3: The McGurk-MacDonald effect. The effect occurs when subjects perceive conflicting auditory and visual information from a speaker. It was originally discovered by creating a video tape on which the movie of a talker's face corresponded to one sound while the audio track corresponded to another. Listeners perceived sounds which were strongly influenced by both the auditory and visual information. The percept is one of a unified sound though one which is different from either the pure auditory or pure visual stimulus. For example a visual `/ga/` and an auditory `/ba/` are together perceived as `/da/` or `/za/`.

This effect highlights the use of speech production information in perception by humans. Whatever extra information is being provided to the listener by the visual stimulus is certainly

[6]The discovery of this psychophysical effect was accidental; however, it is unknown if the creation of conflicting stimuli was also accidental. The original purpose of the stimuli was establish which modality dominated at different developmental stages. A tape with a visual `/ga/` and an auditory `/ba/` was produced. To the surprise of all, `/da/` was perceived even by adults prompting the belief that the technician had made an error in creating the tape.

being incorporated into the perception process at some level by virtue of the fact that it fundamentally changes what is heard.

Another important piece of psychophysical evidence comes from early studies that indicated that the acoustic patterns of synthetic speech had to be modified considerably in different contexts in order to yield an invariant phonetic percept to the listener [37, 116, 117, 115]. For example, in a consonant-vowel (CV) syllable the place of constriction is indicated by the transitions of the formants (peaks in the speech spectrum) after release. However, the same consonant can indicate place of constriction with a rising of formants in one vowel context and a falling in another. Furthermore, by themselves these formant transitions sound like "chirps" and not consonants. When played to listeners in isolation, the transitions "do not sound alike, and, just as important, neither sounds like speech. How is it, then, that, in context, they nevertheless yield the same consonant?" [118]. The obverse effect is also observed: the same acoustic cues in different contexts can lead to different percepts. So perceptual invariance is not necessarily tied to acoustic invariance. This led motor theorists to posit that it might be tied to "gestural" invariance, i.e. invariance of the underlying articulator movements or of the control commands that caused them.

A third important piece of psychophysical evidence has to do with the perception of acoustic stimuli as either speech or non-speech. If synthetic stimuli are created in which formants vary slowly and smoothly but in linguistically abnormal ways, what is perceived is amazingly a **mixture** of synthetic "speech" and background chirps and noises [119]. It seems as though formant trajectories that correspond to plausible articulatory movements are perceived in a way categorically different from those that do not.

1.1.4 Personal inspirations

Motivating a direction of work is always a curious business. Those who agree with an approach rarely need convincing while those who are opposed to it generally will not be swayed by anything but definitive defeat of all other paradigms. Nonetheless, I feel that a few comments to the reader are in order regarding why I have chosen this somewhat disreputable avenue of investigation.

In 1969 John Pierce wrote a caustic letter in the Journal of the Acoustical Society of America titled *Whither Speech Recognition?* which criticized speech researchers for their unscientific behaviour. Many of its points are still valid today:

> *... if those who engage in recognition showed more signs of an effective effort*

to learn something about speech and fewer signs of rapture for computers and for unproven schemes for, and theories of, recognition.

We all believe that a science of speech is possible, despite the scarcity in the field of people who behave like scientists and of results that look like science. Most recognizers behave, not like scientists, but like mad inventors or untrustworthy engineers.

We would expect a scientist to check the literature concerning ideas, schemes, or information. Perhaps some point is old and has been established or confuted by a clear, simple, definitive experiment. If there is no clear experimental evidence, it might be possible for a scientist to devise a simple, clear, definitive experiment. So, a science of speech might grow, certain step by certain step. [149]

In 1995 Hervé Bourlard gave a keynote address at the Eurospeech meeting in Madrid entitled *Towards increasing speech recognition error rates.* Although I was not in Madrid, this talk was written up the following year as a now famous paper in the journal *Speech Communication.* In his talk and in their subsequent paper, Bourlard, Hermansky and Morgan convincingly argued that the endless pursuit of low word error rates on standard databases is suppressing new ideas in the field of speech recognition and forcing research into a competitive feedback loop from which escape by innovation is difficult. They issued an encouraging call for new ideas even though they might initially result in lower bottom line performance:

In the field of Automatic Speech Recognition (ASR) research, it is conventional to pursue those approaches that reduce the word error rate. However, it is the authors' belief that this seemingly sensible strategy often leads to the suppression of innovation. The leading approaches to ASR have been tuned for years, effectively optimizing on test data for a local minimum in the space of available techniques. In this case, almost any sufficiently new approach will necessarily hurt the accuracy of existing systems and thus increase the error rate. However, if progress is to be made against the remaining difficult problems, new approaches will most likely be necessary.

...

We encourage the reader to incorporate both the specific research directions we have described here and others that we have not mentioned (and/or are not aware of) to branch out and better explore the space of possible solutions to problems in ASR. In all of these cases, there is much more unknown than known, and so each new

> *attempt to deal with these issues is likely to result in a higher error rate for initial tests. We hope that the reader will have the patience to wait through such a period in order to provide deeper understanding of the issues that good experiments can bring. The current level of difficulty for mastery of the larger systems is so high that some in the field have tended to discourage newcomers. However, we believe that ASR is still a field in which many of the basic problems are still barely touched, and in which new ideas from newcomers will be critical. We should welcome these ideas, even if they increase the error rate.* [24]

This basic message rang true to me. The problem of speech processing and the idea of articulatory methods had been introduced to me by John Hopfield, Unni Unnikrishnan, Marcus Mitchell and other members of our research group. The new approach seemed exciting and, coupled with my belief, outlined above, that speech processing is a good field in which to work, I decided to pursue an engineering reincarnation of an old idea in linguistics and speech science whose time I believed had come again: *articulatory speech processing.*

1.2 Machine learning

There are many contrasting approaches to designing algorithms for processing sensory data. Biology has chosen an evolutionary paradigm in which both the computational platform and the algorithm being executed incrementally change through random search and reinforcement learning. The success of this process ultimately hinges on the enormous time-scales on which it works. Genetic algorithms are a computational attempt to mimic this evolutionary design process in the space of computer programs running on a fixed architecture. On the other side of the intellectual spectrum are traditional artificial intelligence or rule-based expert systems which herald from the belief that explicit system design by a skilled human is a promising direction. In such systems, human knowledge about a problem is codified into various logical propositions or rules inside the underlying computer program. Unlike biological or artificial evolution, the design process is not carried out by self modification of unsuccessful systems. Rather it falls to the inspired self-reflection of the defeated creator to remake the system by correcting its apparent failures. Such failures can often be extremely difficult to diagnose let alone to fix.

The approach embraced in this thesis is instead one of **learning from example data**. The underlying belief is that for many computational tasks of interest there are strong **statistical pat-**

terns present which relate inputs to desired outputs and to each other. One sets up a system in which adjustable parameters control the computation. Furthermore, the system is designed so that parameter modifications **that improve performance on the task** can be easily computed by repeated analysis of example inputs and outputs. The process of adjusting parameters based on examples is called *training* the system; the examples used are known as the *training data.* After training, performance can be evaluated by *testing* on new (previously unseen) data.

Many fields lay claim to some aspect of this data-driven approach. Statistics, signal processing, approximation and optimization theory and artificial neural networks are all examples. I will use the generic term *machine learning* although it is not the name that is important but rather the basic idea: extracting statistical regularities from example data in order to solve computational problems.

1.2.1 Why use machine learning (data driven statistical) methods?

There are many situations in which we would like a digital computer or electronic circuit to perform a task that implicitly involves a complex computation. For example, the answer to the question "Is this photograph of a face your mother's likeness?" requires the computation of a difficult boolean valued function on the input space of images. The key point is that there are very many scenarios in which it is **essentially impossible** to write down *a priori* an equation or procedure that computes the function of interest. What is the mathematical expression for the set of all images of smiling people or of Popes holding babies or movies of people waltzing or of signals that represent someone saying your name? The difficulty here is not a question of requiring more knowledge about the problem or more precise measurements of physical constants or more accurate simulations of dynamics. Rather it is a struggle to find an approach to the problem that can be effectively pursued. Yet human exhibition of myriad sensory processing skills tells us that two (perhaps surprising) things are certainly true of such tasks: (1) they can be solved and (2) solving them is something that can be **learned** in a finite amount of time from a finite number of examples of correctly solved instances.

In the absence of a structurally motivated, principled approach to a problem (or sometimes in the presence of a strong belief that such an approach is fundamentally unattainable) we turn, as it seems the brain has, to the procedure outlined above. Collect a large training corpus of examples and feed them to a system which incrementally updates some internal settings as it learns to perform the computation in question. The instantiation of this general paradigm in specific

computational algorithms has come to be called *machine learning*.

1.2.2 Why apply machine learning to speech production?

It is very difficult to develop detailed physical models for complex systems in which nonlinear dynamics and noise effects are paramount. Speech production is a prime example of such a system. Turbulence effects are essential to the production of fricative sounds for example, but it is extremely difficult to implement a model that correctly predicts their occurrence in a vocal tract (see for example [174]).

On the other hand, it has recently become possible to collect large amounts of actual data from such a physical system; in effect John Westbury and his team (see chapter 3) have used reality as their simulator. Given a resource such as their large parallel database one is in a perfect position to apply statistical pattern recognition and function estimation techniques and ask empirically: **Can statistical relationships between acoustics, articulation and phoneme sequences be observed in the data themselves?**

Chapter 2 Summary of Contributions

2.1 A geometrical interpretation of the thesis

Consider a geometrical interpretation of human speech data. As the movable structures of the mouth – called *articulators* – change positions and air flows from the lungs through the cavities so created, a complex sound pressure wave is created. If we call the shape of the speaker's mouth at any time the *articulatory configuration* and if we characterize the the speech wave using some short-time acoustical features such as spectral measurements, then we have a system in which two configurations – articulatory and acoustic – evolve in parallel over time. Each spoken utterance is a sequence of articulatory configurations (mouth shapes) accompanied by a sequence of synchronized acoustic features (generated by some kind of sort-time spectral analysis of the audio wave). (Each utterance also has associated with it an ordered list of words or phonemes corresponding to the orthography of what was spoken.) These sequences can be thought of as paths or trajectories through an articulatory space and the acoustic-spectral space. Figure 2.1 shows a cartoon of these paths in the two spaces.

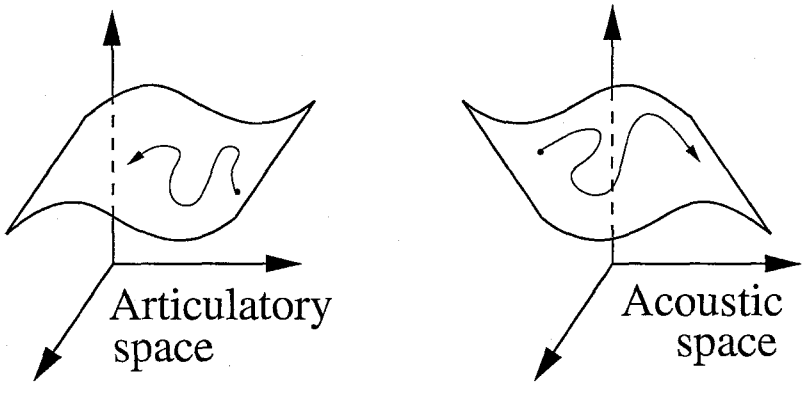


Figure 2.1: A geometrical interpretation of human speech data. Any utterance can be thought of as a pair of paths, one through an articulatory space and the other through an acoustic-spectral feature space. The paths evolve synchronously in the separate spaces and generally lie within the *manifolds* on which most natural speech data falls. Estimation of these manifolds and analysis of how the trajectories in the two spaces are related to each other are the questions addressed in this thesis.

The various investigations with which this thesis is concerned can be thought of as various

geometrical questions about these paths:

1. What parts of their respective spaces do the acoustic and articulatory paths fill? This is a time-independent data modeling question which asks: **What are the possible configurations of the mouth and what are the possible sounds it makes?** Chapter 5 addresses this first question by modeling the observed articulatory configurations from a large database discussed in chapter 3. It shows that simple linear models with very few degrees of freedom can much of the structure present in the original data.

2. If we are near a particular point in one of the spaces, where are we likely to be in the other space? This is again a time-independent question which asks: **What is the instantaneous mapping (if any) between the shape of the mouth and the sound it is making?** Chapters 6 and 8 investigate these questions by constructing a forward model that predicts sounds from mouth shapes (chapter 6) and by illustrating with examples from actual speech data that the instantaneous inverse mapping problem (from sound to mouth shape) is ill-posed and cannot be solved (chapter 8).

3. Given a trajectory in one space, what is the most probable corresponding trajectory in the other space? In other words **Can you hear the movements of a mouth?** or **Can you predict the speech sounds a moving mouth ought to make?** These problems have been called *acoustic to articulatory inversion* and *articulatory speech synthesis* respectively. Again, chapters 6 and 8 discuss these questions. In chapter 6 the improvements to the instantaneous forward model from using timing and derivative information are investigated, and are found to be quite small, suggesting that for this simple forward model, and instantaneous mapping suffices. In chapter 8 a system for recovering articulatory movements from acoustics is proposed and tested on real speech data. It recovers the (midsaggital) movements of articulators to within a few times the measurement error present in the data, and well enough to perform successful speech recognition on a simple word spotting task (see below).

4. Given a trajectory in articulatory space, what is the most likely corresponding sequence of words or phonemes? This is what I will call *articulatory recognition.* Chapter 9 presents preliminary investigations along these lines and suggests directions for future research. I show that using the *true* articulatory movements, a simple template matching system can perform basic word spotting. I further show that articulatory movements *recovered from*

acoustics alone (using techniques from chapter 8) can be used to successfully perform the same word spotting task. This last result represents a proof of concept example of a true articulatory speech recognition system that first recovers movements from acoustics and then does pattern matching based on those movements.

Some important and interesting questions that I do not address directly in this work but are active research questions in speech processing are:

1.x) What parts of sentence space do the word sequences fill? This is *language modeling.*

1.x) What parts of babble space do the phoneme sequences fill? This is a question which belongs to the pronunciation modeling and language characterization fields.

4.x) Given a trajectory in acoustic space, what is the most likely corresponding sequence of words or phonemes? This is the long standing problem of *automatic speech recognition by machine.* The suggestion of this thesis is that a novel solution to this problem might be to first estimate the articulatory trace as per question (3) above and then use the recovered movements as in (4) above to aid in guessing the word sequence.

5.x) Given a sequence of words or phonemes, what are the most probable corresponding articulatory trajectories? In other words, what strategies do talkers use to articulate sounds. This is a problem in the realm of *phonology* and *linguistics.*[1]

5.x) Given a sequence of words or phonemes, what is the corresponding acoustic trajectory? For speech scientists this is the central question is the study of dialects and pronunciation patterns. For engineering this is the problem of *speech synthesis.*

2.1.1 Possible configurations

The most basic question is about speech production data is: what parts of the articulatory and acoustic spaces do the observed paths fill? In other words, the set of all observed trajectories define two low dimensional manifolds in the full spaces. What is the nature of these manifolds? What are their dimensionalities? Can we develop models that capture them effectively? Chapter 5 answers these questions using a large amount of speech production data from a database described

[1]For example see the work on articulatory phonology by Browman and Goldstein [27, 28].

in chapter 3. In both cases the answer is that such manifolds **do** exist, as our intuition would suggest. Speakers cannot place their mouths in arbitrary configurations nor can their mouths produce arbitrary short-time spectra. In the case of acoustics, the spectra are generally very smooth: the power at one frequency is generally very similar to the power at nearby frequencies.[2] Thus, it has long been the practice to use smooth, all-pole approximations (such as low-order linear prediction or cepstral analysis—see Appendix A) for modeling the short-time spectra of human speech. In the case of the articulators, observed data are also highly constrained. Physical (mechanical) restrictions prevent many configurations from occurring. Furthermore, speakers simply do not employ certain physically possible but linguistically unusual mouth shapes.

The essential result of this investigation is that these constraints are not at all difficult to discover. Extremely basic techniques such as principal component analysis (see Appendix B) yield models which are interesting and useful for practical data analysis. With between four and six degrees of freedom, these models can represent the original data to within the same precision as the measurement error of the X-ray machine. The models can be used to define pseudo-mechanical models of the mouth and to fill in missing portions of the database. The structure in the data is very clear; this is a situation in which almost any data-driven approach would have discovered a large amount of statistical regularity. It is merely a question of actually having tried such an approach on a large amount of real speech production data.

2.1.2 Instantaneous mappings

Once we have a large amount of simultaneous acoustic and articulatory data in hand, it becomes possible to simply **look** at what is there and answer questions about instantaneous mappings between acoustics and articulation. My approach is to ask if "similar" configurations in one space imply "similar" configurations in the other. Do all times in the data at which the mouth is very close to a certain configuration have similar acoustics? The answer as shown in chapter 6 is yes. Simple linear models can be constructed to predict the instantaneous spectrum of the speech from the articulator configuration at that time. The inverse question is: do all times at which the instantaneous spectrum is very close to a certain spectrum have similar articulatory configurations? The answer is no; chapter 8 shows many examples taken from the database of very similar spectra

[2]To a good approximation the vocal system produces its pressure wave output by filtering noise or pulse excitations. If we define the "transfer function" of the vocal tract to be the ratio of volume velocity at the output to volume velocity at the input then physically it cannot have any zeros. However, the linear filtering assumption is not perfect and so it is still empirically useful to model the observed spectra using a small number of both poles and zeros to give a better fit.

which have very different articulator positions. Of course the key to answering these questions properly is to define "close" or "similar" correctly. In the case of articulators we use a distance measure based on the eigenvectors of the data covariance; this is shown in chapter 5 to be an effective metric for compressing and reconstructing data. In the case of the short-time spectral features we employ a similar distance metric in the space of line spectral pairs (LSP) as discussed in chapter 8.

Again, this analysis is extremely rudimentary—the forward mapping can be done with nothing more than linear regression and the inverse question can be answered by a simple program that searches through the database for matching frames based on a basic covariance distance measure. However, these questions have been debated for some time in the speech processing field for various models and assumptions. The contribution of this thesis is to present an **empirical** answer to the question culled from actual speech production data that we are now in a position to analyze.

2.1.3 Articulatory to acoustic trajectory mappings: talking heads

Since the instantaneous forward mapping is relatively successful, the question of interest here is how much improvement can be gained by using timing information. My approach has been to simply add instantaneous derivative information (velocities and accelerations) to the position information and investigate the additional predictive power of this expanded state. Chapter 6 shows that derivative information does not significantly improve the quality of acoustic recovery. This means that for the limited articulatory data at hand (i.e. midsaggital information only) and for the relatively simple methods employed (linear models and mixtures of linear models), the forward trajectory problem can be well solved by employing the instantaneous forward mapping learned previously without making use of trajectory information.

Such estimated spectral parameters can be inverted to reconstruct time domain waveforms, thus recovering speech utterances from articulatory movements alone. This achievement of a successful data-driven "talking head" is important for two reasons. First, it shows that **in the case of normal speech production** it is possible to go from a silent movie of articulator movements to an understandable synthesized audio track. Second, it demonstrates that, although detailed physical simulations for articulatory synthesis are extremely difficult and computationally demanding, there exist simple fast algorithms that perform fairly well based only on statistical analysis of example data.

2.1.4 Acoustic to articulatory inversion: can you hear the movements of the mouth

The strong empirical evidence that the instantaneous inverse mapping problem is ill posed suggests that the dynamic equivalent will also be very difficult. However, the existence of a very simple yet relatively accurate forward model is encouraging. An approach to dynamic inversion based on a new model known as a *self-organizing hidden Markov model* is presented in chapter 8. The approach can, **to a certain degree**, successfully recover articulation (as measured in the database) from acoustics. How good does recovery have to be to consider the problem solved? I believe that the only reasonable answer to this question lies in the **application** of recovered movements to a recognition, compression or speaker identification task. Chapter 9 presents investigations of these applications using both the true and recovered movements. Limited success has been achieved in the recovery of articulator movements: the system can recover movements to within a few times the measurement error of the data. More importantly, however, articulatory movements recovered from acoustics alone were used to successfully perform a simple word spotting task. At the present time I still consider acoustic to articulatory inversion an open problem, although it is one into which investigation is bound to be fruitful considering the considerable advances in understanding, data availability and modeling which have come out of other parts of this work, especially an example of a true articulatory speech recognition system.

2.1.5 Applications: articulatory speech processing

The final chapter of the dissertation presents some preliminary investigations and ideas on the subject of how articulatory methods might be brought to bear on "practical" speech applications such as recognition, synthesis, speaker identification and coding. Comparisons with existing systems are **not** presented on any aspect of performance. Rather, an attempt is made to outline ideas about how these problems could be tackled using an articulatory approach. In many cases, for example for recognition and speaker identification, preliminary experiments have been performed showing that **some** sort of articulatory system can be constructed to solve a certain speech processing task. The hope is to stimulate thought and excitement about future research directions and decidedly *not* to make strong quantitative claims about the promise of articulatory methods. The word spotting system described above which uses only recovered articulatory movements represents exactly such a proof of concept—it is a simple task but it has been solved here in a fundamentally new way.

2.2 Two parallel outlines: a guide to the thesis

This thesis contains contributions to both speech processing and machine learning. The reader interested primarily in speech may concentrate on the following sections:

- Appendix A provides an brief introduction to basic speech processing terminology, definitions and algorithms.
- Chapter 3 introduces the X-ray microbeam database of simultaneous acoustics and articulator movements on which all of the analyses reported are based.
- Chapter 5 discusses how the observed articulatory configurations are constrained, quantifies the number of degrees of freedom present and develops a method for filling in missing information in the database.
- Chapter 6 develops a simple instantaneous mapping between articulator positions and acoustic features and includes several examples of utterances resynthesized from only articulator movements.
- Chapter 8 discusses the problem of recovering articulator movements from acoustics.
- Chapter 9 concludes with some investigations of articulatory speech recognition and speaker identification using the true articulatory movements. It discusses directions for future work and presents an analysis of certain similarities between speakers which is very encouraging for the development of speaker independent models.

The reader interested primarily in machine learning is directed to:

- Appendix B provides a brief introduction to machine learning methods.
- Chapter 4 gives a detailed description of an EM algorithm for principal component analysis (PCA) that is extremely efficient for finding a few eigenvectors in very large, very high-dimensional datasets, as well as a probabilistic extension of PCA that defines a proper density model in the data space.
- Chapter 7 introduces a new model called the Self-Organizing Hidden Markov Model (SOHMM) that endows HMM states with a topology similar to a Kohonen map but through time. This model is useful for learning low-dimensional maps to explain high-dimensional sequences of data.

- Chapters 8 and 9 include some new algorithms for state inference and learning in SOHMMs and for recognition of time series data.

2.2.1 Public data resources

As a result of this thesis, several datasets and code packages of potential use to future researchers have been produced. These will be publicly and freely distributed over the world wide web in the hopes that others can build upon the work started here. These resources include:

- A distribution of a subset of the Wisconsin microbeam database that includes filled-in sections of missing data, collected statistics and pre-trained models for articulator configurations and articulatory to acoustic mapping. A link to these data will be entered at the Articulatory Database Registry maintained by the Centre for Speech Technology Research at the University of Edinburgh.
  (`http://www.cstr.ed.ac.uk/artic/`)

- A MATLAB code package containing some functions for analysis and display of the Wisconsin data. A link to this package will be placed on my home page.
  (`http://www.gatsby.ucl.ac.uk/~roweis/`)

- A dataset of paired articulatory and acoustic sequences for use in the machine learning community to study supervised learning, time series analysis and the acoustic to articulatory recovery problem as an instance of inverse modeling. A link to these data will be entered into the UCI machine learning repository.
  (`http://www.ics.uci.edu/~mlearn/MLRepository.html`)

Chapter 3 Speech Production Data

3.1 Simultaneous data recording

Simultaneous recording of the speech wave and of articulator position information has long been a goal of scientists and engineers studying speech production [33, 56, 146, 94]. Early attempts involving full saggital X-ray movies captured only low quality images and audio, and were quickly abandoned because of health concerns regarding radiation dosages and the barium coating on tongue and lips used to make these structures more visible [59]. Recently, several "quasi-static" studies involving fMRI [8, 205, 187, 3] and ultrasound [183, 195, 182] have provided high spatial resolution, three-dimensional information about the shape of the tongue during production of various sounds. However, these techniques do not have the time resolution required to capture the complex articulator movements present during continuous speech production.

Several high temporal resolution technologies are available, including *electropalatography* (EPA/EPG) [79, 29], *electromagnetic midsaggital articulography* (EMMA) [147, 92] and *X-ray microbeam tracking* [69, 70, 188]. These techniques offer temporal sampling rates in excess of 100Hz, but typically only provided limited spatial information about the configuration of the mouth. For example, EPA only indicates points of contact between the tongue and palate.[1] (This is often valuable, however, since other techniques are not accurate enough to distinguish between contact and close separation of articulators.) EMMA and X-ray microbeam techniques track the positions of points defined by special physical markers adhesively attached to a speaker's articulators. They provide movement information about only a few discrete points on the articulators, usually restricted to the midsaggital plane (the plane whose normal is the line between the ears and which bisects the speaker's head). Nonetheless, these techniques are fast and accurate and allow experimenters to collect invaluable information about how human speakers produce the sounds of their language. Figure 3.1 shows an example of an EMMA system from the Massachusetts Institute of Technology. EMMA works by placing tiny coils on the articulators and applying a magnetic field to the head during recording. The induced voltages in the coils allows their movements to be inferred. Microbeam tracking on the other hand uses a controlled beam of X-rays

[1] Although it is possible to design electrode placements which indicate lip contact and other features.

which irradiates and tracks only the articulatory points of interest.

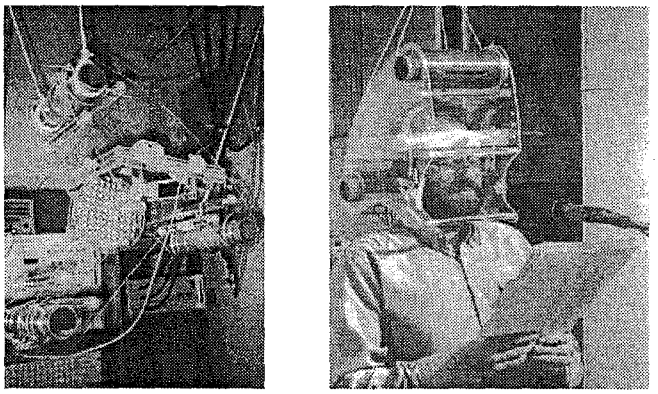


Figure 3.1: An example of a bead tracking system for simultaneously measuring the movements of speech articulators and the audio pressure wave. Shown on the left is the head mount for an electromagnetic midsaggital articulography (EMMA) system at MIT which tracks beads based on induced currents in a magnetic field. On the right is shown an example data collection session with a speaker reading from a text. The database used in this thesis comes from a machine (photos below) at the University of Wisconsin, Madison, which uses a different technology called X-ray microbeam tracking, but collects essentially the same data.

3.1.1 Microbeam database (UBDB)

The speech production data analyzed in this thesis comes from John Westbury and his team at the University of Wisconsin, Madison [201, 202, 1, 134]. The public X-ray microbeam database (UBDB from μ-Beam Data Base) they have collected and recently released is a truly remarkable achievement in speech science and represents a significant advance in the quality of speech production data available to engineers and linguists. Like previous databases, the UBDB datasets provide simultaneous audio and articulator movement recordings of spoken speech. It is unique, however, in the extent of the speaker sample and in the richness of the corpus of recorded tasks. Each speaker in the database has roughly 20 minutes of simultaneously recorded data representing read sentences, isolated words, number sequences, paragraphs and a variety of other speech and oral-motor tasks. The database contains recordings for 57 native English speakers (25 men, 32 women) from a variety of geographic regions and backgrounds. During the recording, four synchronized data streams are simultaneously collected:

- a high bandwidth recording of the speech wave using a directional microphone but subjected to the noise of the X-ray machine

- traces of the positions of eight reference points in the midsaggital plane on the speaker's articulators (four on the tongue, one on each lip, one at the front of the jaw and one at the back of the jaw – see below for more details)

- a neck wall vibration (NWV) signal which measures the acceleration of the skin over the thyroid lamina – this signal indicates the movement of the vocal cords and is roughly a voicing indicator

- low grade lateral and frontal video images at 60 frames/second

In addition to these time series, an static estimate of the outline of the *palatal vault* (the front portion of the roof of the mouth known as the *hard palate*) was manually collected for each speaker. This was usually done by making a stone cast of the speaker's palatal arch and placing the cast in the X-ray machine with string of gold beads on its mid-line. (For a few speakers, a single bead was drawn across the arch of the mouth by an experimenter.) Also, the posterior *phyarengyal wall* (back of the vocal tract) was estimated visually by an operator from machine calibration scans. These two static contours are crucial to a proper interpretation of the movement data from the four tongue beads: the airflow which occurs during speech moves **between** the tongue and palate. Full closures of this airway which stop the sound, tight constrictions which cause turbulent sources to develop, and other such common phenomena in speech production can only be understood by looking at the position of the tongue relative to the structure above it.

It turns out that there is an automated way to derive this palate trace from the data itself, which yields a higher resolution palate estimate that is valid over a larger region of space and agrees very well with the manually measured trace. This method will be discussed further in section 3.3.4.

Is is important to remember that all eight articulatory tracking points lie on the midsaggital plane. This means that the measured point positions do not **directly** provide information about the three-dimensional shape of the tongue. However, as Stone and others [184, 185, 186] have shown, the shapes that the tongue typically assumes during speech production are sufficiently constrained that it is often possible to infer something about the full volumetric tongue shape from knowledge only of its mid-line profile.

3.2 UBDB data collection specifics

Below I provide some important details about the various data in the UBDB and how they were collected. Most of this information comes from the database handbook [201] to which the reader is referred for even more technical information.

- The **sound pressure wave** is normally recorded with a sampling period of 46μsec (approximately but not exactly 21739Hz) with 16 bits of nominal amplitude resolution (actually only 14 bit are present but that is still plenty). The audio signal was recorded using a high quality directional microphone (a SHURE SM81 condenser microphone) but unfortunately in the presence of significant room and machine noise. Westbury estimates the environmental noise to be roughly 60dB SPL which puts the signal to noise ratio at no better than 30dB for loud talkers and significantly worse for quiet ones. An anti-aliasing filter (3dB point at 7500Hz) was applied to the microphone signal before digitizing.

- The **neck wall vibration** signal does *not* measure conductance across the vocal cords as with an *electroglottogram* (EGG), but rather the acceleration of the skin over the thyroid lamina. However, its signal does provide a similar indication of the degree of voicing during an utterance. It was normally sampled at 5434Hz (although there is some confusion about whether this is exactly one quarter of the audio sample rate, i.e. 5434.78... or exactly 5434:00...).

- **Two anomalous speakers** (JW7 & JW8) have different sampling rates for the data (16129Hz for the audio and 5376Hz for the NWV). Future data analysts are cautioned to take note of this, especially since the NWV sampling rate is close enough to the normal rate to be easily confused. In addition, a **third irregularity** is that speaker JW9 has a sampling rate of 4831Hzfor the NWV signal.

- **Un-filtered, very high sample rate DAT recordings** of both the raw speech pressure wave microphone signal (audio) and accelerometer (NWV) signal were made using a Sony PCM-2500 digital audio tape recorder and no signal preprocessing, filtering or pre-emphasis at all. The space occupied by these signals, however, prevents them from being distributed with the main database (as is also true for the video images).

In all of the work that follows, we make use of neither the DAT recordings nor the video image data. In addition to the enormous computational burden of working with these huge streams, I did not feel that they would add significant information to my analysis. The sample rates for the audio and neck wall vibration signal are sufficiently high and the filtering sufficiently benign that the original raw signals are not essential. And it is important to keep in mind the DAT records the *same* microphone and accelerometer signals, so it suffers from exactly the same noise as the lower sample rate recordings. The video images, while of high temporal resolution, are of extremely low spatial quality (and to make things worse the lateral and frontal shots were split field onto a single NTSC frame). Westbury himself suggests that "it is unlikely that the images would be of future use to investigators interested, for example, in speech reading" (database handbook, p.30).

3.2.1 Bead placement, coordinate systems, and tracking mechanism

The marker pellets which provide the tracking points for the speech articulators are gold beads $\approx$ 3mm in diameter which are attached by dental adhesives (Ketac and Isodent) to the speaker's articulators, teeth and head. The movements of the pellets within a vertical plane are tracked by an X-ray system described below. All position information lies within a plane whose normal is the line of sight of the X-ray beam. Speakers were seated in the machine so as to align this beam axis to be parallel with the axis between their ears so that the recording plane was approximately the midsaggital plane of the speaker's head (that which bisects the head saggitally). However, the speakers' heads were never restrained during recording. The reference beads described below allow the system to compensate for translations and pitching of the speaker's head within the midsaggital plane. But they provide only a very limited ability to correct for skew distortions outside this plane.

Eleven pellets in total were placed on each speaker, three for reference and eight for recording the movements of speech articulators. The three reference pellets are placed on the speaker's head: two on the bridge of nose and one on the upper incisors. These reference pellets allowed head translation and pitching (within the midsaggital plane and about an axis perpendicular to it) to be removed during data recording. Thus, the remaining eight pellets are tracked relative to a floating coordinate system defined by these three reference beads. The axes for this floating coordinate system are established by having each speaker bite down on a calibrated bite-plate to estimate the plane between the teeth. The intersection of this plane and the plane in which the reference pellets lie becomes the x-axis of all the recorded data (positive direction out of the

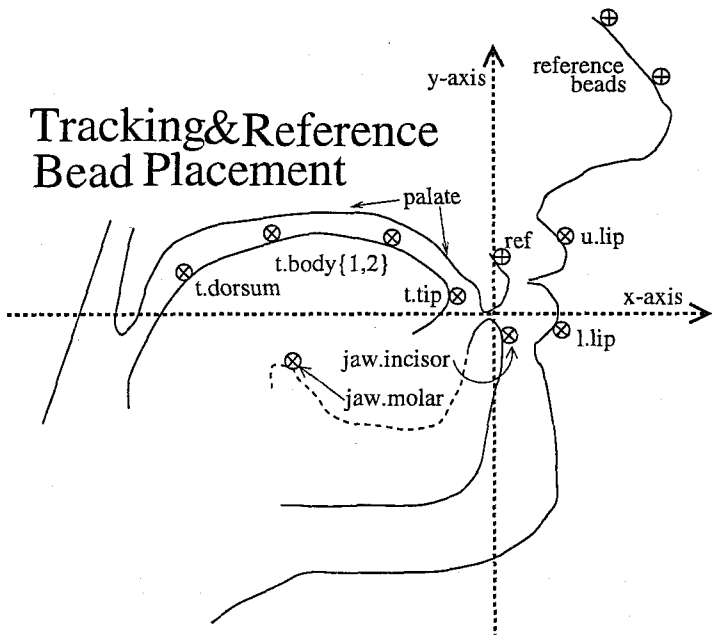


Figure 3.2: Bead placement and coordinate system used to collect the movement data used in the thesis. The eight tracking beads are placed on the tongue (4), lips (2) and jaw (2). Their motions are recorded relative to the three reference beads in a coordinate system defined by the plane between the teeth and the plane of the reference beads. See text for details.

mouth). The y-axis is the perpendicular to this which also lies in the plane of the reference beads (positive direction towards the roof of the mouth). The origin of the coordinate system lies just below (negative y-direction) the reference bead placed on the upper incisors.

The eight tracking beads are allocated as follows. Four beads were placed on the speaker's tongue: one on the tip, one as far back as the speaker could tolerate without gagging, and the other two roughly equally in between. One bead was placed on the lower lip and one on the upper lip. The last two beads were placed on the lower incisor (front of the jaw) and a lower molar (back of the jaw). Figure 3.2 shows an illustration of the bead placement and coordinate system alignment. Figure 3.3 shows an actual photograph of a subject with the tracking and reference beads attached by adhesive and secured with tape.

The beads are tracked by a narrow 1/2mm X-ray microbeam which is produced by shooting a 450kV electron beam at a tungsten target using between 1.5A and 2.0A of current. The angle of the incident electron beam can be controlled so as to direct the resulting X-ray beam to certain parts of the midsaggital plane. The X-ray beam "paints" a 6mm square area where the computer anticipates a particular bead will be; the actual gold bead leaves a shadow on a sodium-iodide (NaI)

Figure 3.3: A subject in the X-ray microbeam study. Eleven gold beads are affixed to the speaker's articulators and head with dental adhesive and secured with tape. Of these, eight are tracked relative to the other three reference beads which correct for the movement of the subject's head within the machine.

crystal detector which indicates its true position within the 6mm search square. The machine then illuminates the search square for the next bead and locates it using its shadow and so forth. In this fashion, up to 12 pellets can be tracked, with the sampling rate of each varying between 20Hz and 180Hz, totaling no more than about 700 "shots" per second. This last limitation is because it is necessary to allow the tungsten target to cool before the next electron beam blast, otherwise it will burn up. This induced X-ray microbeam tracking technique was first developed in Japan by Fujimura, Kiritani & Ishida and their colleagues [69, 70] and pioneered in North America by John Westbury [201, 202]. Figures 3.4 and 3.5 show schematics of the X-ray beam path and of the physical process used to collect the data. Figure 3.6 shows a photograph of the actual machine used to collect the data.

3.2.2 Temporal and spatial resolution

It is possible to program the computer that controls the beam generation to sample some beads more often than others. Because the different articulators have different velocities during the production of normal speech, it is necessary to sample some of them faster than others to correctly capture all motions. For example the tongue tip moves much more quickly than the jaw or lip points. Westbury and his team decided on a schedule of nominal sampling rates for each bead. Unfortunately, the machine does not take each sample in exactly the same amount of time. This means two things: (1) the raw samples are not evenly spaced in time and (2) the nominal sampling rate is not exactly achieved. Table 3.2.2 shows the requested sampling rate and the average

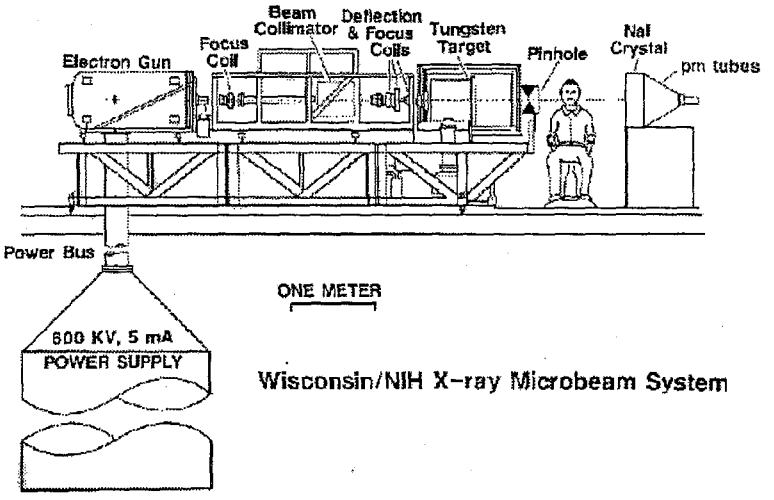


Figure 3.4: A schematic of the X-ray microbeam machine. An electron beam strikes a tungsten target which produces a beam of X-rays. This beam can be carefully focused and directed to strike different parts of the side of a speaker's head. It passes through the head and strikes a NaI detector crystal on the far side. From Abbs et. al.

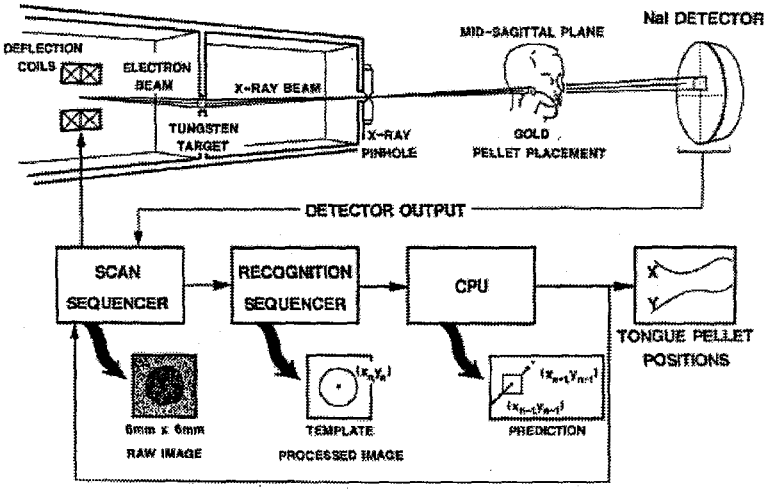


Figure 3.5: The data collection mechanism for the X-ray microbeam facility. Based on a computer prediction of the next position of the tracking beads, the X-ray beam "paints" a search square in the general area. The bead leaves a shadow on the NaI detector which allows its true position to be computed. From Abbs et. al.

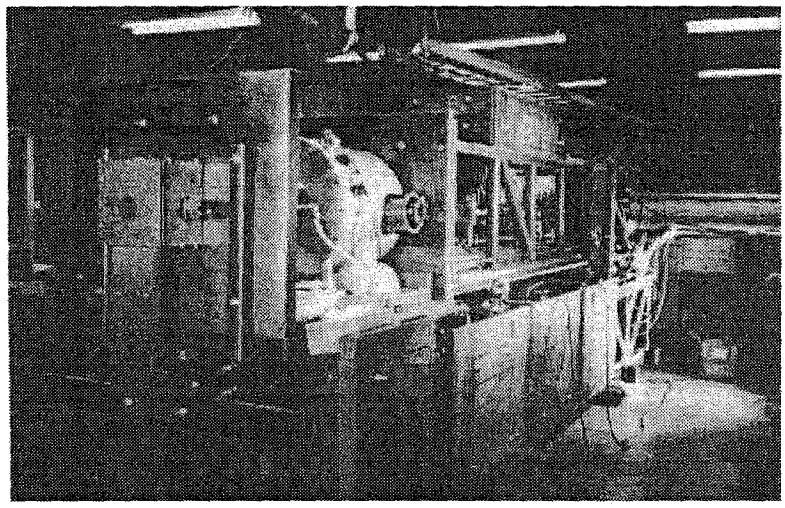


Figure 3.6: The actual machine used in Madison to collect the data in the database over the period December 1990 to October 1991. A typical recording session lasted several hours for each subject and produced roughly 25 minutes of simultaneous movement and audio data. A total of 57 subjects participated in the study.

Bead (number)	Nominal Rate (each bead)	Actual Average Rate
reference (x3)	40Hz	37.6Hz
jaw(x2)	40Hz	37.6Hz
upper lip	40Hz	37.6Hz
lower lip	80Hz	75.2Hz
tongue tip	160Hz	150.4Hz
tongue body(x3)	80Hz	75.2Hz

Table 3.1: Nominal and actual sampling rates for each tracking bead in the X-ray microbeam database.

achieved rate for each bead.

The data recorded by the system was then post-processed to **re-sample every bead uniformly in time and all at the same sampling rate**. For technical reasons this rate was chosen to be approximately 146Hz, the exact sampling period being **6.866 ms**. Thus, the data as it was released consists of 16 numbers (x and y coordinates for each of the eight tracking beads) sampled at equal intervals of 6.866 ms. The time of the first articulatory measurement defines the zero of the time axis for all other data.

The spatial resolution of the microbeam system is difficult to estimate. Each scan locates pellets only to the nearest point on a grid whose spacing is 0.44mm in both the x and y directions. In principal if pellets remain stationary for long periods of time, it would be possible to collect

statistics over the different grid points at which they were located and attempt to use these statistics to achieve sub-pixel accuracy through averaging. In practice, however, the beads are moving quickly enough that they often traverse more than a grid spacing between samples and so the limiting factor becomes the size of the shadow that they cast on the detector as they move. It is possible to estimate this error by calibration runs of the machine or by theoretical estimates. But it is better to have an estimate of the error level that is derived directly from the data themselves; this avoids concerns about the relevance of the theoretical calculations or about differences between the calibration environment and the actual data recording sessions. In section 3.3.4 below I present a simple method for automatically estimating the scale of measurement error directly from the data. It is this estimate which I use throughout the remainder of the document as the scale of the machine error.

3.2.3 Mistracks

A technical limitation of the data capture system is that at certain times a bead may be lost by the X-ray beam. This means that for certain intervals in the database the position information about certain beads is unknown. This condition is called *mistracking* of a pellet. Although the total percentage of time in which beads are mistracked is low (roughly 2%), the proportions of records that involve at least one mistracking is approximately one half. For this reason it is important to deal with the missing information in the database rather than throwing out the affected portions. Chapter 5 presents a method for recovering the lost information based on the positions of other, correctly tracked beads.

3.2.4 Corpus

The speaking tasks in the corpus consisted of reading English words, sentences, paragraphs, and number strings; sequences of vowels, diphthongs, and consonant-vowel-consonant (CVC) syllables; as well as various oral-motor tasks. Details of these tasks are available in the UBDB handbook. For all the analysis in this thesis, only the normal speaking tasks were used. The oral-motor, silent (e.g. swallowing) and diadochokinesis tasks were eliminated. This left 102 of the 118 original tasks comprising over 90% of the original data for each speaker.

The presence of beads is said to not significantly disrupt the speakers' ability to talk except for a brief adaptation period after the beads are initially glued to their mouths. While this thesis

does not investigate this question, the database does provide several audio tracks for each speaker that were recorded before bead attachment. It is thus possible, in principle, to compare the speech audio with and without beads, for example in listening tests.

3.3 Basic characteristics of the data

3.3.1 Some example data

Figures 3.7 and 3.8 show examples of the movement data from the database. The data are displayed both in the conventional x versus t signal plots and in the phase-space of x versus y in which case time is implicit. Also included is an example audio file (sound 3.3.1).

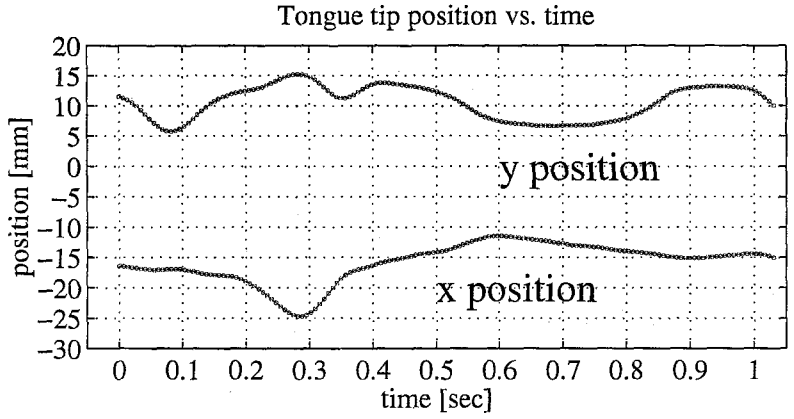


Figure 3.7: Example of a sequence of articulatory data. A conventional plot of the x and y coordinates of a single bead (the tongue tip) versus time.

Audio file 3.3.1 (`track 4`) *An example audio track from the microbeam database.*

3.3.2 What was *actually* said?

A crucial point to keep in mind when working with the UBDB is that there is no record of what is actually being said on the audio tracks. The database does provide, of course, **nominal** word level transcripts of each task. These transcripts are identical for each speaker – they are merely the texts from which subjects were instructed to read. However, even a cursory listening of the data shows that what is actually said often differs from these ideal transcripts. High level errors, such as deleting, adding or repeating entire words or phrases are present; this means that the orthography is different for different speakers. Nor do all speakers choose the same phonetic pronunciation

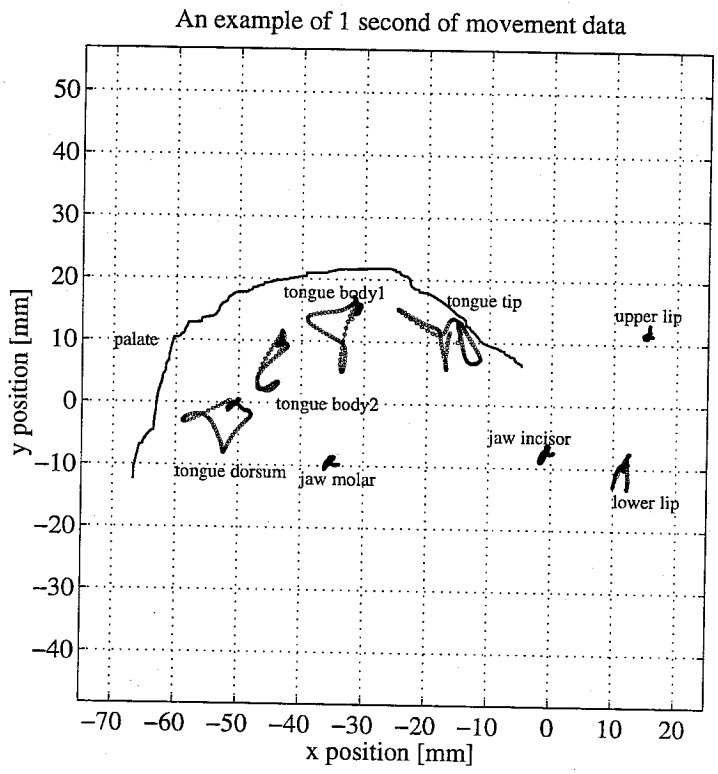


Figure 3.8: An example of one second of articulatory data recordings. The graph shows the bead positions (dots) from 146 articulatory frames with the names of tracking beads beside their positions. A brief sequence of movement data is plotted in x-y phase-space. Time is implicit in the tracks. Also shown is the palate trace (solid line) as computed by an automatic algorithm.

of the words: low level variations in pronunciation and degree of co-articulation are also present. In the end, there exist neither true word-level nor true phone-level transcripts. Furthermore, the timing of the words and phonemes is unknown. The best way to solve these transcription problems is to hand transcribe the data by manual listening; unfortunately this is both extremely tiring and very time consuming. I have done this at the word level for all 118 utterances for two speakers in the database, `jw45` and `jw502`.

The low level phonemic transcription and timing information can often be fairly well estimated **given correct word level orthography** using a technique from automatic speech recognition called *forced alignment.* The idea here is to start with the correct word-level transcript and a dictionary with many pronunciation variants for each word that appears in the transcript. A previously trained hidden Markov model (HMM) system (see Appendix A) can then be used to find the optimum (that is, maximizing likelihood under the pre-trained models) phoneme sequence and switching times between phones. This has been done for all utterances of the two speakers for which hand transcriptions are available.[2] An example of such forced alignment is shown in figure 3.9

3.3.3 Simple statistics of the data

It is useful to compute some rudimentary statistics for each speaker before proceeding with further analysis to establish (a) that the data do not contain severe outliers, encoding errors, etc., and (b) to set maximum and minimum parameter ranges for the various analysis algorithms. The statistics I initially computed were the means and covariances of each bead as well as points that define the convex hull of each bead's data. Notice that these are all computations which are performed independently from bead to bead. Covariances **between** beads and other such global statistics such as global eigenvectors are discussed in chapter 5. Figure 3.10 shows these simple statistics as computed for speaker `jw45`.

3.3.4 Palate and error estimation

Two extremely important pieces of information which are not part of the normal data streams in the UBDB are the location and shape of the speaker's palate and the scale of measurement error of the machine. Although some information about these quantities has been distributed with the

[2]To do this, I have used sophisticated pre-trained cross-word triphone HMMs generously provided by the CLSP at Jons Hopkins University.

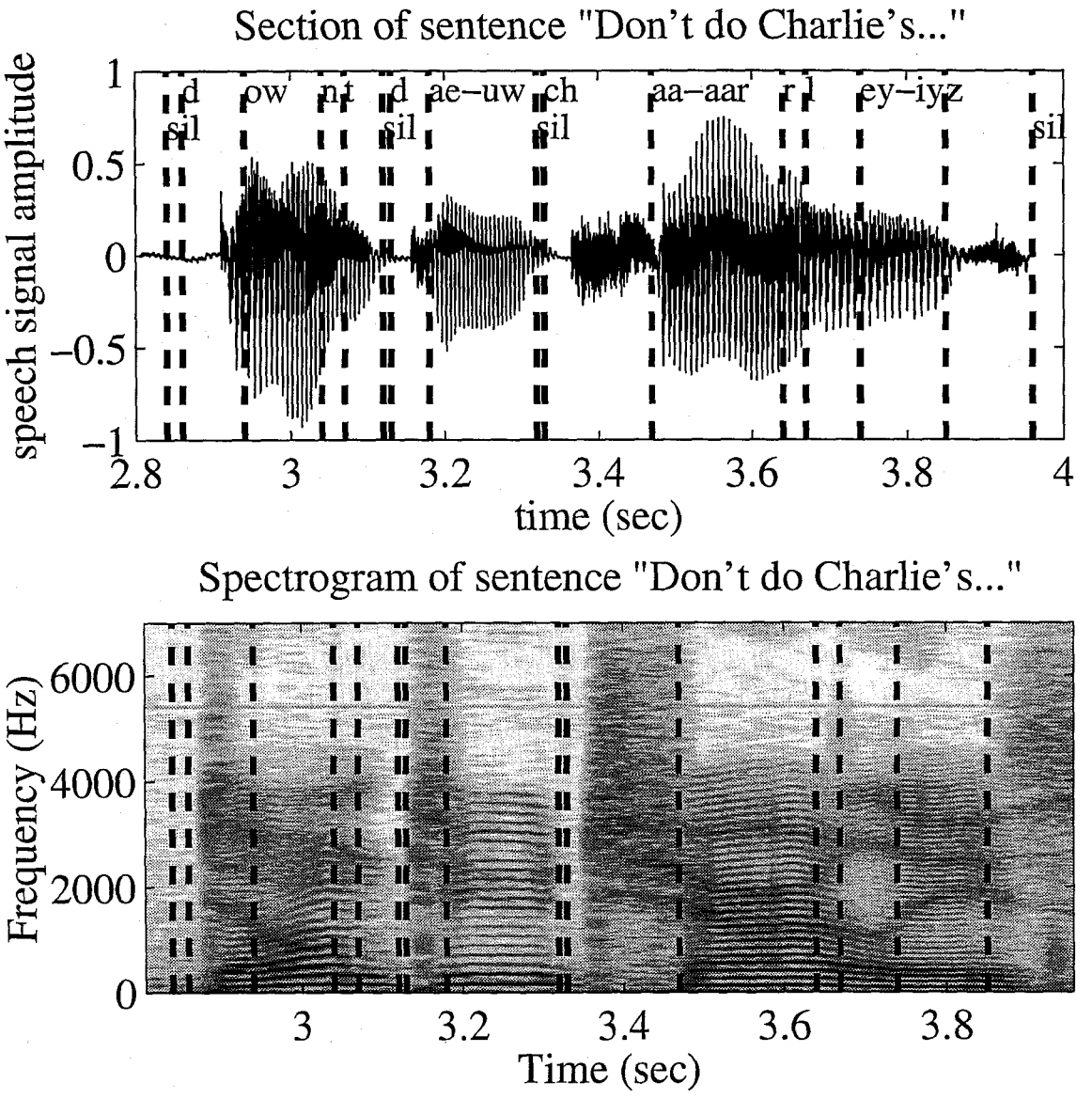


Figure 3.9: An example of forced alignment. A complex hidden Markov model recognition system was pre-trained on a much larger multi-speaker database of transcribed audio. The **correct** word-level transcription of an utterance from the UBDB is then provided along with the audio signal and a dictionary containing multiple possible pronunciations of each word. Pre-trained HMMs then select the best pronunciation variant for each word and also find the optimal switching times from phoneme to phoneme. Shown is the forced alignment of part of utterance (`tp010` for speaker `jw45`) whose true transcription is: *Don't do Charlie's dirty dishes.* The vertical lines indicate the transition times between the phoneme models; printed labels indicate the optimum chosen phoneme. Below the pressure wave is a spectrogram showing the same information in the time-frequency plane.

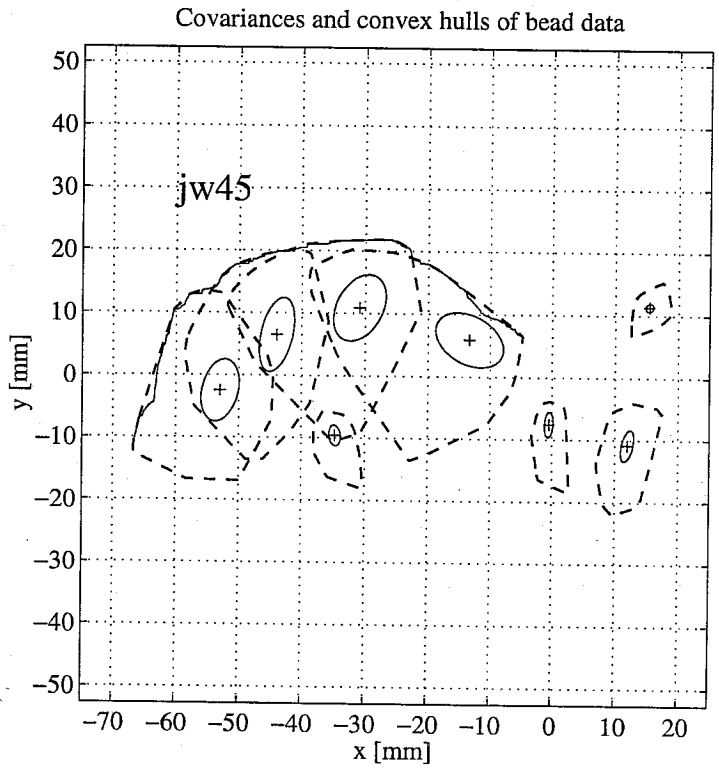


Figure 3.10: Simple statistics for each bead: mean, covariance and convex hull. The crosses indicate the mean positions, the solid lines are the one standard deviation contours of the bead covariances and the dashed lines are the convex hulls. Data for speaker `jw45` is shown.

database, it is somewhat inadequate. In both cases it was possible to improve upon the information provided by directly estimating values from the available data.

Palate outline

In the case of estimating the palate from the data the general idea is to look for vertical extrema of the data. Connecting the uppermost points that lie on the convex hulls of each bead's data provides a satisfactory estimate of the maximum excursions of the tongue beads. But it is possible to do significantly better than this using a simple algorithm I will call the *no-local-minima* smoother. The algorithm requires a single parameter which is a length scale that sets the smoothness of the estimated palate. It then proceeds as follows:

1) Begin by putting the complete database of tongue points for a given speaker into a list.
   Chose the length scale (smoothness distance) to be ℓ.

2) For each measured point (x_0, y_0) on each tongue bead, locate the remaining point in the list with the closest x coordinate to x_0 while still being smaller than x_0. Call this point (x_{left}, y_{left}).
   Similarly locate the remaining point with the closest x coordinate to x_0 while still being greater than x_0. Call this point (x_{right}, y_{right}).

3) If: $x_{right} - \ell \leq x_0 \leq x_{left} + \ell$ and $y_0 \leq y_{left}$ and $y_0 \leq y_{right}$,
   then eliminate (x_0, y_0) from the list and return to step (2).

This algorithm finds a curve that passes through the data points and has no local minima in the vertical direction of horizontal width ℓ or less. It agrees extremely well with the measured palates provided with the database but provides higher spatial resolution and extends further in each horizontal direction. Figure 3.11 shows the result of a palate outline estimated using this algorithm compared to the measured palate points provided with the database.

Spatial measurement errors

In order to estimate the scale of spatial measurement errors in the data, we take advantage of a fortunate physical fact of the bead placements. The two beads placed on the lower jaw are affixed to the same inflexible bony structure. Therefore, regardless of how they move in relation to the rest of the head, the distance between them should be a constant value. We can compute this distance

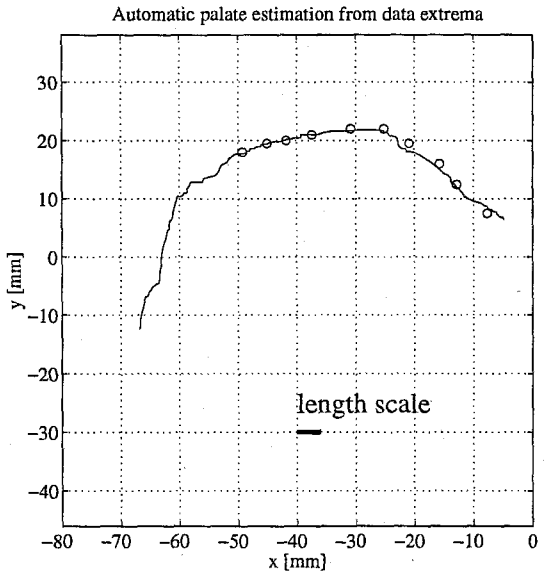


Figure 3.11: Automatically estimated palate outline from bead extrema. The solid line shows the palate outline estimated using the *no-local-minima* smoother. The length scale of the smoother is shown as a solid bar below. The measured points on the palate provided with the database are shown as circles for comparison. Data for speaker `jw45` is shown.

as reported by the measurements in the database for each speaker individually. The degree of spread in the data is taken to be an indication of the measurement error in the machine during that speaker's recording session.[3] Typically this is roughly **0.7mm**; somewhat more than the spatial sampling grid size of 0.44mm quoted in the handbook but certainly not unacceptable. Figure 3.12 shows this nominally constant distance for all the frames in the database for speaker `jw45`, along with the median and percentile levels.

[3]In particular, the median bead error is computed along with the 15.87 and 84.23 percentile levels. These levels are chosen so that between them they contain as much probability mass from the histogram of error values as does a Gaussian distribution within one standard deviation of its mean; thus these statistics are very like the mean and standard deviation of a Gaussian model of the errors. However, the median and percentile measures are much less sensitive to outliers and also have the advantage that they do not depend on monotonic nonlinear transformations. For example, the square root of the median squared error is the same as the median error. I have used half the distance between the upper and lower percentile levels as the estimate of the average bead measurement error.

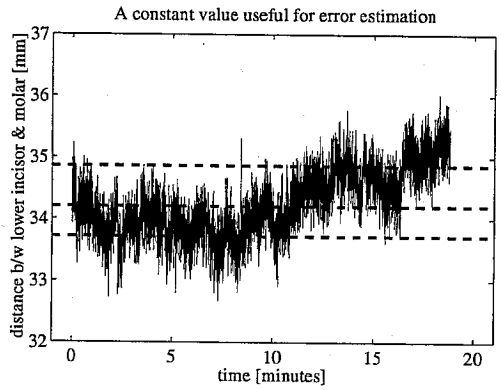


Figure 3.12: A constant value useful for estimating the level of machine measurement error. The two beads placed on the lower jaw are affixed to the same inflexible bony structure. Therefore, regardless of how they move in relation to the rest of the head, the physical distance between is a constant value. This value can be computed from the measured positions of the beads as reported in the database. The spread in this computed value is an indicator of the machine measurement error. The dashed lines show the median, 85 and 15 percentile levels. The error is taken to be half the distance between the upper and lower percentiles.

Chapter 4 EM Algorithms for PCA and SPCA

I present an expectation-maximization (EM) algorithm for principal component analysis (PCA).[1] The algorithm allows a few eigenvectors and eigenvalues to be extracted from large collections of high-dimensional data. It is computationally very efficient in space and time. It also naturally accommodates missing information. I also introduce a new variant of PCA called *sensible* principal component analysis (SPCA) which defines a proper density model in the data space. Learning for SPCA is also done with an EM algorithm. I report results on synthetic and real data showing that these EM algorithms correctly and efficiently find the leading eigenvectors of the covariance of datasets in a few iterations using up to hundreds of thousands of datapoints in thousands of dimensions.

NOTE: This algorithm was originally designed in response to the difficulty of performing the large PCA computations on the speech database and in response to the need for a probabilistic version of PCA as detailed in chapter 5. However, computing power has grown sufficiently since its conception that direct brute force methods for the calculations are now possible and hence are used. The probabilistic version of PCA called SPCA is still employed. I am including the full chapter (which focuses on the computational aspect) here as a record of the way in which the original computations were performed and as a useful algorithm in its own right.

4.1 Why EM for PCA?

Principal component analysis (PCA) is a widely used dimensionality reduction technique in data analysis. Its popularity comes from three important properties. First, it is the *optimal* (in terms of mean squared error) *linear* scheme for compressing a set of high-dimensional vectors into a set of lower-dimensional vectors and then reconstructing. Second, the model parameters can be computed *directly* from the data – for example by diagonalizing the sample covariance. Third, compression and decompression are easy operations to perform given the model parameters –

[1] Material for this chapter was taken from [162]. The comments of three anonymous reviewers and many visitors to my poster at NIPS improved this manuscript greatly. Erik Winfree contributed useful discussions towards the missing data portion of this work. At the time of publication, Chris Bishop and Mike Tipping were pursuing independent but yet unpublished work on a virtually identical model which has now appeared as [191].

they require only matrix multiplications.

Despite these attractive features, however, PCA models have several shortcomings. One is that naive methods for finding the principal component directions have trouble with high-dimensional data or large numbers of datapoints. Consider attempting to diagonalize the sample covariance matrix of n vectors in a space of p dimensions when n and p are several hundred or several thousand. Difficulties can arise both in the form of computational complexity and also data scarcity.[2] Even computing the sample covariance itself is very costly, requiring $O(np^2)$ operations. In general it is best to avoid altogether computing the sample covariance explicitly. Methods such as the *snap-shot* algorithm [176] do this by assuming that the eigenvectors being searched for are linear combinations of the datapoints; their complexity is $O(n^3)$. In this note, I present a version of the expectation-maximization (EM) algorithm [43] for learning the principal components of a dataset. The algorithm does not require computing the sample covariance and has a complexity limited by $O(knp)$ operations where k is the number of leading eigenvectors to be learned.

Another shortcoming of standard approaches to PCA is that it is not obvious how to deal properly with missing data. Most of the methods discussed above cannot accommodate missing values and so incomplete points must either be discarded or completed using a variety of ad-hoc interpolation methods. On the other hand, the EM algorithm for PCA enjoys all the benefits [74] of other EM algorithms in terms of estimating the maximum likelihood values for missing information directly at each iteration.

Finally, the PCA model itself suffers from a critical flaw which is independent of the technique used to compute its parameters: it does not define a proper probability model in the space of inputs. This is because the density is not normalized within the principal subspace. In other words, if we perform PCA on some data and then ask how well *new* data are fit by the model, the only criterion used is the squared distance of the new data from their projections into the principal subspace. A datapoint far away from the training data but nonetheless near the principal subspace will be assigned a high "pseudo-likelihood" or low error. Similarly, it is not possible to generate "fantasy" data from a PCA model. In this note I introduce a new model called *sensible* principal component analysis (SPCA), an obvious modification of PCA, which *does* define a proper covariance structure in the data space. Its parameters can also be learned with an EM algorithm, given below.

[2] On the data scarcity front, we often do not have enough data in high dimensions for the sample covariance to be of full rank and so we must be careful to employ techniques which do not require full rank matrices. On the complexity front, direct diagonalization of a symmetric matrix thousands of rows in size can be extremely costly since this operation is $O(p^3)$ for $p \times p$ inputs. Fortunately, several techniques exist for efficient matrix diagonalization when only the first few leading eigenvectors and eigenvalues are required (for example the power method [204] which is only $O(p^2)$).

In summary, the methods developed in this paper provide three advantages. They allow *simple and efficient* computation of a few eigenvectors and eigenvalues when working with many data-points in high dimensions. They permit this computation even in the presence of missing data. On a real vision problem with missing information, I have computed the 10 leading eigenvectors and eigenvalues of 2^{17} points in 2^{12} dimensions in a few hours using MATLAB on a modest workstation. Finally, through a small variation, these methods allow the computation not only of the principal subspace but of a complete Gaussian probabilistic model which allows one to generate data and compute true likelihoods.

4.2 Whence EM for PCA?

Principal component analysis can be viewed as a limiting case of a particular class of linear-Gaussian models. The goal of such models is to capture the covariance structure of an observed p-dimensional variable $\mathbf{y}$ using fewer than the $p(p+1)/2$ free parameters required in a full covariance matrix. Linear-Gaussian models do this by assuming that $\mathbf{y}$ was produced as a linear transformation of some k-dimensional latent variable $\mathbf{x}$ plus additive Gaussian noise. Denoting the transformation by the $p \times k$ matrix $\mathbf{C}$, and the (p-dimensional) noise by $\mathbf{v}$ (with covariance matrix $\mathbf{R}$), the generative model can be written[3] as

$$\mathbf{y} = \mathbf{C}\mathbf{x} + \mathbf{v} \qquad \mathbf{x} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \quad \mathbf{v} \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \tag{4.1a}$$

The latent or cause variables $\mathbf{x}$ are assumed to be independent and identically distributed according to a unit variance spherical Gaussian. Since $\mathbf{v}$ are also independent and normal distributed (and assumed independent of $\mathbf{x}$), the model reduces to a single Gaussian model for $\mathbf{y}$ which we can write explicitly:

$$\mathbf{y} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{C}\mathbf{C}^T + \mathbf{R}\right) \tag{4.1b}$$

In order to save parameters over the direct covariance representation in p-space, it is necessary to choose $k < p$ and also to restrict the covariance structure of the Gaussian noise $\mathbf{v}$ by constraining

[3] All vectors are column vectors. To denote the transpose of a vector or matrix, I use the notation $\mathbf{x}^T$. The determinant of a matrix is denoted by $|\mathbf{A}|$ and matrix inversion by $\mathbf{A}^{-1}$. The zero matrix is $\mathbf{0}$ and the identity matrix is $\mathbf{I}$. The symbol $\sim$ means "distributed according to." A multivariate normal (Gaussian) distribution with mean μ and covariance matrix $\mathbf{\Sigma}$ is written as $\mathcal{N}(\mu, \mathbf{\Sigma})$. The same Gaussian evaluated at the point $\mathbf{x}$ is denoted $\mathcal{N}(\mu, \mathbf{\Sigma})|_{\mathbf{x}}$.

the matrix $\mathbf{R}$.[4] For example, if the shape of the noise distribution is restricted to be axis aligned (its covariance matrix is diagonal), the model is known as *factor analysis*.

4.2.1 Inference and learning

There are two central problems of interest when working with the linear-Gaussian models described above. The first problem is that of *state inference* or *compression* which asks: *given* fixed model parameters $\mathbf{C}$ and $\mathbf{R}$, what can be said about the unknown hidden states $\mathbf{x}$ given some observations $\mathbf{y}$? Since the datapoints are independent, we are interested in the posterior probability $P(\mathbf{x}|\mathbf{y})$ over a single hidden state given the corresponding single observation. This can be easily computed by linear matrix projection and the resulting density is itself Gaussian:

$$P(\mathbf{x}|\mathbf{y}) = \frac{P(\mathbf{y}|\mathbf{x})\,P(\mathbf{x})}{P(\mathbf{y})} = \frac{\mathcal{N}(\mathbf{C}\mathbf{x}, \mathbf{R})\,|_{\mathbf{y}}\;\mathcal{N}(\mathbf{0}, \mathbf{I})\,|_{\mathbf{x}}}{\mathcal{N}(\mathbf{0}, \mathbf{C}\mathbf{C}^T + \mathbf{R})\,|_{\mathbf{y}}} \tag{4.2a}$$

$$P(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\boldsymbol{\beta}\mathbf{y}, I - \boldsymbol{\beta}\mathbf{C})\,|_{\mathbf{x}}\,, \qquad \boldsymbol{\beta} = \mathbf{C}^T(\mathbf{C}\mathbf{C}^T + \mathbf{R})^{-1} \tag{4.2b}$$

from which we obtain not only the expected value $\boldsymbol{\beta}\mathbf{y}$ of the unknown state but also an estimate of the uncertainty in this value in the form of the covariance $\mathbf{I} - \boldsymbol{\beta}\mathbf{C}$. Computing $\mathbf{y}$ from $\mathbf{x}$ (*reconstruction*) is also straightforward: $P(\mathbf{y}|\mathbf{x}) = \mathcal{N}(\mathbf{C}\mathbf{x}, \mathbf{R})\,|_{\mathbf{y}}$. Finally, computing the likelihood of any datapoint $\mathbf{y}$ is merely an evaluation under (4.1b).

The second problem is that of *learning*, or *parameter fitting*, which consists of identifying the matrices $\mathbf{C}$ and $\mathbf{R}$ that make the model assign the highest likelihood to the observed data. There are a family of EM algorithms to do this for the various cases of restrictions to $\mathbf{R}$ but all follow a similar structure: they use the inference formula (4.2b) above in the **e-step** to estimate the unknown state and then choose $\mathbf{C}$ and the restricted $\mathbf{R}$ in the **m-step** so as to maximize the expected joint likelihood of the estimated $\mathbf{x}$ and the observed $\mathbf{y}$.

4.2.2 Zero noise limit

Principal component analysis is a limiting case of the linear-Gaussian model as the covariance of the noise $\mathbf{v}$ becomes infinitesimally small and equal in all directions. Mathematically, PCA is

[4] This restriction on $\mathbf{R}$ is not merely to save on parameters: the covariance of the observation noise *must* be restricted in some way for the model to capture any interesting or informative projections in the state $\mathbf{x}$. If $\mathbf{R}$ were not restricted, the learning algorithm could simply choose $\mathbf{C} = \mathbf{0}$ and then set $\mathbf{R}$ to be the covariance of the data thus trivially achieving the maximum likelihood model by explaining all of the structure in the data as noise. (Remember that since the model has reduced to a single Gaussian distribution for $\mathbf{y}$, we can do no better than having the covariance of our model equal the sample covariance of our data.)

obtained by taking the limit $\mathbf{R} = \lim_{\epsilon \to 0} \epsilon \mathbf{I}$. This has the effect of making the likelihood of a point $\mathbf{y}$ dominated solely by the squared distance between it and its reconstruction $\mathbf{Cx}$. The directions of the columns of $\mathbf{C}$ which minimize this error are known as the *principal components*. Inference now reduces to[5] simple least squares projection:

$$P(\mathbf{x}|\mathbf{y}) = \mathcal{N}(\boldsymbol{\beta}\mathbf{y}, I - \boldsymbol{\beta}\mathbf{C})\,|_{\mathbf{x}}\,, \qquad \boldsymbol{\beta} = \lim_{\epsilon \to 0} \mathbf{C}^T(\mathbf{C}\mathbf{C}^T + \epsilon\mathbf{I})^{-1} \tag{4.3a}$$

$$P(\mathbf{x}|\mathbf{y}) = \mathcal{N}\left((\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{y}, \mathbf{0}\right)|_{\mathbf{x}} = \delta(\mathbf{x} - (\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{y}) \tag{4.3b}$$

Since the noise has become infinitesimal, the posterior over states collapses to a single point and the covariance becomes zero.

4.3 An EM algorithm for PCA

The key observation of this note is that even though the principal components can be computed explicitly, there is still an EM algorithm for learning them. It can be easily derived as the zero noise limit of the standard algorithms (see for example [73, 54] and section 4 below) by replacing the usual **e-step** with the projection above. The algorithm is:

- **e-step:** $\mathbf{X} = (\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{Y}$
- **m-step:** $\mathbf{C}^{new} = \mathbf{Y}\mathbf{X}^T(\mathbf{X}\mathbf{X}^T)^{-1}$

where $\mathbf{Y}$ is a $p \times n$ matrix of all the observed data and $\mathbf{X}$ is a $k \times n$ matrix of the unknown states. The columns of $\mathbf{C}$ will span the space of the first k principal components. (To compute the corresponding eigenvectors and eigenvalues explicitly, the data can be projected into this k-dimensional subspace and an ordered orthogonal basis for the covariance in the subspace can be constructed.) Notice that the algorithm can be performed *online* using only a single datapoint at a time and so its storage requirements are only $O(kp) + O(k^2)$. (This is done by computing $\mathbf{x}$ as $(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{y}$ individually for each datapoint and accumulating the results in the matrices $\mathbf{X}\mathbf{X}^T$ and $\mathbf{Y}\mathbf{X}^T$.) The workings of the algorithm are illustrated graphically in figure 4.1 below.

The intuition behind the algorithm is as follows: guess an orientation for the principal subspace. *Fix* the guessed subspace and project the data $\mathbf{y}$ into it to give the values of the hidden

[5] Recall that if $\mathbf{C}$ is $p \times k$ with $p > k$ and is rank k then left multiplication by $\mathbf{C}^T(\mathbf{C}\mathbf{C}^T)^{-1}$ (which appears not to be well defined because $(\mathbf{C}\mathbf{C}^T)$ is not invertible) is *exactly equivalent to* left multiplication by $(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T$. The intuition is that even though $\mathbf{C}\mathbf{C}^T$ truly is not invertible, the directions along which it is not invertible are exactly those which $\mathbf{C}^T$ is about to project out.

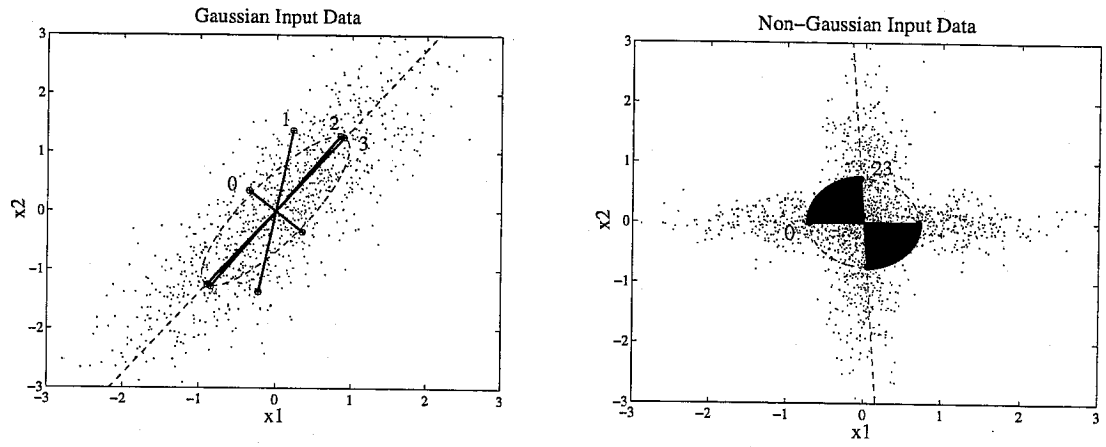


Figure 4.1: Examples of iterations of the algorithm. The left-hand panel shows the learning of the first principal component of data drawn from a Gaussian distribution, while the right-hand panel shows learning on data from a non-Gaussian distribution. The dashed lines indicate the direction of the leading eigenvector of the sample covariance. The dashed ellipse is the one standard deviation contour of the sample covariance. The progress of the algorithm is indicated by the solid lines whose directions indicate the guess of the eigenvector and whose lengths indicate the guess of the eigenvalue at each iteration. The iterations are numbered; number 0 is the initial condition. Notice that the difficult learning on the right does not get stuck in a local minimum, although it does take more than 20 iterations to converge which is unusual for Gaussian data (see figure 2).

states $\mathbf{x}$. Now *fix* the values of the hidden states and choose the subspace orientation that minimizes the squared reconstruction errors of the datapoints. For the simple two-dimensional example above, I can give a physical analogy. Imagine that we have a rod pinned at the origin which is free to rotate. Pick an orientation for the rod. Holding the rod still, project every datapoint onto the rod, and attach each projected point to its original point with a spring. Now release the rod. Repeat. The direction of the rod represents our guess of the principal component of the dataset. The energy stored in the springs is the reconstruction error we are trying to minimize.

4.3.1 Convergence and complexity

The EM learning algorithm for PCA amounts to an iterative procedure for finding the subspace spanned by the k leading eigenvectors without explicit computation of the sample covariance. It is attractive for small k because its complexity is limited by $O(knp)$ per iteration and so depends only linearly on *both* the dimensionality of the data and the number of points. Methods that explicitly compute the sample covariance matrix have complexities limited by $O(np^2)$, while methods like the snap-shot method that form linear combinations of the data must compute and diagonalize a

matrix of all possible inner products between points and thus are limited by $O(n^2p)$ complexity. The complexity scaling of the algorithm compared to these methods is shown in figure 2 below. For each dimensionality, a random covariance matrix $\mathbf{\Sigma}$ was generated[6] and then $10p$ points were drawn from $\mathcal{N}(\mathbf{0}, \mathbf{\Sigma})$. The number of floating point operations required to find the first principal component was recorded using MATLAB's `flops` function. As expected, the EM algorithm scales more favourably in cases where k is small and both p and n are large. If $k \approx p \approx n$ (we want all the eigenvectors) then all methods are $O(p^3)$.

The standard convergence proofs for EM [43] apply to this algorithm as well, so we can be sure that it will always reach a local maximum of likelihood. Furthermore, Tipping and Bishop have shown [189, 190] that the only stable local extremum is the *global maximum* at which the true principal subspace is found; so it converges to the correct result. Another possible concern is that the number of iterations required for convergence may scale with p or n. To investigate this question, I have explicitly computed the leading eigenvector for synthetic datasets (as above, with $n = 10p$) of varying dimension and recorded the number of iterations of the EM algorithm required for the inner product of the eigendirection with the current guess of the algorithm to be 0.999 or greater. Up to 450 dimensions (4500 datapoints), the number of iterations remains roughly constant with a mean of 3.6. The ratios of the first k eigenvalues seem to be the critical parameters controlling the number of iterations until convergence. (For example, in the right-hand panel of figure 4.1 this ratio was 1.0001.)

4.3.2 Missing data

In the complete data setting, the values of the projections or hidden states $\mathbf{x}$ are viewed as the "missing information" for EM. During the **e-step** we compute these values by projecting the observed data into the current subspace. This minimizes the model error given the observed data and the model parameters. However, if some of the input points are missing certain coordinate values, we can easily estimate those values in the same fashion. Instead of estimating only $\mathbf{x}$ as the value that minimizes the squared distance between the point and its reconstruction, we can generalize the **e-step** to:

[6] First, an axis-aligned covariance is created with the p eigenvalues drawn at random from a uniform distribution in some positive range. Then $(p-1)$ points are drawn from a p-dimensional zero mean spherical Gaussian and the axes are aligned in space using these points.

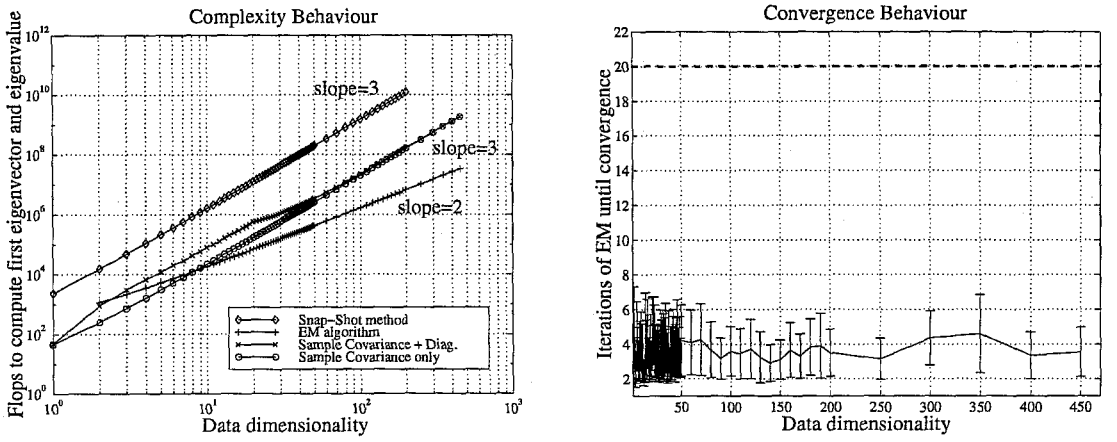


Figure 4.2: Time complexity and convergence behaviour of the algorithm. In all cases, the number of datapoints n is 10 times the dimensionality p. For the left-hand panel, the number of floating point operations to find the leading eigenvector and eigenvalue were recorded. The EM algorithm was always run for exactly 20 iterations. The cost shown for EM includes the explicit orthogonalization into the eigenbasis after the principal subspace is found. The cost shown for diagonalization of the sample covariance uses the MATLAB functions `cov` and `eigs`. The snapshot method is shown to indicate scaling only; one would not normally use it when $n > p$. In the right-hand panel, convergence was investigated by explicitly computing the leading eigenvector and then running the EM algorithm until the dot product of its guess and the true eigendirection was 0.999 or more. The error bars show $\pm$ one standard deviation across many runs. The dashed line shows the number of iterations used to produce the EM algorithm curve ('+') in the left panel.

- **generalized e-step**: For each (possibly incomplete) point $\mathbf{y}$ find the unique pair of points $\mathbf{x}^*$ and $\mathbf{y}^*$ (such that $\mathbf{x}^*$ lies in the current principal subspace and $\mathbf{y}^*$ lies in the subspace defined by the known information about $\mathbf{y}$) which minimize the norm $\|\mathbf{C}\mathbf{x}^* - \mathbf{y}^*\|$. Set the corresponding column of $\mathbf{X}$ to $\mathbf{x}^*$ and the corresponding column of $\mathbf{Y}$ to $\mathbf{y}^*$.

If $\mathbf{y}$ is complete, then $\mathbf{y}^* = \mathbf{y}$ and $\mathbf{x}^*$ is found exactly as before. If not, then $\mathbf{x}^*$ and $\mathbf{y}^*$ are the solution to a least squares problem and can be found by, for example, QR factorization of a particular constraint matrix. Using this generalized **e-step** I have found the leading principal components for datasets in which *every* point is missing some coordinates.

4.4 Sensible principal component analysis

If we require $\mathbf{R}$ to be a multiple $\epsilon\mathbf{I}$ of the identity matrix (in other words the covariance ellipsoid of $\mathbf{v}$ is spherical) but *do not* take the limit as $\epsilon \to 0$, then we have a model which I shall call *sensible principal component analysis* or SPCA. The columns of $\mathbf{C}$ are still known as the *principal components* (it can be shown that they are the same as in regular PCA) and we will call the scalar value ϵ on the diagonal of $\mathbf{R}$ the *global noise level.* Note that SPCA uses $1 + pk - k(k-1)/2$ free parameters to model the covariance. Once again, inference is done with equation (4.2b). Notice, however, that even though the principal components found by SPCA are the same as those for PCA, the mean of the posterior is not in general the same as the point given by the PCA projection (4.3b). Learning for SPCA also uses an EM algorithm (given below).

Because it has a *finite* noise level ϵ, SPCA defines a proper generative model and probability distribution in the data space:

$$\mathbf{y} \sim \mathcal{N}\left(\mathbf{0}, \mathbf{C}\mathbf{C}^T + \epsilon\mathbf{I}\right) \tag{4.4}$$

which makes it possible to generate data from or to evaluate the actual *likelihood* of *new* test data under an SPCA model. Furthermore, this likelihood will be much lower for data far from the training set even if they are near the principal subspace, unlike the reconstruction error reported by a PCA model.

The EM algorithm for learning an SPCA model is:

- **e-step**: $\beta = \mathbf{C}^T(\mathbf{C}\mathbf{C}^T + \epsilon\mathbf{I})^{-1} \qquad \mu_{\mathbf{x}} = \beta\mathbf{Y} \qquad \Sigma_{\mathbf{x}} = n\mathbf{I} - n\beta\mathbf{C} + \mu_{\mathbf{x}}\mu_{\mathbf{x}}^T$
- **m-step**: $\mathbf{C}^{new} = \mathbf{Y}\mu_{\mathbf{x}}^T\Sigma^{-1} \qquad \epsilon^{new} = \text{trace}[\mathbf{X}\mathbf{X}^T - \mathbf{C}\mu_{\mathbf{x}}\mathbf{Y}^T]/n^2$

Two subtle points about complexity[7] are important to notice; they show that learning for SPCA also enjoys a complexity limited by $O(knp)$ and not worse.

4.5 Relationships to previous methods

The EM algorithm for PCA, derived above using probabilistic arguments, is closely related to two well known sets of algorithms. The first are *power iteration* methods for solving matrix eigenvalue problems. Roughly speaking, these methods iteratively update their eigenvector estimates through repeated multiplication by the matrix to be diagonalized. In the case of PCA, explicitly forming the sample covariance and multiplying by it to perform such power iterations would be disastrous. However, since the sample covariance is in fact a sum of outer products of individual vectors, we can multiply by it efficiently without ever computing it. In fact, the EM algorithm is exactly equivalent to performing power iterations for finding $\mathbf{C}$ using this trick. Iterative methods for partial least squares (e.g. the NIPALS algorithm) are doing the same trick for regression. Taking the singular value decomposition (SVD) of the data matrix directly is a related way to find the principal subspace. If Lanczos or Arnoldi methods are used to compute this SVD, the resulting iterations are similar to those of the EM algorithm. Space prohibits detailed discussion of these sophisticated methods, but two excellent general references are the book by Golub and Van Loan and the ARPACK documentation [76, 113]. The second class of methods are the competitive learning methods for finding the principal subspace such as Sanger's and Oja's rules [167, 168, 141]. These methods enjoy the same storage and time complexities as the EM algorithm; however, their update steps reduce but do not minimize the cost and so they typically need more iterations and require a learning rate parameter to be set by hand.

4.6 An example with real image data

The EMPCA algorithm was tested on a large database of natural images.[8] Each grayscale image consisted of a 256×256 array of real values between 0 and 1. Each image was raster-scanned into a long column vector, making the dimensionality p of the data space equal to 2^{16}. Several

[7] First, since $\epsilon\mathbf{I}$ is diagonal, the inversion in the **e-step** can be performed efficiently using the matrix inversion lemma: $(\mathbf{C}\mathbf{C}^T + \epsilon\mathbf{I})^{-1} = (\mathbf{I}/\epsilon - \mathbf{C}(\mathbf{I} + \mathbf{C}^T\mathbf{C}/\epsilon)^{-1}\mathbf{C}^T/\epsilon^2)$. Second, since we are only taking the trace of the matrix in the **m-step**, we do not need to compute the full sample covariance $\mathbf{X}\mathbf{X}^T$ but instead can compute only the variance along each coordinate.

[8] Thanks to Dawei Dong for providing this data.

hundred thousand images were used, making the number n of datapoints roughly 2^{19}. The first 16 eigenvectors and eigenvalues were extracted from the data. Furthermore, to minimize the memory impact of the algorithm, the online version of the algorithm was used: each datapoint was read in from disk and then discarded before the next point was processed. The entire computation took less than four hours of wall clock time on a 133 MHz original Pentium processor running MATLAB. The point of the experiment was not so much to obtain the results as to show that such a computation could be performed using the algorithm in an extremely short time.

The results are shown in the figures 4.3 and 4.4. The mean frame is almost uniform, although expanding its dynamic range shows a "horizon" effect in which the upper portion is lighter than the lower portion indicating that on average, images have more light above than below. The first eigenmode is an almost purely DC mode, although again some structure is observed in the expanded dynamic range image. This structure is believed to be the response function of the CCD that was used to digitize the images. The first 10 eigenmodes are shown and correspond roughly to Fourier modes as is expected.

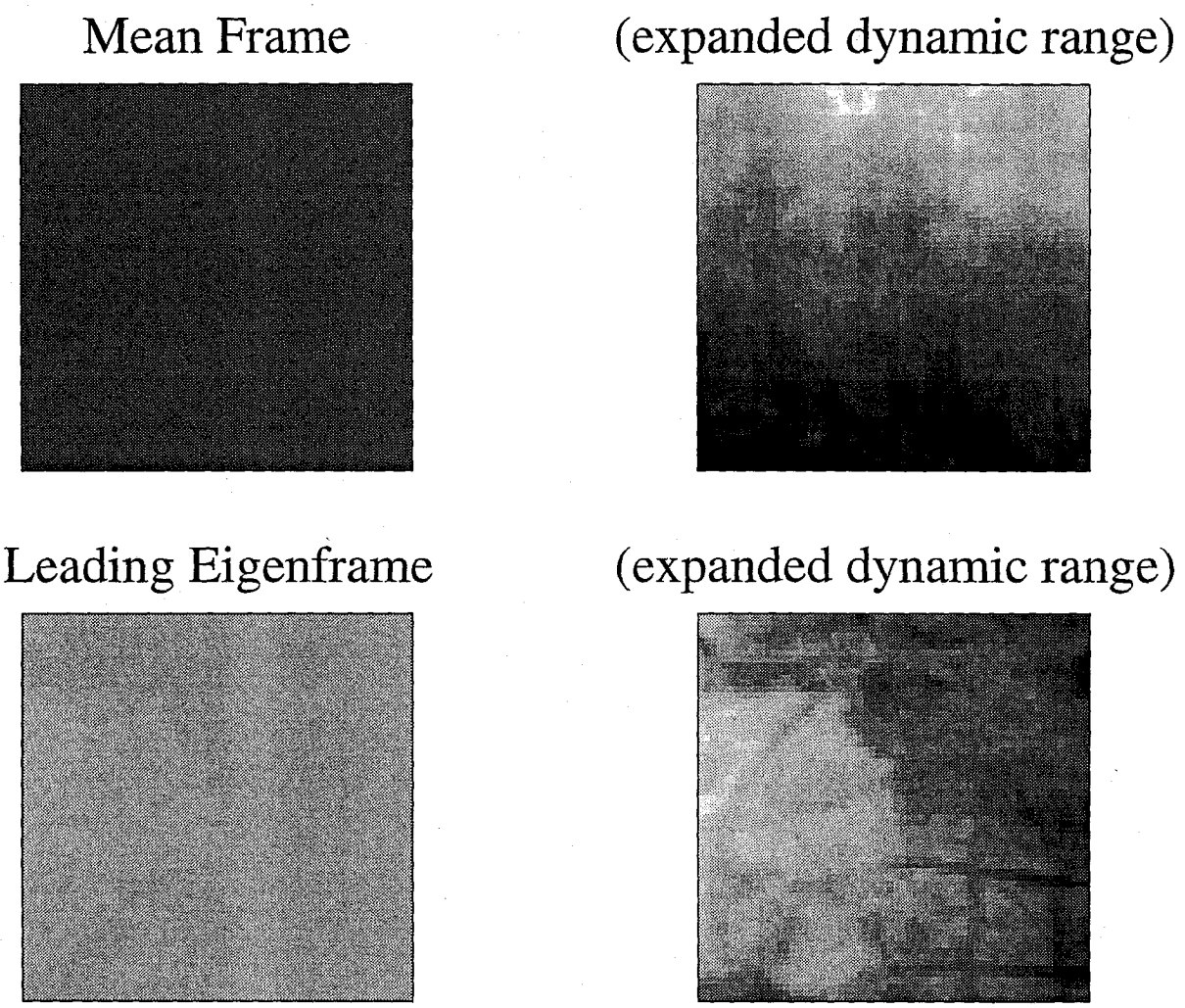


Figure 4.3: Mean frame and leading eigenmode as computed by the EMPCA algorithm for a large database of roughly 2^{19} natural images. Each image contains 2^{16} pixels. The mean frame is almost uniform, although expanding its dynamic range shows a "horizon" effect in which the upper portion is lighter than the lower portion indicating that on average, images have more light above than below. The first eigenmode is an almost purely DC mode, although again some structure is observed in the expanded dynamic range image. This structure is believed to be the response function of the CCD that was used to digitize the images.

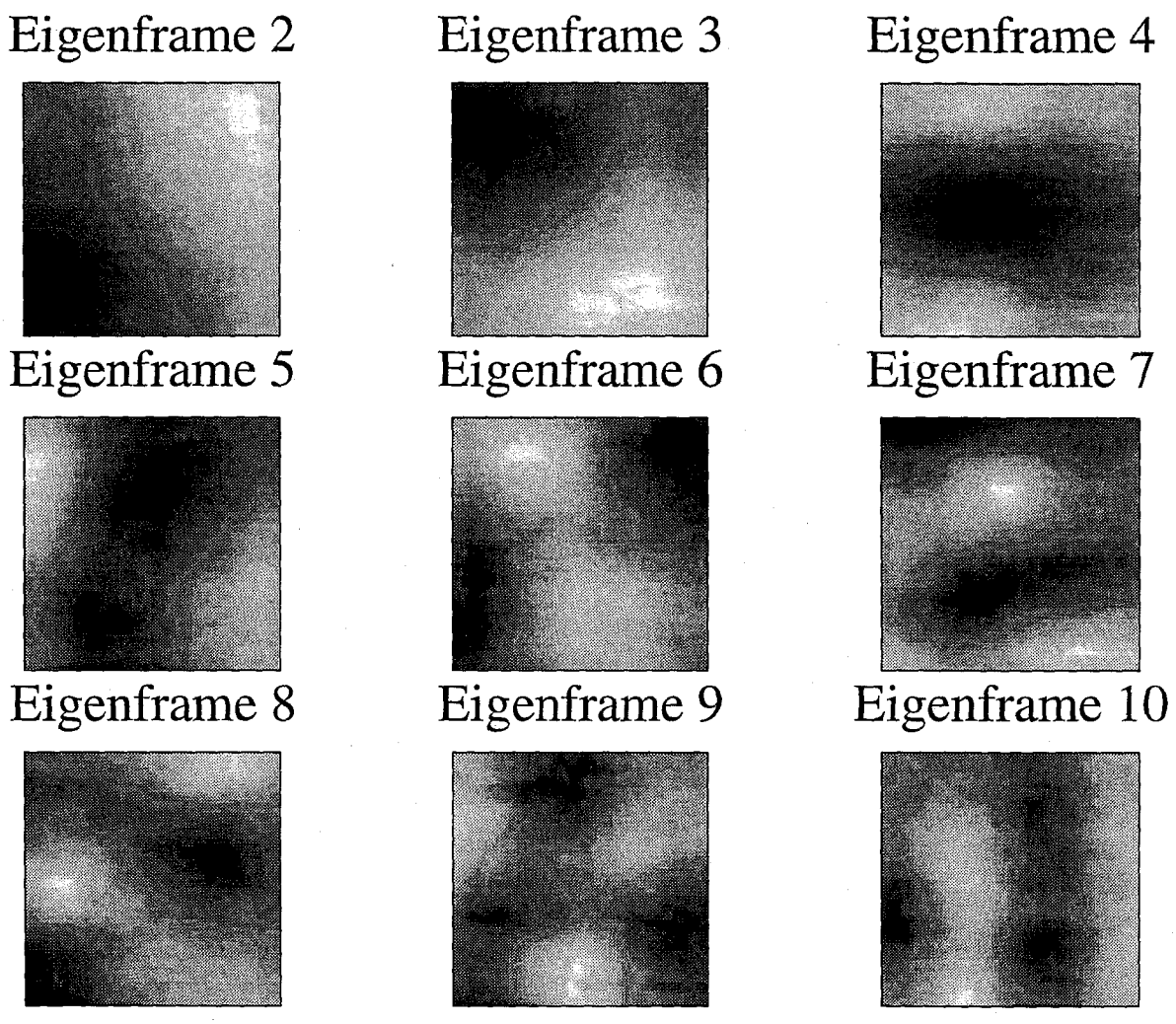


Figure 4.4: Eigenmodes 2 through 8 of the natural images, corresponding roughly to the Fourier modes. Each image contains 2^{16} pixels. Computing the first 16 eigenvectors for a dataset of roughly 2^{19} images took less than four hours in MATLAB.

Chapter 5 Static Models of Articulator Positions

5.1 How constrained are the *positions* of the articulators?

Each of the 57 speakers in the database has approximately 20 minutes of simultaneously recorded audio and movement data. This amount of data contains roughly 175,000 midsaggital "snapshots" (which I will call *frames*) of the articulators. Each of these snapshots corresponds to a point in a 16 dimensional space (the x and y coordinates for each of eight beads) which I will generically call the *bead space* or *articulatory space*. As discussed in chapter 2 several questions can be asked about these data. Some of these questions are static questions, i.e. they do not involve the temporal ordering of the data. Those questions are addressed in this chapter and in the next. The remaining questions are dynamic, i.e. they view each data record as a directed thread or snake in the articulatory space. These are in some sense more exciting questions, and will be addressed in chapters 8 and 9.

In this chapter I address the question: **what part of the articulatory space do the observed positions actually fill**? In other words, **how constrained are the shapes of the mouth**? This question does not involve the audio portion of the database at all, nor does it involve the time ordering of the data points. It is a question about the dimension and shape of the *manifold* in the articulator space on which the observed measurements lie. If this manifold is of significantly lower effective dimensionality than the dimensionality of the original bead space, it means that the articulator configurations *are* highly constrained. (Here "effective dimensionality" means the dimensionality required to achieve a reconstruction error on the order of the measurement error present in the original data—see section 5.1.2 below.)

This manifold simultaneously describes two important sets of constraints about the articulators. First, it excludes any points that represent **physically impossible** configurations. For example, the tongue beads cannot ever be found above the hard palate. The jaw cannot open nor close more than a certain amount. The lips, being attached to the head and jaw, have a very restricted space of movement. And so on. Second, the manifold excludes articulatory configurations that are **not typically used** during the production of normal speech. For example, although humans are physically capable of swallowing, or placing the tip of their tongue under their upper lip or biting

their tongue, these actions are not performed during speaking.

5.1.1 Why estimate the manifold?

There are many reasons to seek reduced dimensionality models of articulatory configurations. First, the effective number of degrees of freedom in the articulators is an intrinsically interesting question from a speech science perspective. It is one that is traditionally answered by proposing various physical/mechanical models of the vocal apparatus. In this work the question is answered using statistical estimation of model parameters on a large dataset. The resulting models have no built-in assumptions about the specific ways in which the articulators can or cannot be configured; instead they assume a certain number of underlying degrees of freedom and learn the rest from the data. But the models so obtained can be used as mechanical models in many of the same ways that physically based ones can. For example, one can specify the locations of a few beads and ask where the remaining beads are likely to fall. Or one can evaluate the probability of a particular configuration in analogy to evaluating the energy stored in a mass and spring model. Forces can also be naturally defined by examining the derivatives of free bead movements with respect to applied constraints. By examining the error or models of various underlying dimension, one can begin to answer this question of effective dimensionality in a more data-driven way (see section 5.1.2 below).

Second, from an engineering perspective there are several benefits. Reduced dimensionality in models lessens the "curse" of exponential explosion of numbers of parameters and computational effort. Furthermore, discovering a representation of data that is less redundant or more uncorrelated often makes probabilistic modeling and pattern recognition easier to do. Finally, as discussed in section 5.4, discovering the manifold on which the articulatory configurations lie allows outlier detection as well as reconstruction from partial information. In terms of the UBDB, these translate to de-noising and filling in those portions of the data which are missing due to mistracking of the X-ray machine.

The results below suggest that **an underlying dimensionality of between four and six is sufficient to capture all the observed variations in articulator configurations**. In other words, the articulator positions are highly constrained in a way that is straightforward to deduce from the data. The manifold on which they lie can be characterized extremely well with simple models such as global principal component analysis.

5.1.2 A data reconstruction approach

The approach I will take to identifying the set of possible articulatory configurations is one of **data reconstruction**. Any reduced degrees-of-freedom model of data implicitly defines a manifold in the original space. The model's representation of points is constrained to lie within this manifold. In the case of the UBDB measurements, the original bead positions can be projected onto a low-dimensional manifold represented by a statistical model and then reconstructed from that manifold back into the original bead space. The reconstructed locations of the beads can be compared to their original true positions to give a positional error. This reconstruction error can be compared to the scale of the natural variability in the data or to the scale of the measurement error of the X-ray machine. If it is acceptably low in a high enough fraction of cases, one can claim that the model for the data is "good" in the sense that it almost always captures the structure of the data without sacrificing its precision. It is important to notice that explicit compression of the data in the sense of minimizing the total bits to represent each data point is not the goal per se. Rather the goal is to find reduced dimensionality models that still capture the essential regularities of the original data.

The error statistics reported in the investigations below (see figures 5.5 and 5.8) are computed using median and percentile errors for each bead. In particular, the median bead error is computed along with the 15.87 and 84.23 percentile error levels. These levels are chosen so that between them they contain as much probability mass from the histogram of error values as does a Gaussian distribution within one standard deviation of its mean; thus these statistics are very like the mean and standard deviation of a Gaussian model of the errors. However, the median and percentile measures are much less sensitive to outliers and also have the advantage that they do not depend on monotonic nonlinear transformations. For example, the square root of the median squared error is the same as the median error.

5.2 A quick word about acoustics

This chapter quantifies the constraints on the articulator positions observed in the X-ray microbeam database. However, the acoustic stream is also constrained. Human speech does not contain much energy above roughly 20kHz and it typically has a power-law spectrum over a wide range of power. (This is mostly true above about 1kHz; below this the spectra are typically almost linear with frequency). The amplitude histogram is also exponential, meaning that large pressure amplitudes are exponentially less likely than small ones.

Finally, the short-time spectra are smooth across frequency: the power in neighbouring frequencies is similar. This is due to the fact that speech is produced by excitation of the physical tube system created by the vocal tract and mouth. The nature of this system does not permit highly tuned resonances which might cause abrupt changes in power across frequencies. (Although notice that the harmonic pitch stack during voiced speech does lay down "spikes" on top of the smooth base spectrum.) These features of the signal have led to the assumption that speech spectra can to a good approximation be modeled by all-pole resonators of low order. Several detailed studies on the structure of natural sounds provide more detailed examinations of these statistics (see for example [7, 17]).

5.3 Global linear models

The simplest possible class of models one can consider for reducing the number of degrees of freedom in the dataset are linear models. Such models are characterized by a memoryless projection matrix of rank less than the dimensionality of the input data. Each data-point $\mathbf{y}$ (a column vector) is left multiplied by the projection matrix $\mathbf{D}$ to obtain its reduced representation $\mathbf{x}$:

$$\mathbf{x} = \mathbf{D}\mathbf{y} \tag{5.1}$$

where $\mathbf{D}$ is a matrix having fewer rows than columns and thus of reduced rank. Recovery of the original data, to the extent possible, is optimally performed by a similar matrix multiplication using a reconstruction matrix $\mathbf{C}$:

$$\hat{\mathbf{y}} = \mathbf{C}\mathbf{x} \tag{5.2}$$

where I have used the notation $\hat{\mathbf{y}}$ to denote a reconstructed data point. (In what follows I will use the symbol $\mathbf{y}$ to indicate a vector representing a point in articulatory space. The dimension of $\mathbf{y}$ will be generically denoted with p although for the data considered in this chapter $p = 16$ always. The vector $\mathbf{x}$ will denote a reduced dimensionality representation of $\mathbf{y}$, and $\hat{\mathbf{y}}$ the reconstructed or recovered version of and original vector $\mathbf{y}$. The dimensionality of $\mathbf{x}$ is k. All vectors are column vectors.)

It turns out that for any such pair of compression and reconstruction matrices $\mathbf{D}$ and $\mathbf{C}$ there

is an exactly equivalent $k \times p$ orthonormal matrix[1] $\mathbf{G}$ such that:

$$\mathbf{x} = \mathbf{G}\mathbf{y} \qquad \text{and} \tag{5.3}$$

$$\hat{\mathbf{y}} = \mathbf{G}^T\mathbf{x} \tag{5.4}$$

has exactly the same effect as using the original $\mathbf{C}$ and $\mathbf{D}$. This means that when considering linear dimensionality reduction, we really only need to think about how to pick the k columns of $\mathbf{G}$, each of which is a unit p-vector. In other words, we are choosing a set of axes (in the original space) that span the subspace into which we want to project the data.

Many extremely rudimentary dimensionality reduction techniques such as discarding some of the original dimensions correspond to specific reduced rank linear models. However, the construction of the *optimal* reduced rank linear model is well known and is called[2] alternately *principal component analysis* (PCA) in statistics and machine learning and the *Karhunen-Loève transform* (KLT) in engineering and signal processing [93, 106, 124]. In what follows, I will use the term PCA. The model definition is easy: take the columns of $\mathbf{G}$ to be the k leading eigenvectors of the sample covariance matrix (not correlation matrix so the mean must be removed beforehand) of the data:

$$\mathbf{C}_{yy} = \langle (\mathbf{y} - \bar{\mathbf{y}})(\mathbf{y} - \bar{\mathbf{y}})^T \rangle \tag{5.5}$$

$$\mathbf{C}_{yy} = \mathbf{V}^T\mathbf{D}\mathbf{V} \qquad \text{(diagonalize covariance)} \tag{5.6}$$

$$\mathbf{V} = [\mathbf{v}_1\, \mathbf{v}_2\, \ldots \mathbf{v}_k\, \ldots \mathbf{v}_p] \tag{5.7}$$

$$\mathbf{G}_{PCA} = [\mathbf{v}_1\, \mathbf{v}_2\, \ldots \mathbf{v}_k] \tag{5.8}$$

This choice of $\mathbf{G}$ minimize the expected squared reconstruction error between the original data $\mathbf{y}$ and the PCA estimate $\hat{\mathbf{y}} = \mathbf{G}^T\mathbf{G}\mathbf{y}$ (see for example [20] for a proof). What PCA does is very simple: it finds the global linear subspace that best fits the data. In this context, "best fits" is defined as follows: if each point is projected onto the subspace and the squared errors from the original data points to their projections are totalled, the subspace found by PCA minimizes this total error. Notice that in general this subspace is distinct from the subspace that is found by doing

[1] An orthonormal matrix, more often called an orthogonal matrix, is one whose columns are mutually orthogonal (have dot product zero with each other) and are also unit vectors (have norm unity or dot product unity with themselves).

[2] Hotelling (1933) was the first to derive and publish this transformation for discrete data under the name "principal components." The corresponding continuous transformation was developed independently by Karhunen (1947) and Loève (1948).

linear regression from some coordinates onto others. A physical analogy is that PCA attaches a spring from each data point to its perpendicular projection on the subspace and sets the orientation of the hyperplane to minimize the total energy in the springs. Regression attaches a spring from each data point to its "vertical" projection onto the regression plane, where "vertical" is taken to mean the direction of the dependent variable. Figure 5.1 graphically illustrates these three types of linear subspace fitting for two variable systems.

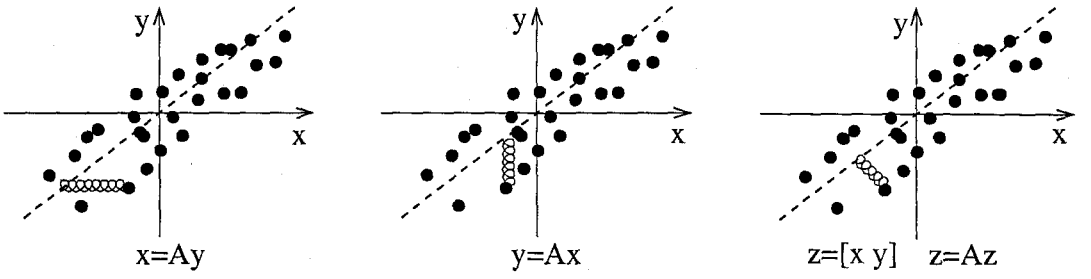


Figure 5.1: Three different ways to fit a straight line to two-dimensional data: regression of x onto y, regression of y onto x and PCA. The left picture shows physical system whose energy is minimized by fitting x as a linear function of y using least squares. The middle shows the system whose energy is minimized by least squares fitting of y as a function of x. The right picture shows the physical systems whose energy is minimized by principal component analysis.

PCA is a method based only on first- and second-order moments of the data. While this makes it relatively easy to compute, it also makes it less powerful than more sophisticated models. One intuitive way to think about this second-order restriction is by imagining fitting a single Gaussian distribution to the dataset, regardless of whether or not the data are actually Gaussian at all. The best Gaussian fit is one which matches the first and second sample moments of the data to the mean and covariance of the fit. The k longest principal axes of this Gaussian fit give exactly the directions of the first k principal components of the dataset. The variances of the true data along these k directions give the eigenvalues. For "blob-like" data, this model works well, but if the data have a more complex structure or shape in the space then linear subspace methods such as PCA do very poorly. One possible extension is to fit a **mixture** of linear subspace models to the data in order to achieve more flexibility. This is similar to the idea of fitting piecewise linear functions in supervised learning. A more complex model in this spirit, which fits a mixture of PCA subspaces is discussed below in section 5.3.2.

5.3.1 Global eigenvectors for articulatory data

What do the global eigenvectors of the articulatory data as computed by PCA look like? How well do they reconstruct the data as a function of the dimensionality of the subspace? Below I present the results of computing the first few leading eigenvectors of the dataset composed of all articulatory points for a single speaker.

ASIDE: Conceptually this eigenvector computation is done by forming a large matrix $\mathbf{Y}$ with $p = 16$ rows and several hundred thousand columns, each of which is an articulatory frame taken at some instant during an utterance recorded by a speaker in the database. The sample covariance of $\mathbf{Y}$ is then computed and diagonalized to give the principal components. In practice, however, computing the sample covariance can be computationally demanding. The resources originally available did not permit direct calculation of the sample covariance and so another method had to be developed to find the first few leading principal components. This led to the development of an EM algorithm for PCA which is described in detail in chapter 4. At first this EM algorithm was used to compute the eigenvectors, but as larger memory and faster computers became available, the direct brute force approach became possible and that is what is currently employed.

The principal components (eigenvectors) themselves are 16-dimensional vectors, and as such displaying them requires some care. The technique I have chosen is to plot each eigenvector as a set of eight two-dimensional vectors centred on the respective bead means. This can be thought of as follows. Start at the mean of the data in 16-dimensional articulatory space. This corresponds to a two-dimensional position for each of the eight beads. Now take a small 16-dimensional step in the direction of the eigenvector. This corresponds to a small two-dimensional step for each bead away from its mean position. Take another step along the eigenvector and compute the corresponding step for each bead, and so on. The lines formed in this fashion for each bead can be used to graphically represent a single 16-dimensional eigenvector. In order to complete the information, the viewer of such a display needs to know in which directions the beads move relative to each other. This is indicated by placing a symbol (triangle or square) at one end of each two-dimensional bead line to show the coordinated polarities that make up the 16-dimensional vector. This is illustrated in figure 5.2 by plotting the leading global eigenvector for speaker `jw45`. The second and third eigenvectors for this same speakers are shown in figure 5.3.

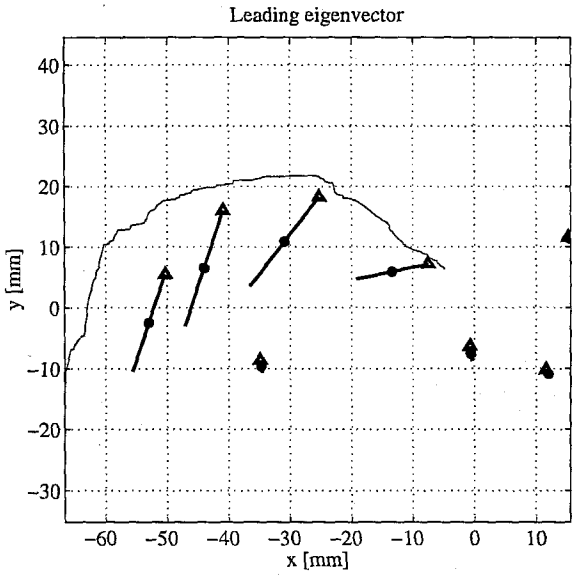


Figure 5.2: An illustration of the leading global eigenvector of bead positions. The eight lines on the plot above together with their polarities as indicated by the symbols at one end of each line represent a single 16-dimensional eigenvector in articulatory space. (The mean position is indicated by solid circles.) Eigenvectors are computed by diagonalizing the sample covariance of all the bead data for a single speaker. Shown is the leading eigenvector (the mode with the greatest power) for speaker `jw45`.

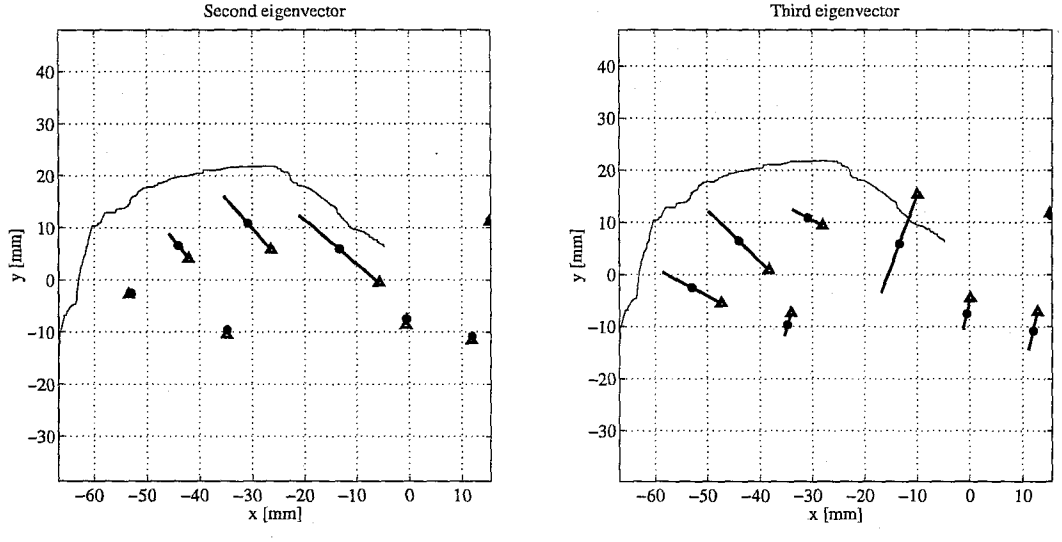


Figure 5.3: The second and third eigenmodes for the same speaker. See figure 5.2 for a description of the display convention.

Associated with each eigenvector is an eigenvalue which indicates the amount of power or total variance from the original data which is carried by that mode. If the data does in fact approximately lie on a low-dimensional hyperplane in the original space, then we would expect the sequence of ranked eigenvalues (called the eigenvalue spectrum of the data) to be rapidly decreasing. This is indeed observed for the articulatory data, as shown in the left panel of figure 5.4. The right panel shows a cumulative sum of the eigenspectrum which indicates what fraction of the total variance in the data is captured by projecting onto a certain number of eigenmodes, or equivalently what the average reconstruction error is, using only a few eigenvectors.

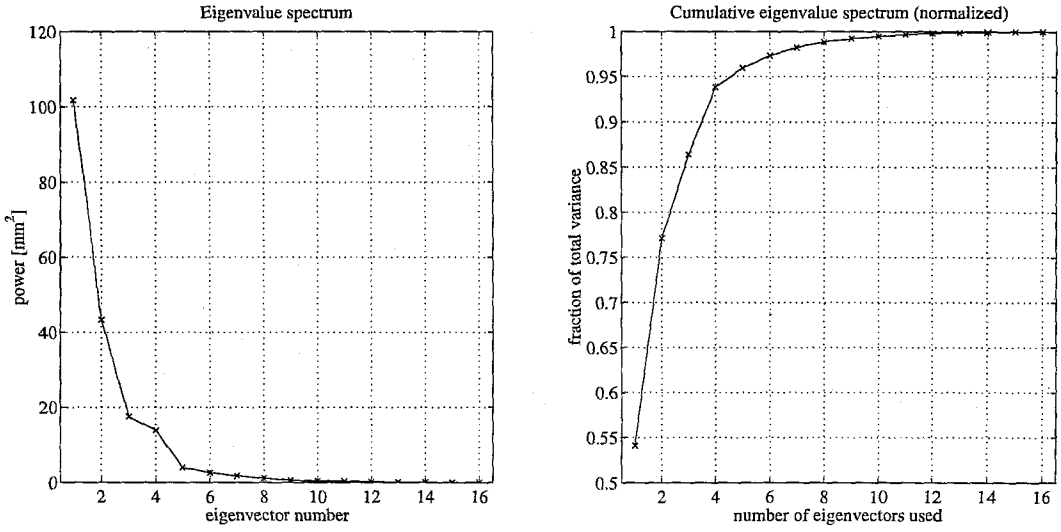


Figure 5.4: Eigenvalue spectrum for speaker `jw45`. In the left panel the ranked eigenvalues are plotted. On the right is plotted the cumulative sum of the first k eigenvalues divided by the sum of all eigenvalues. The right plot indicates the fraction of the total variance "captured" by projecting onto a certain number of eigenmodes; or equivalently the squared reconstruction error that would result from eliminating the remaining modes.

This error information tells us only the *average* reconstruction error across all beads using a certain number of eigenvectors. The errors, however, are different for each bead. There is also a spread in the reconstruction error of each bead which can be computed. To describe these differences, I plot the mean error of each bead separately with error bars in figure 5.5. To compute the error bars, I use the median and percentile errors as described above. The results are shown in figure 5.5. The dashed horizontal line across each panel indicates the estimate of the machine error computed as described in section 3.3.4. The point indicated at zero eigenvectors represents the error reconstructing with the mean alone.

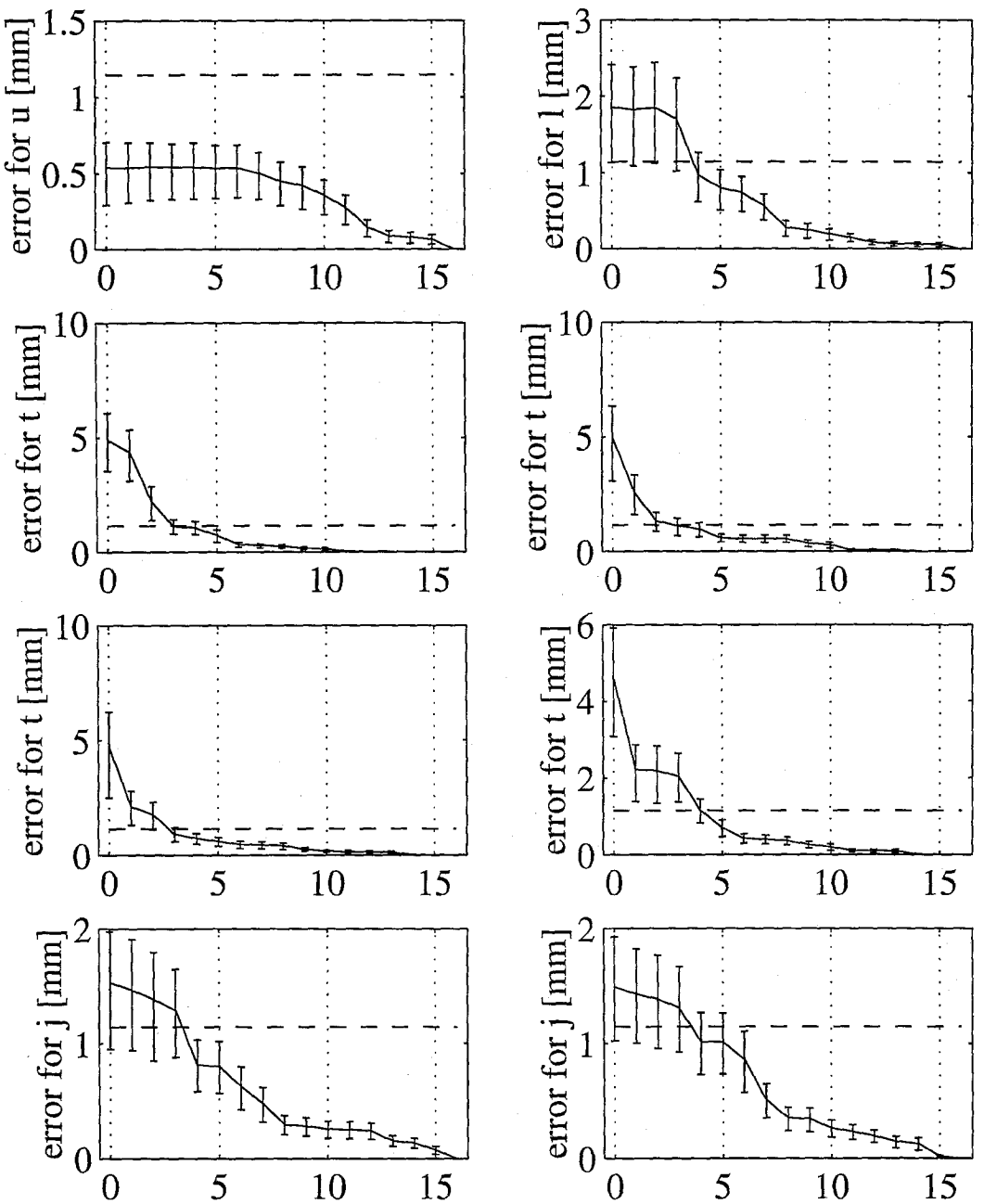


Figure 5.5: Individual bead errors using reconstruction by global eigenvectors. Plotted are the median errors and the 85 and 15 percentile limits. The dashed horizontal line across each panel indicates the estimate of the machine error computed as described in section 3.3.4. The point indicated at zero on the horizontal axes represents the error reconstructing with the mean vector alone. Compare with figure 5.8.

5.3.2 Mixtures of linear models

A common approach in function approximation or supervised learning is to use *piecewise linear models*. The idea is to model the input-output behaviour of the data with a mixture of simple models, each of which applies in only one region of the input space. In unsupervised learning this approach is also quite common and is known as *mixture modeling*. For example, k-means clustering is just a special case of mixture-of-Gaussian modeling. Of particular interest here, it is also possible to model a collection of data using a mixture of linear PCA models. Conceptually, some procedure is used to cluster the points in the dataset into distinct groups (regions). Then PCA is performed within each group (about the local mean vector of the group) using the local covariance matrix. In order to compress a new point using the model, the point is first assigned to one of the sub-models using the same rule as was used at the clustering stage of learning. Then the chosen sub-model's mean vector is subtracted from the new datapoint and its eigenbasis is used to compress the coordinates.

Algorithms for local PCA differ in how they perform the clustering. Two main techniques have been employed. Hinton et al. have used a k-means type training procedure in which points are assigned to the closest subspace and then each subspace is re-estimated based on the points currently assigned to it [83]. Subsequently, Ghahramani and Hinton [73] extended this approach to a full EM style algorithm and also to factor analysis which is a slightly different subspace model than PCA (see Appendix B). Another approach is to cluster the points independently of the subspace estimation and then estimate the subspaces only once. One popular choice for this initial clustering is to use vector quantization (VQ). The hybrid algorithm so obtained has been called VQPCA by Kambhatla and Leen [105]. This is the algorithm I have used to estimate local PCA models. First the k-means training procedure is used to cluster the dataset into disjoint subsets (I have typically used 32 regions). Then PCA is performed within each region. Figure 5.6 shows this algorithm applied to a simple synthetic dataset of annular data.

The VQPCA algorithm can easily be applied to the articulator position data. In this case the dimensionality of each datapoint is 16 rather than 2, but the algorithm remains unchanged. First, VQ is performed to cluster the aritculatory data. In this work I have used 32 clusters (codebook centres), a value which was chosen by leave-one-out cross validation. Then PCA is performed on all points assigned to the same cluster to compute the local models. The resulting models perform slightly better than their global counterparts at the same dimension as is shown in figure 5.8 below.

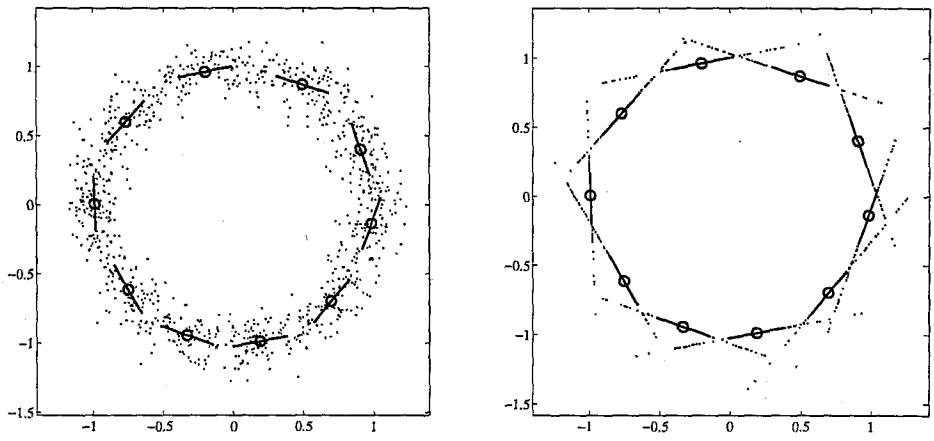


Figure 5.6: An illustration of local PCA models. Each model is locally linear and is responsible for capturing the subspace of the data nearer to its centre than to any other model. The left panel shows the original data and the trained model using 10 sub-models. The right panel shows the same dataset represented using the best local model. Notice that in some cases a data point is represented using a model whose centre is not closer to it than all other centres, but whose subspace passes closer than any other local subspace.

It is also possible to "view" the resulting models by making plots of the mean and covariance of each sub-model. Figure 5.7 shows models in this graphical form for the 32 sub-models trained on the data for speaker `jw45`.

5.3.3 Nonlinear models

The results of the above section indicate that the unsupervised learning problem of capturing the manifold of articulator configurations is not an extremely difficult one. Even the most basic approach of a global linear model does quite well with only 4–6 dimensions. A mixture of a few locally linear models does even better at the cost of only a few extra bits per data point (the log of the number of local models). However "better" models (in the sense that they reconstruct the data better with the same or smaller dimensionality) could certainly be developed using nonlinear techniques.

In this direction, I have tried several approaches. One was to use generalized linear models. In this approach one adds nonlinear functions of data as new basis vectors and then identifies the optimal linear subspace in this new, bigger space. For example in second order polynomial modeling one adds all the squares and pairwise product terms of the original coordinates. In radial basis function fitting, one adds potential functions that have localized activation to certain

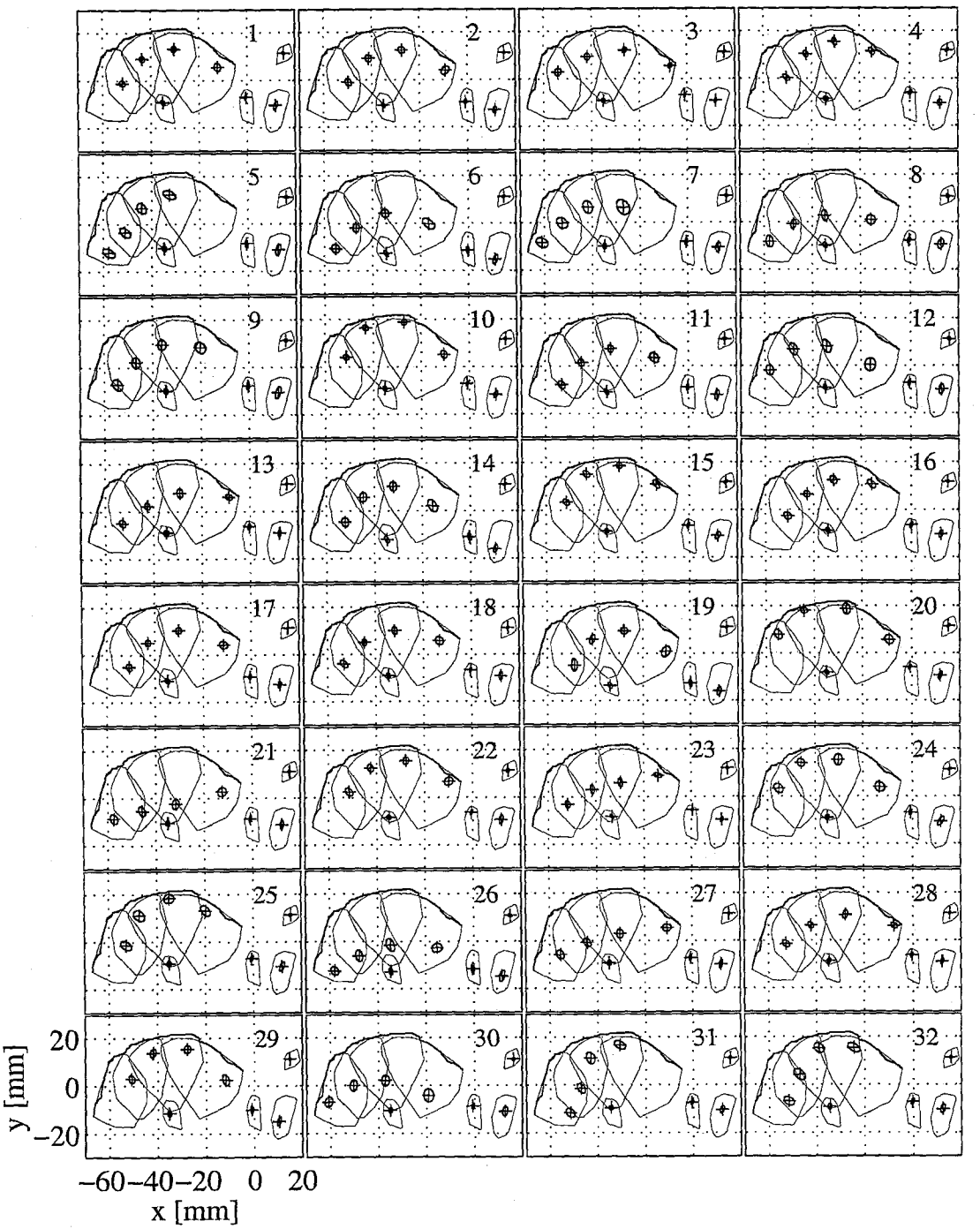


Figure 5.7: Local PCA models trained on articulator positions. Each panel in the figure above shows one of the 32 sub-models trained using the VQPCA algorithm described in the text. The crosses show the mean of the local model while the thick solid lines show the local covariances of each bead. The thin lines show the global convex hull of each bead. The models are for speaker `jw45`.

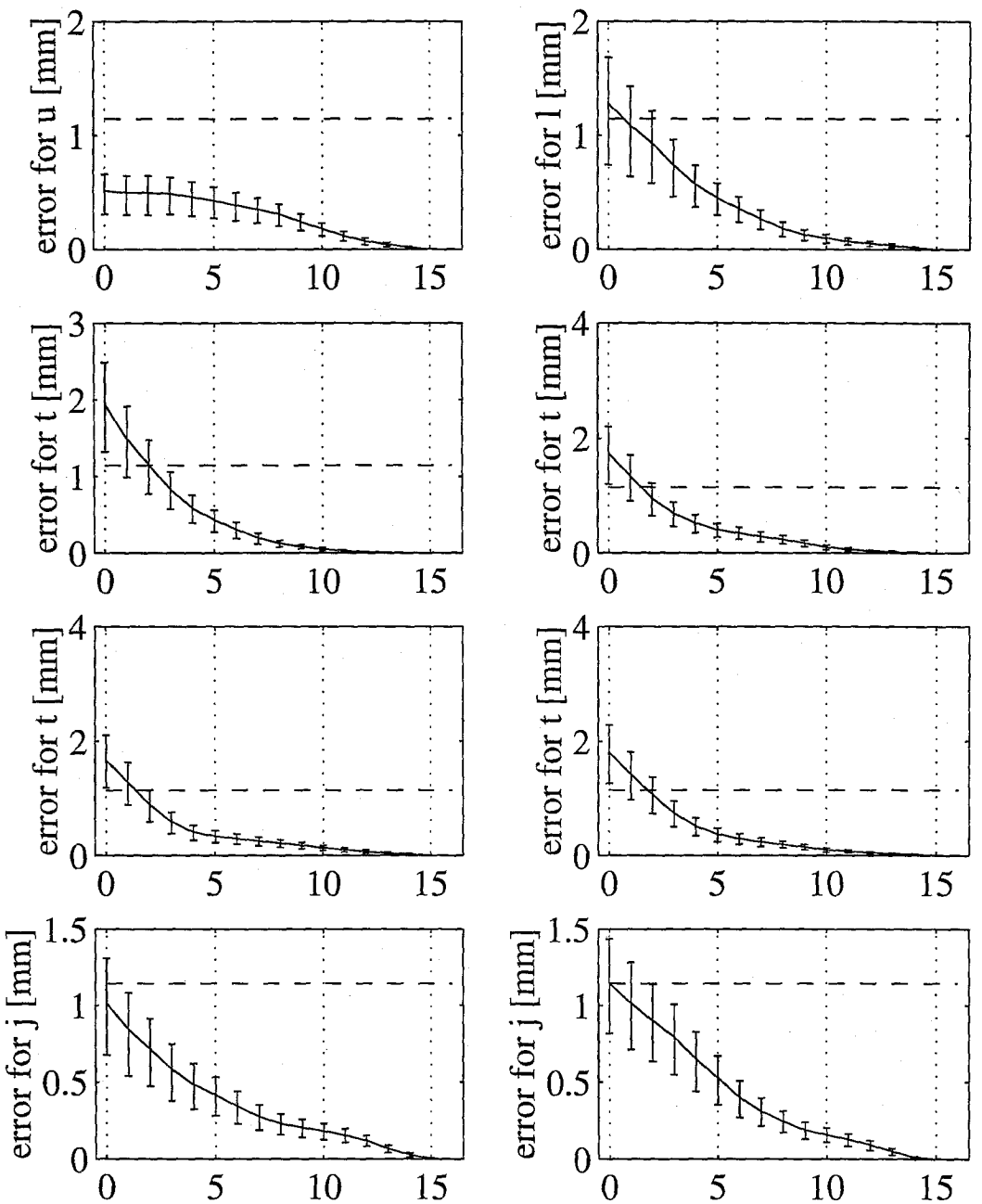


Figure 5.8: Individual bead errors using reconstruction by 32 local eigenvector models. Plotted are the median errors and the 85 and 15 percentile limits of the errors versus the dimensionality of each local model. The dashed horizontal line across each panel indicates the estimate of the machine error computed as described in section 3.3.4. The point indicated at zero eigenvectors represents the error reconstructing with the local means alone – this is equivalent to vector quantization of the data. In this case 32 separate linear models were used with their centres determined by Lloyd's algorithm. Compare with figure 5.5.

regions of space. An interesting generalization of this technique which was not investigated in this thesis) is a recently developed algorithm known as a *kernel support vector machine* [131]. Such nonlinear techniques all resulted in a slight improvement of reconstruction error, but at the cost of much larger models, much slower learning and much more difficult computations of reduced representations and reconstructions. Thus, for the remainder of this chapter I work only with the globally linear (PCA) and mixtures of locally linear (VQPCA) models. These models are very small in their numbers of parameters, extremely fast to work with since both compression and reconstruction involve only matrix operations, and (in the regime of data size I am working) do not suffer from overfitting.

5.4 Applications of manifold models

The previous section demonstrated that it is possible to identify both globally and locally linear models of the manifold on which the articulatory configurations lie. These models capture the structure of the data without sacrificing the accuracy with which it is represented. I now turn to some practical uses of these models. Here I focus on uses of these models which are consistent with the topic of this chapter in that they involve *only* articulatory data. These applications are filling-in of missing data from the database, de-noising and outlier detection and building simple mechanical models. As discussed in chapters 8 and 9, these models will also be later used to model the relationship between acoustics and articulatory configurations, to provide a representation in which to do simple recognition of utterances and to make initial investigations into speaker independent modeling.

5.4.1 Mistrack elimination

As mentioned in section 3.2.3, the X-ray microbeam device which records the articulator movement data occasionally loses one or more of the beads in a condition known as a *mistrack.* The result is that position information about mistracked beads is unavailable for the duration of the failure; sometimes as little as a 100ms and sometimes as long as the entire utterance. It is tempting to ignore this problem and discard the problematic data records. Unfortunately, mistracks are common enough that if one were to simply throw out all utterances in which any mistrack at all occurred, more than half of the database would be eliminated. Another strategy is to use sophisticated data analysis and time series modeling techniques which can deal with missing data in a

principled way. These techniques do not attempt to replace the missing data, but rather leave it missing and perform the rest of their calculations taking this into account. However, such methods are often difficult to understand or implement and in many cases no accepted approaches even exist.

Instead, I propose to *fill in* the missing data by employing the manifold models learned above. The basic idea is straightforward: if we have partial information about the articulator configuration, this defines some constraint surface in articulatory space within which the true configuration lies. If we intersect this constraint surface with our manifold model of the data, we can often uniquely recover the full articulatory configuration. For example, if we know where *some* of the beads are but not others, the partial information defines an axis-aligned constraint hyperplane of dimensionality equal to the number of unknown coordinates and perpendicular to the known directions.

5.4.2 Methods: subspace projection vs. direct regression, error bars

The implementation of the above idea in practice is relatively straightforward although it requires some careful linear algebra to derive. For global PCA models we simply find the point in the eigensubspace which is "closest" to the constraint plane defined by the partial coordinates we know. The only trick is that we must be careful in our computation of "closest." Using simple least squares will not work because we want distance to be measured according to the eigenvalues in the eigenspace. In particular, assume that left multiplication by the matrix $\mathbf{G}$ rotates into the eigenbasis and that left multiplication by $\mathbf{G}^T$ rotates out of the eigenbasis into the original coordinates. We have a situation in which we would like to find a vector $\mathbf{y}$ which satisfies:

$$\mathbf{y} = \mathbf{G}^T\mathbf{x} \tag{5.9}$$

given partial info about $\mathbf{x}$. If we directly solve the least squares system above, we will get the $\mathbf{y}$ of smallest radius (Euclidean norm) that still satisfies the partial information about $\mathbf{x}$. However, what we want is the $\mathbf{y}$ of smallest radius *according to the eigenvalues.* So first we "whiten" $\mathbf{x}$ to get $\tilde{\mathbf{x}}$ by dividing each coordinate by the square root of the corresponding eigenvalue:

$$\tilde{\mathbf{x}}_k = \mathbf{x}_k/\sqrt{\gamma_k} \tag{5.10}$$

and then we solve using ordinary least squares.

In order to obtain error bars on the predictions, we use the global noise level computed by the probabilistic version of PCA (SPCA) introduced in chapter 4. This is an improvement of standard PCA which provides no probabilistic model and thus cannot provide error bars or estimate noise levels. However, one limitation is that the global noise model in SPCA is spherical and so it yields the same error estimate for all predictions regardless of which bead is being predicted.

The PCA subspace approach can be compared against a direct regression approach in which we consider the original covariance matrix as specifying a single Gaussian model for the data. If we take the conditional distribution of the unknown bead coordinates given the known bead coordinates, this yields a new Gaussian model. In particular, let the vector $\mathbf{z} = [\mathbf{x}^T \mathbf{y}^T]^T$ be normally distributed according to:

$$\mathbf{z} = \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \boldsymbol{\alpha} \\ \mathbf{b} \end{bmatrix}, \begin{bmatrix} \mathbf{A} & \mathbf{C} \\ \mathbf{C}^T & \mathbf{B} \end{bmatrix} \right) \tag{5.11}$$

where $\mathbf{C}$ is the (non-symmetric) cross-covariance matrix between $\mathbf{x}$ and $\mathbf{y}$ which has as many rows as the size of $\mathbf{x}$ and as many columns as the size of $\mathbf{y}$. Then the marginal distributions are:

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\alpha}, \mathbf{A}) \tag{5.12}$$

$$\mathbf{y} \sim \mathcal{N}(\mathbf{b}, \mathbf{B}) \tag{5.13}$$

and the conditional distributions are:

$$\mathbf{x}|\mathbf{y} \sim \mathcal{N}\left(\boldsymbol{\alpha} + \mathbf{C}\mathbf{B}^{-1}(\mathbf{y} - \mathbf{b}), \mathbf{A} - \mathbf{C}\mathbf{B}^{-1}\mathbf{C}^T\right) \tag{5.14}$$

$$\mathbf{y}|\mathbf{x} \sim \mathcal{N}\left(\mathbf{b} + \mathbf{C}^T\mathbf{A}^{-1}(\mathbf{x} - \boldsymbol{\alpha}), \mathbf{B} - \mathbf{C}^T\mathbf{A}^{-1}\mathbf{C}\right) \tag{5.15}$$

computing the mean of the conditional distribution is equivalent to doing direct regression from the known beads onto the known beads. The standard deviation of the conditional distribution provides error bars.

5.4.3 Filling in with mixtures of local models

It is of course also possible to use the mixtures of local linear models to do filling-in. Within each sub-model, filling in is performed exactly as described above—the complete point is computed

as the point of minimum distance from the known subspace according to the local eigenbasis for the sub-model. The only difficult question is how to pick the correct sub-model (VQ region). This is an issue because with incomplete data we cannot merely chose the sub-model whose mean is closest to the current point. The approach I have taken is to choose the sub-model based on Euclidean distance in the reduced space of known bead coordinates. Then, once the sub-model has been selected and the missing information is filled in, I can estimate distance in the full space and see if the VQ region has changed. If it has, I can repeat the filling-in procedure using the new sub-model. In the experiments below, this iteration has always converged to a self-consistent estimation of sub-model although such convergence has not been proved theoretically.

5.5 Filling in procedures for real data

It is straightforward to apply the strategy outlined above to real articulatory data. Initially, the global PCA model and VQPCA sub-models are fit using only the completely intact data for a single speaker. Then, the remaining sections of the data (that have missing values) are filled in. Finally, the global and local PCA models can be re-estimated using the original intact (complete) data plus a weighted fraction of the newly filled-in data. This can be repeated several times. Typically I have found that such re-fitting does not change the model estimates very much at all because even the intact data is more than enough to fit such simple models. (However, in future sections when the audio data is considered in conjunction with the articulatory position data I have found that data is extremely scarce.)

5.5.1 An example with real data

I will illustrate the filling-in technique described above on real data from the UBDB. As an extra validation step, intact sections of the database for which there are no missing values have been processed. In these tests, the traces of one or more beads that are actually known are suppressed or "blacked out" and then recovered using the principal subspace projection technique described above. This allows comparison of the filled in traces with the ground truth of known measurements. Following this validation, portions of the database in which data is actually missing are processed in a filling-in step. For these filling-in experiment there is of course no truth with which to compare. However, if the validation step is done by suppressing exactly the same beads as are missing in the filling-in step, then we have a statistical estimate of how large or small the errors

from filling-in are likely to be. It is important to note that this statistical estimate assumes the missing data in the database to be missing at random, i.e. in a way not related to the values of the coordinates. This is a strong but reasonable assumption given that the origin of the missing data is mistracking failure by a measurement machine and not, for example, non-respondents on a questionnaire (for which there is reason to believe that missing data is related to the values).

Figure 5.9 shows a schematic representation of the missing data structure involved in one of these experiments. Beginning with a single utterance (`tp011` for speaker `jw45`), the mistrack sections that will eventually be used in the filling-in step are noted. These are displayed in figure 5.9 as black regions. Also, validation regions are selected within the same utterance. The validation regions are chosen to be somewhat longer than the fill-in regions but are constructed to have exactly the same missing data patterns. Each region in the figure spans the time of the mistrack (fill-in region) or validation region and covers the row(s) corresponding to the bead or beads which were lost. For example, in the mistrack region labelled **a**, the tongue dorsum bead has been mistracked from approximately 2sec to 10sec while in mistrack region **b** the three front-most tongue beads are all missing.

The subspace projection method outlined above can then be applied to fill in the missing data from both the validation regions and the fill-in regions. Notice that this filling-in is done on a frame by frame basis and is distinctly *not* a smoothing or interpolation technique across time.[3] For the example shown, the regions labelled **x**, **y** and **z** are the validation regions while regions **a**, **b** and **c** are the actual fill-in regions.

Validation

The results of recovering missing beads in the validation regions are shown in figures 5.10, 5.12 and 5.11. Results for both global and mixtures of local models are shown. Notice that in regions **x** and **z** the reconstructed data matches the true data almost exactly. Also, the error estimates from the recovery procedure are correspondingly small. This is because only one bead has been mistracked in each case. The trajectories reconstructed by the local models are not jumpy even though no smoothing was performed across switchings between models. These two regions are examples of the procedure working well.

However, in region **y** the three most informative beads (tongue tip and tongue body beads)

[3]However, at very beginning and very end of the sections with missing data it is important to smooth for a few frames to line up with the known endpoint conditions. In the example presented here this smoothing is done over 5 frames at the beginning and end of each mistrack block only.

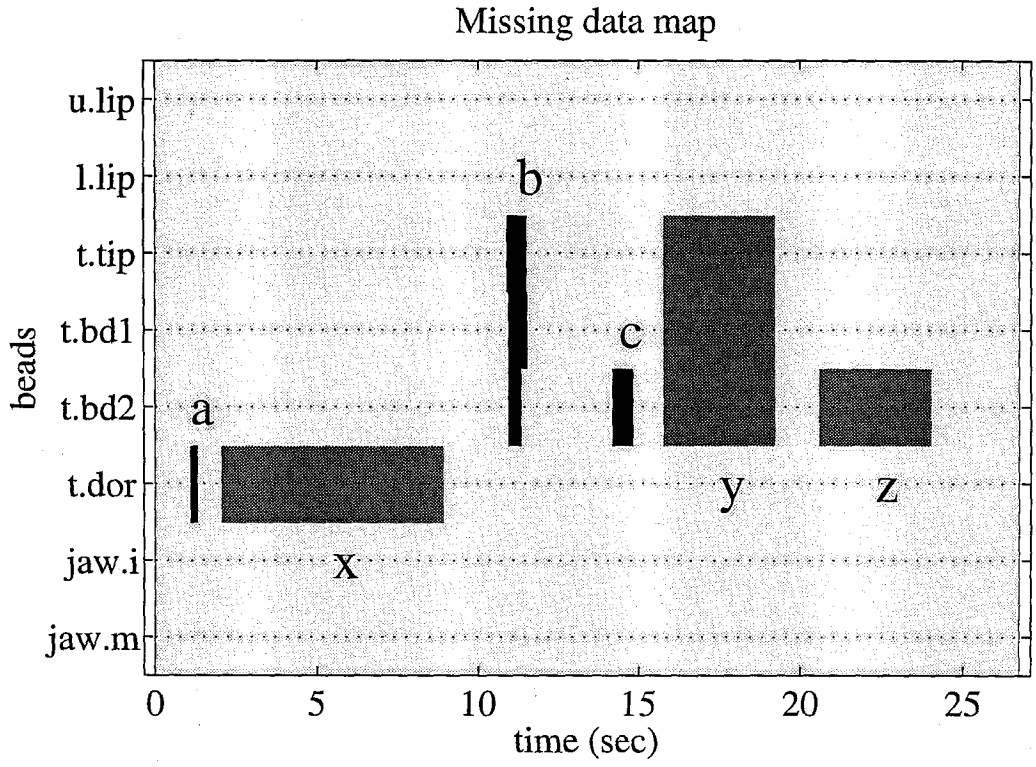


Figure 5.9: Map of missing data for filling-in example. Beginning with a single utterance (`tp011` for speaker `jw45`), the mistrack sections that will eventually be used in the filling-in step are noted. These are displayed as black regions (**a,b,c**). Also, validation regions are selected within the same utterance (**x,y,z**). The validation regions are chosen to be somewhat longer than the fill-in regions but are constructed to have exactly the same missing data patterns. Each region in the figure spans the time of the mistrack (fill-in) region or validation region and covers the row(s) corresponding to the bead or beads which were lost. For example, in the mistrack region labelled **a**, the tongue dorsum bead has been mistracked from approximately 2sec to 10sec while in mistrack region **b** the three front-most tongue beads are all missing. In validation region **y**, the three front-most beads have been blacked-out.

are all missing; thus the quality of the reconstruction is quite poor, and the error estimates quite broad. The local model trajectories are now quite jumpy and it is clear that in this case smoothing is necessary. This region is an example of the procedure not working well, although it is difficult for any method to recover information that is not present at all in the data.

In each of these validation regions, the same beads that were mistracked in the fill-in regions **a**, **b** and **c** were "blacked-out" in the complete data record. The recovery algorithm was then run to estimate the artificially "missing" beads and its output compared to ground truth. In all the figures the thick solid line shows the reconstruction. The dotted points indicate the true measured data. The thin solid lines indicate the error estimates from the reconstruction procedure. The vertical lines at the tops of the local model plots indicate the times at which the reconstruction procedure switched from one sub-model to another. No smoothing was done across these switching times or across any other times in the reconstructions (except at the block ends as described in footnote 3).

Reconstruction of truly missing data

After validating the recovery procedure on completely intact data, we can apply it to filling in data in sections of the database where mistracks have really occurred and thus data is missing. The identical procedure is applied except that in this case we do not have any ground truth with which to compare. Figures 5.13, 5.15 and 5.14 show the results of this data recovery in using both global models and mixtures of local models.

Once again, it is easy to see that in regions **a** and **c** the error estimates are relatively low while in region **b**, where the three most informative tongue beads are missing, the reconstruction procedure has large error estimates. The model switching causes noisy jumps in the recovered trace and smoothing would be necessary.

This reconstruction procedure has been applied to all of the data used in all subsequent analysis in this thesis. Once the reconstruction was performed, the new data was taken as being complete without regard to the size of error estimates from the reconstruction although this is clearly not the optimal way to proceed. However, for our purposes we will henceforth ignore the question of missing data in the movement trajectories.

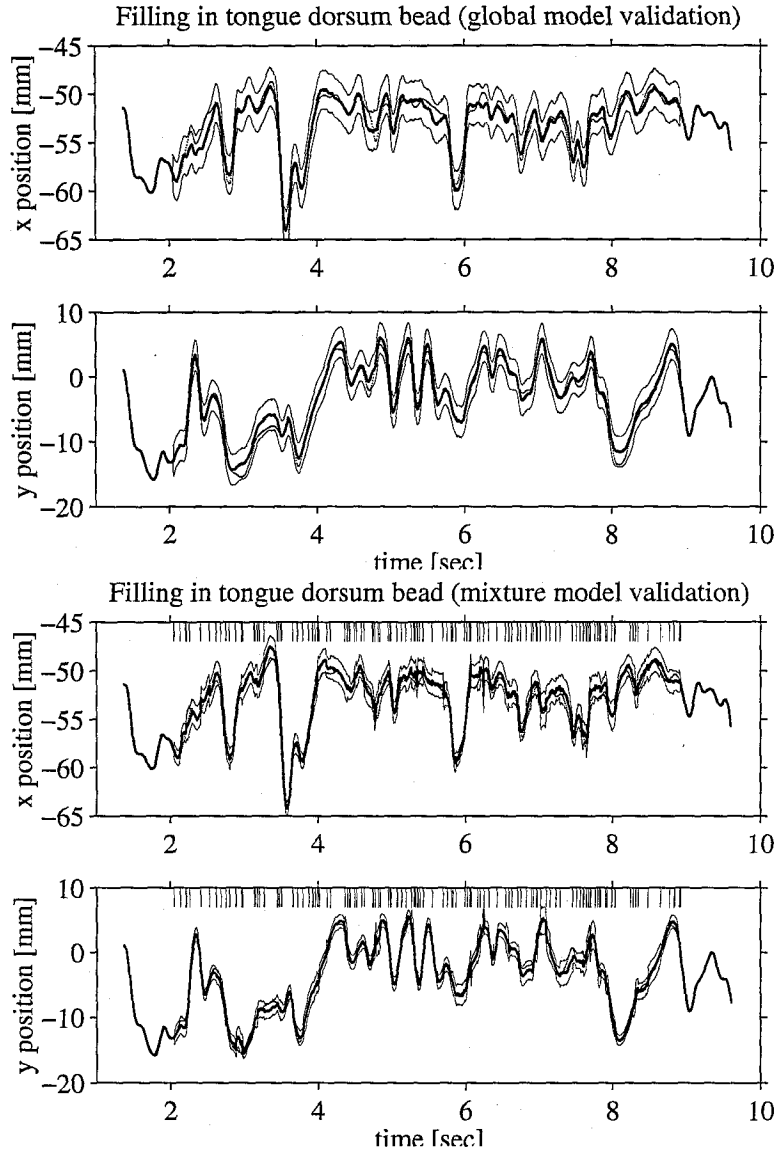


Figure 5.10: Reconstructions of bead data to validate the filling-in model. In this section of data, no values are missing. But the tongue dorsum bead has been artificially "blacked-out". The recovery algorithm was then run to estimate this artificially missing bead and its estimate compared to ground truth. In all the figures the thick solid line shows the reconstruction. The dotted points indicate the true measured data. The thin solid lines indicate the error estimates from the reconstruction procedure. The vertical lines at the tops of the local model plots indicate the times at which the reconstruction procedure switched from one sub-model to another. No smoothing was done across these switching times or across any other times in the reconstructions. Shown here is an approximately 10 second long section from utterance `tp011` for speaker `jw45`. This is the region labelled **x** in figure 5.9.

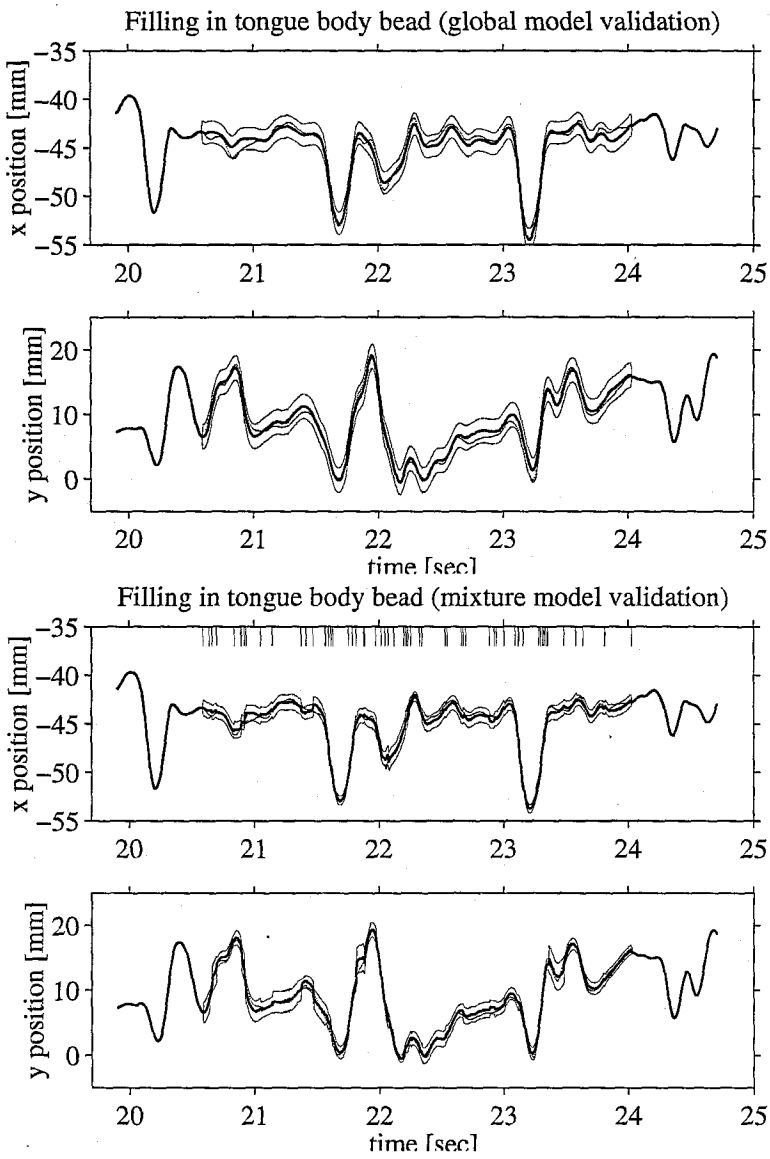


Figure 5.11: Reconstructions of bead data to validate the filling-in model. In this section of data, no values are missing. But the second tongue body bead has been artificially "blacked-out". The recovery algorithm was then run to estimate this artificially missing bead and its estimate compared to ground truth. In all the figures the thick solid line shows the reconstruction. The dotted points indicate the true measured data. The thin solid lines indicate the error estimates from the reconstruction procedure. The vertical lines at the tops of the local model plots indicate the times at which the reconstruction procedure switched from one sub-model to another. No smoothing was done across these switching times or across any other times in the reconstructions. Shown here is an approximately 4 second long section from utterance `tp011` for speaker `jw45`. This is the region labelled **z** in figure 5.9.

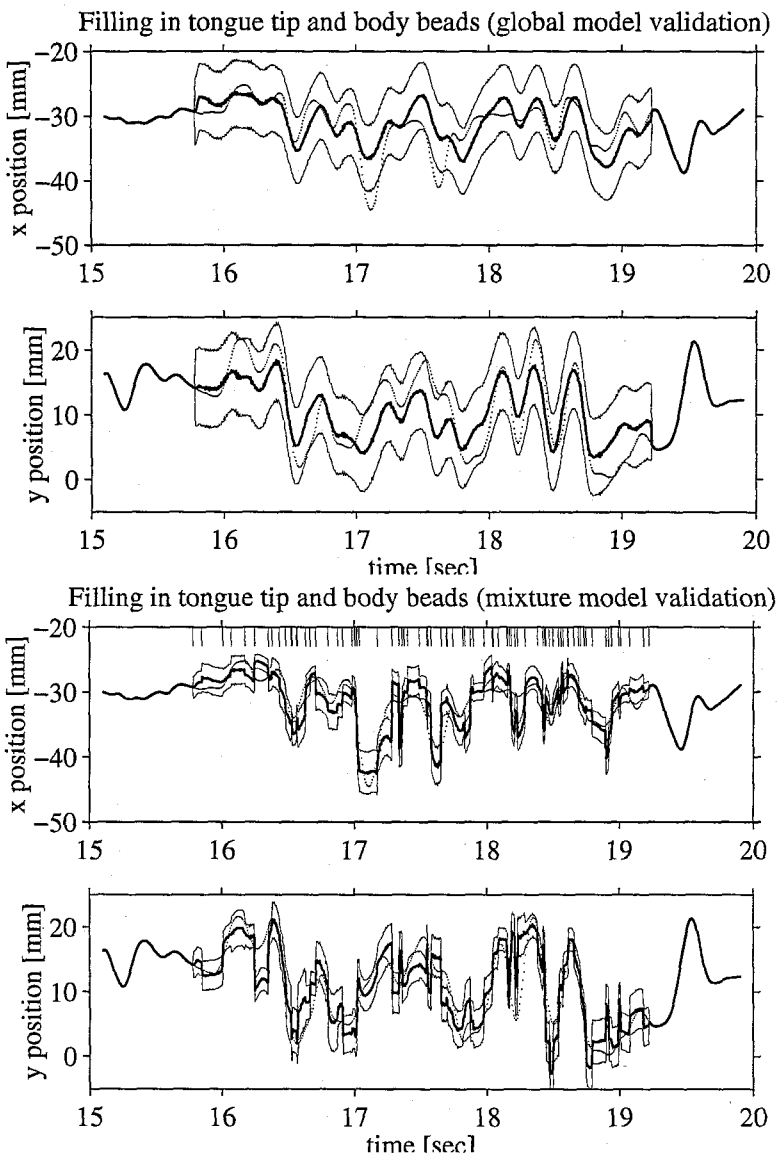


Figure 5.12: Reconstructions of bead data to validate the filling-in model. In this section of data, no values are missing. But the tongue body beads and tongue tip bead have been artificially "blacked-out". The recovery algorithm was then run to estimate thes artificially missing beads and its estimates compared to ground truth. In all the figures the thick solid line shows the reconstruction. The dotted points indicate the true measured data. The thin solid lines indicate the error estimates from the reconstruction procedure. The vertical lines at the tops of the local model plots indicate the times at which the reconstruction procedure switched from one sub-model to another. No smoothing was done across these switching times or across any other times in the reconstructions. Shown here is an approximately 4 second long section from utterance `tp011` for speaker `jw45`. This is the region labelled **y** in figure 5.9.

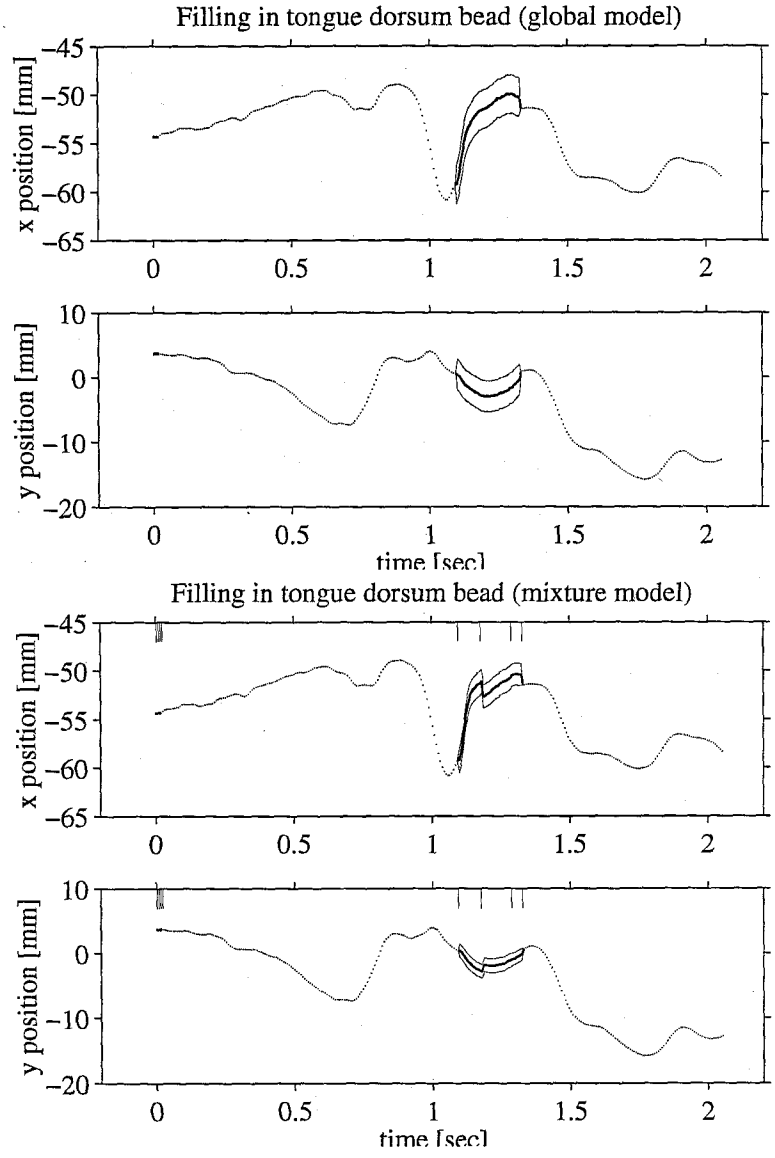


Figure 5.13: Reconstruction of missing data from the database. The recovery algorithm described in the text was run to estimate the missing trajectory for the tongue dorsum bead which has been briefly mistracked in the section of data shown. Thick solid line shows the reconstruction. The dotted points indicate the true measured data. The thin solid lines indicate the error estimates from the reconstruction procedure. The vertical lines at the tops of the local model plots indicate the times at which the reconstruction procedure switched from one sub-model to another. No smoothing was done across these switching times or across any other times in the reconstructions. Shown here is an approximately 0.5 second long section from utterance `tp011` for speaker `jw45`. This is the region labelled **a** in figure 5.9.

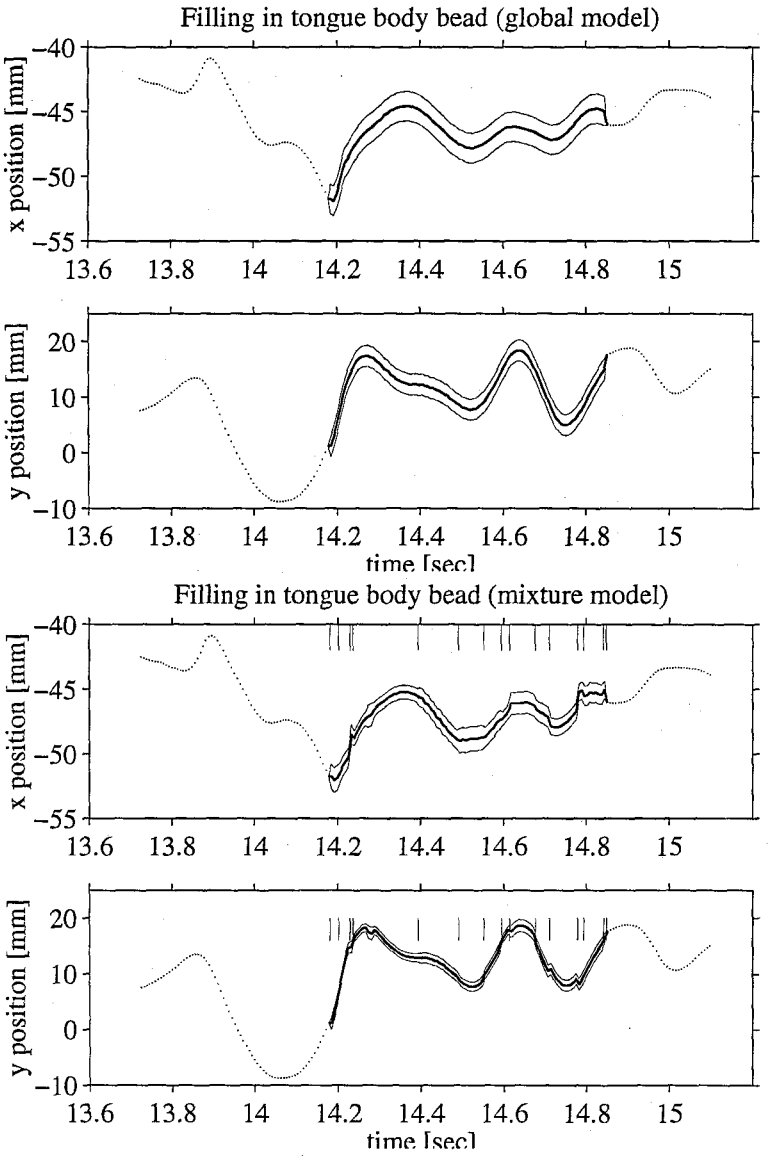


Figure 5.14: Reconstruction of missing data from the database. The recovery algorithm described in the text was run to estimate the missing trajectory for the tongue body bead which has been briefly mistracked in the section of data shown. Thick solid line shows the reconstruction. The dotted points indicate the true measured data. The thin solid lines indicate the error estimates from the reconstruction procedure. The vertical lines at the tops of the local model plots indicate the times at which the reconstruction procedure switched from one sub-model to another. No smoothing was done across these switching times or across any other times in the reconstructions. Shown here is an approximately 1 second long section from utterance `tp011` for speaker `jw45`. This is the region labelled **c** in figure 5.9.

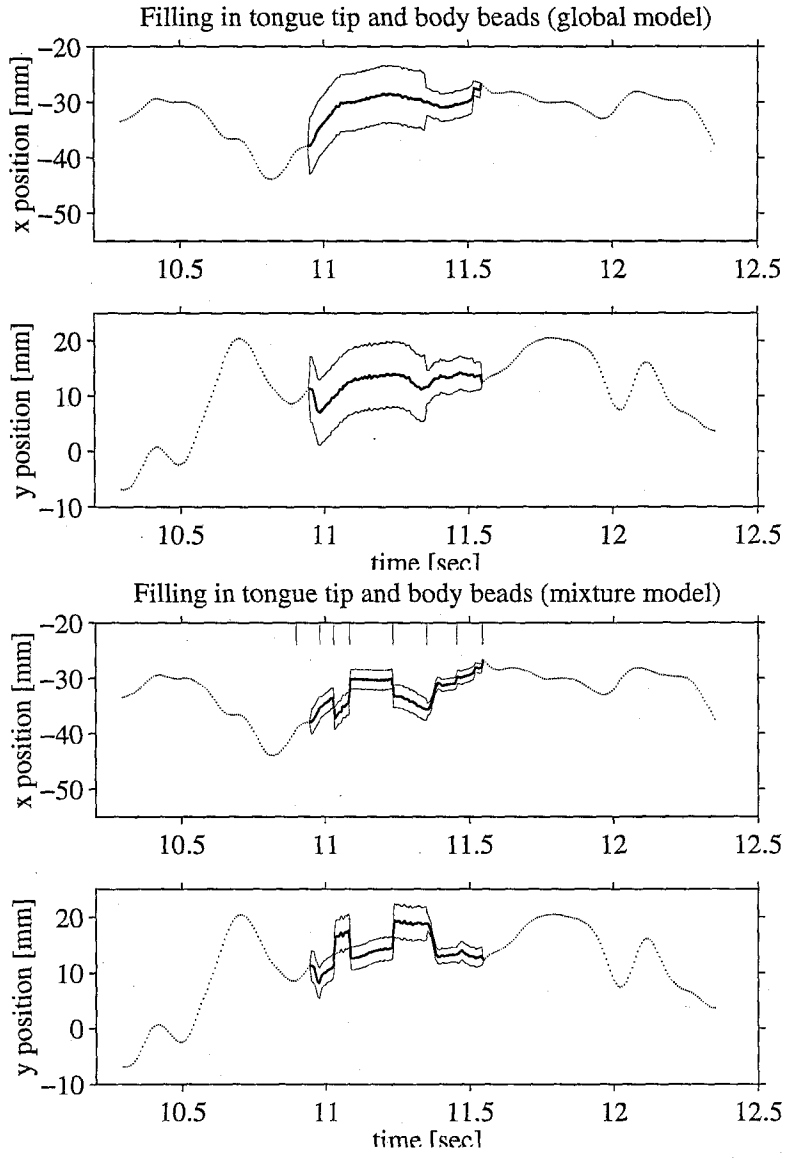


Figure 5.15: Reconstruction of missing data from the database. The recovery algorithm described in the text was run to estimate the missing trajectory for the tongue body and tongue tip beads which have been briefly mistracked in the section of data shown. Thick solid line shows the reconstruction. The dotted points indicate the true measured data. The thin solid lines indicate the error estimates from the reconstruction procedure. The vertical lines at the tops of the local model plots indicate the times at which the reconstruction procedure switched from one sub-model to another. No smoothing was done across these switching times or across any other times in the reconstructions. Shown here is an approximately 1 second long section from utterance `tp011` for speaker `jw45`. This is the region labelled **b** in figure 5.9.

5.5.2 A pseudo-mechanical kinematic model

The manifold models and associated subspace projection algorithm can be used together in another interesting application. They can serve as a mechanical model of the tongue. Just as in a normal mechanical model, the subspace projection specifies the equilibrium positions of some points if others are clamped to certain locations. It is even possible to compute the equivalent of force—the derivatives of free bead positions with respect to the clamped beads positions describes how strongly the model will try to move a free bead in response to a change in position of a clamped bead. Because the calculations involved are straightforward matrix operations, such a model can easily be constructed to run in real time. Figure 5.16 shows a screen shot of a MATLAB application constructed in this way. The user can click on any bead and move it using the mouse. The program will move the rest of the beads according to the model of articulator configurations. The user can also set the number of underlying degrees of freedom.

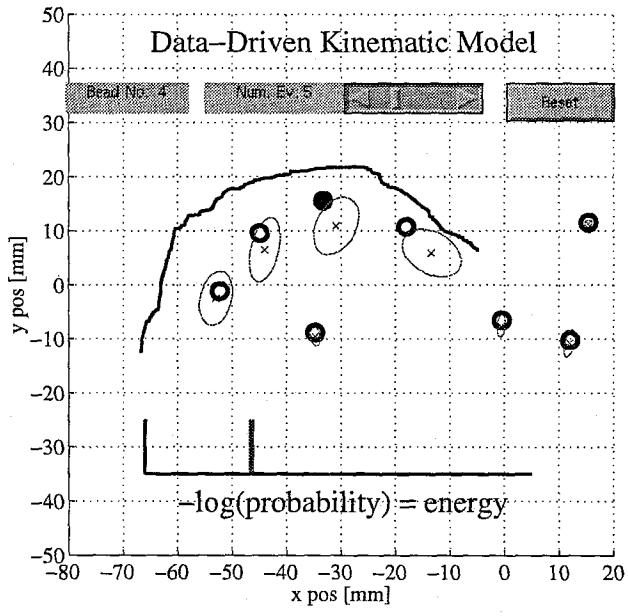


Figure 5.16: Screen shot of a MATLAB application which implements a pseudo-mechanical model of the tongue. The subspace projection method described in the text specifies the equilibrium positions of some points if others are clamped to certain locations. The user can click on any bead and move it using the mouse. The program will move the rest of the beads according to the model of articulator configurations. The user can set the number of underlying degrees of freedom.

Chapter 6 Articulatory to Acoustic Forward Mapping

The analysis of chapter 5 used only the articulatory portion of the database and answered questions about the set of observed mouth configurations. In this chapter we examine the synchronously recorded audio portion of the database and investigate the **forward mapping** between articulator positions and spectral information. The term forward indicates a believed causality—since at some level the lungs, vocal tract and articulators form a real physical system, it should in principle be possible to compute from their states the spectrum of the speech pressure wave. Physics, after all, does this computation by direct simulation! The goal here is to do it with rather less effort—admittedly at the expense of rather less fidelity and using rather less information. Physics, of course, has available to it the complete state of the speech production apparatus including the detailed three-dimensional shape of the vocal tract and the exact excitation pressure from the glottis. The computer algorithms described below have knowledge of only the midsaggital measurements of eight key points on the articulators. Whether or not this **limited** information about the state of the vocal apparatus is sufficient to recover acoustic information is an open question **not** directly answered by the causality of physics. Below I answer it empirically by demonstrating a positive example of a system that synthesizes audio from articulator positions. The intuition behind the success of such a system is that while the midsaggital measurements do not alone contain enough information to directly perform a forward physical simulation, they seem to be sufficiently correlated with other quantities that it is possible to *infer* acoustic information from them. I investigate both an **instantaneous** mapping (which attempts to predict spectral shape and signal energy from only the midsaggital positions of the articulators) and a dynamic extension which uses articulator positions and their instantaneous velocities. The dynamic model does not perform significantly better than the instantaneous model suggesting that for this limited recovery task a time-invariant mapping is sufficient.

Quantifying success in this task is difficult. While it is possible to use numerical measures such as squared error to indicate how well certain acoustic parameters are estimated from the mouth shape, these measures often do not correlate well with how intelligible the resulting sequence of spectra sounds to a human as synthetic speech. Whenever possible I will include example audio files of resynthesized utterances so that the reader (also a listener) may judge the perceptual

aspects herself. In judging such results it is important to remember that all resynthesis begins with a short-time spectral representation of the signal such as a spectrogram and attempts to invert this representation back into the time domain. Because the spectrogram preserves only the magnitude and not the phase of the short-time spectra, this inversion is not exact nor straightforward. Even resynthesis from the **true spectrogram** suffers from a considerable number of artifacts. Thus it is always important in resynthesis experiments to compare the sound obtained by inversion of the original (true) acoustic parameters with the sound obtained by inversion of the estimated acoustic parameters. Comparing to the original raw waveform is not a fair test.

6.1 Audio data parameterization

The speech pressure wave in the database is sampled at approximately 21739Hz. In order to make a one-to-one association between articulatory frames and acoustic information, it is necessary to summarize in a single feature vector all of the new information contained in the approximately 150 samples of speech that arrive between every two articulatory frames. Motivated loosely by the perceptual observation that humans are insensitive to short-time phase but more strongly by engineering tradition (see Appendix A) we employ the general technique of *short-time spectral analysis*. In all of the experiments described below, short-time spectral analysis was performed with a frame rate exactly equal to the frequency of articulatory samples, i.e. with period 6.866ms. Notice that in order to maintain this as an average frame rate, the system has to alternate between skipping 149 samples and 150 samples between analysis windows as appropriate. The frame length was taken to be exactly 512 samples corresponding to approximately 23.6ms. A Hamming window was applied to each frame before spectral analysis. Pre-emphasis was applied to the original signal, using a weighting of $\alpha = 0.95$.

6.1.1 Spectrogram-like characterizations of the audio signal

Spectrogram analysis generates a spectral feature vector which encodes in some way a smoothed version of the amplitude of the power spectrum for a short portion of the audio signal around a certain time. The section of the signal which is examined to determine each short-time spectrum is called the *analysis window* or *analysis frame*. From one feature vector to the next, the analysis window moves forward in the audio signal by a time called the frame shift, the inverse of the *frame rate*. Below I review quickly some of the basic equations for such analysis. (But see also

Appendix A for more details.)

The simplest kind of spectrogram analysis is the *periodogram* which can be thought of as a "DFT-o-gram." The periodogram takes successive analysis windows of the signal and computes their discrete Fourier transform (DFT). Often a window such as a Hamming or Hanning window is applied to each frame to lessen edge effects that come from the implicit periodic boundary assumptions of the DFT. The resulting power spectrum information, however, is highly correlated across frequency. The power estimates returned by the DFT are also heavily biased by windowing and finite sample effects. Neighbouring frequency bins contain similar power but it is difficult to estimate accurately the power in a narrow frequency region using only a small number of samples. For this reason the periodogram is often replaced by or converted to another time-frequency representation in which the spectra are implicitly *smoothed* across both frequency and time. One such representation is a filter bank. Another is to use a low order autoregressive model of the signal amplitude values. An autoregressive model predicts the current signal amplitude as a linear function of previous values:

$$s_t = a_1 s_{t-1} + a_2 s_{t-2} + \ldots + a_k s_{t-k} + G \cdot u_t \tag{6.1}$$

where u_t is a scalable driving or noise term that accounts for the difference between the prediction and the true sample and G is the noise gain. Taking z-transforms of the above model yields:

$$S(z) = a_1 \frac{S(z)}{z} + a_2 \frac{S(z)}{z^2} + \ldots + a_k \frac{S(z)}{z^k} + G \cdot U(z) \tag{6.2}$$

which gives an all-pole model of the transfer function from the driving noise u_t to the output signal s_t:

$$H(z) = \frac{S(z)}{G \cdot U(z)} = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2} - \ldots - a_k z^{-k}} = \frac{1}{A(z)} \tag{6.3}$$

This is known as *linear predictive coding* of speech, and it is this term which gives the commonly used abbreviation LPC. The coefficients a_k are the *linear prediction coefficients.* It is common in LPC analysis to normalize the coefficients so that $a_0 = 1$ always as shown above, and to store the total gain in each frame as a separate one-dimensional energy signal G_t.

6.1.2 Line spectral pairs (LSP)

Since LPC coefficients form an all-pole model of the short-time spectrum, they implicitly encode the peaks in spectral energy. These peaks are often associated with formants in the speech and so this technique is useful and effective. The LPC coefficients can be difficult to work with, however, because they are related in a complex way to the locations of the poles they are modeling. In particular, the peaks in the spectrum occur at the roots of the polynomial:

$$A(z) = 1 - a_1 z^{-1} - a_2 z^{-2} - \ldots - a_k z^{-k} \tag{6.4}$$

which means that a small change in one of the coefficients causes a difficult to model change in the peak locations. Furthermore, it is difficult to write down simple constraints on the coefficients such that the corresponding filter is *stable*—that is has all its poles inside the unit circle in the z plane. While stable filters are not technically essential for speech analysis and resynthesis they are much easier to work with since they can be excited with white noise sources and will always produce bounded outputs with well defined spectra.

For these reasons, in all of the analyses in this thesis I use a set of coefficients derived from the LPC coefficients, but which are much better behaved, called the *line spectral frequencies* or *line spectral pairs* (LSP).As their name implies, the line spectral frequencies are actual frequencies (and thus always positive and less than half the sample rate). They contain exactly the same information as the LPC coefficients, but they represent it by clustering around the peaks in the spectrum. This means that moving a single LSP affects only the peak near its frequency. Furthermore, the relationship is straightforward to understand: if we lower a LSP the nearby pole is moved down in frequency and if we raise a LSP the nearby pole is moved upwards. A final attractive property of the line spectral pairs is that as long as the successive line spectral frequencies are kept *monotonic*—in other words $lsp_{k+1} > lsp_k$—the corresponding all-pole filter is guaranteed to be stable. There are exactly the same number of LSP frequencies as LPC coefficients. They are defined by the roots of the two polynomials:

$$P(z) = A(z) + z^{-(k+1)} A(z^{-1}) \tag{6.5}$$

$$Q(z) = A(z) - z^{-(k+1)} A(z^{-1}) \tag{6.6}$$

where $A(z) = 1 - a_1 z^{-1} - a_2 z^{-2} - \ldots - a_k z^{-k}$ is the LPC polynomial given above. The

polynomials $P(z)$ and $Q(z)$ have the property that all of their roots lie on the unit circle. The roots can thus be expressed in the form $e^{j\omega}$ and the ω's are then called the LSP frequencies. It is furthermore true that the zeros of $P(z)$ and $Q(z)$ are interlaced with each other which gives rise to the monotonic ordering property mentioned above. As long as these two conditions are preserved, the corresponding LPC filter will always be stable. Figure 6.1 shows examples of fitting a spectrum with line spectral pairs. It illustrates the effects of moving the pairs in frequency and of changing the distance between paired frequencies. Generally peaks in the estimated spectrum occur near each pair with the bandwidth set by the separation between the pair.

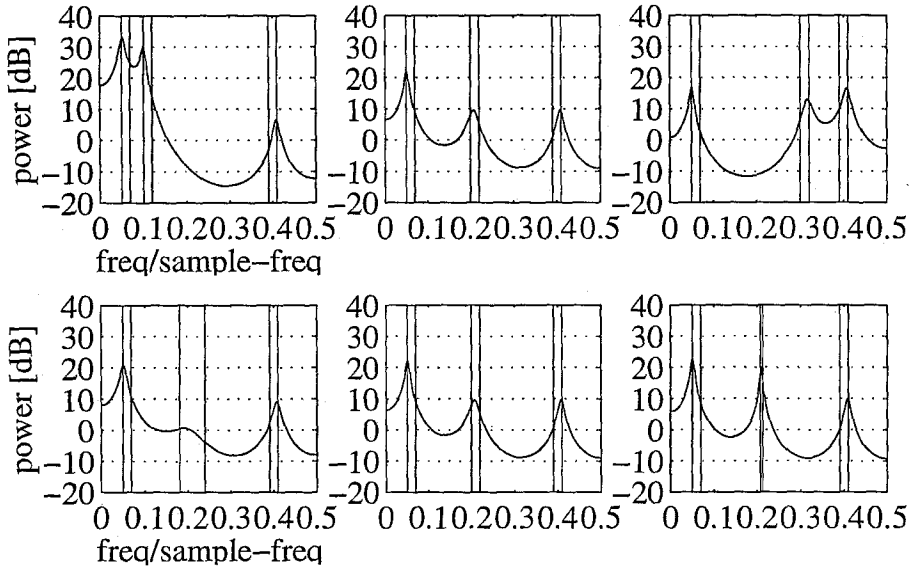


Figure 6.1: Line spectral frequency parameterization of a spectrum. The top panels show the effect on the spectrum of moving the frequencies of a line spectral pair together; the right panels show the effect of separation between a pair of frequencies. Generally peaks in the estimated spectrum occur near each pair with the bandwidth set by the separation between the pair.

6.1.3 Resynthesis of a time domain waveform from short-time spectra

Resynthesis of a time-domain audio signal from short-time spectral representations (plus an energy signal) is possible in many ways. The procedure I have used involves exciting the filter defined by each short-time spectral frequency vector (measured or computed during one corresponding analysis frame) with a source of the correct power as defined by the energy signal. A spectrally broad source should be used to allow the resonances estimated by the short-time filters to be excited no matter where they lie. A common choice for such broadband excitation is a white noise

source analogous to the turbulent excitation present in unvoiced sounds (for example fricatives) in speech. In this case the resulting synthetic signal sounds "breathy," "whispered" or "hoarse." Another choice is to use a pulse sequence analogous to the pulses of air coming from the glottis during voices speech. This causes the synthesized signal to sound "robotic" or "buzzing." It is possible to estimate the degree of voicing from the original signal using features such as zero-crossing rate and proportion of low-frequency energy. The excitation used for resynthesis can then be modified on a frame by frame basis to contain more or less driving noise or driving pulse train power. I mention again that in general such reconstructions (even when performed using the **true** short-time spectral parameters as measured from the original audio signal) are understandable but unnatural sounding. However, high quality synthesis is not the purpose of this investigation. The key point is that virtually *any* resynthesis scheme that takes into account the correct frame energy and magnitude spectrum will result in intelligible although unnatural sounding synthetic speech. In order to familiarize the reader with such reconstructions two examples are provided:

Audio file 6.1.1 (`track 4`) *An example utterance.*

Audio file 6.1.2 (`track 5`) *The same utterance resynthesized from a 12th order LPC analysis using white noise excitation.*

Audio file 6.1.3 (`track 6`) *Another example utterance.*

Audio file 6.1.4 (`track 7`) *The second utterance resynthesized from a 12th order LPC analysis using a mix of white noise and glottal pulse excitation.*

These utterances are synthesized in the time domain by converting the LSP coefficients back into LPC a_k coefficients and then substituting into the forward equations for autoregressive analysis using the desired excitation as the source u_k:

$$s_t = a_1 s_{t-1} + a_2 s_{t-2} + \ldots + a_k s_{t-k} + u_k \tag{6.7}$$

Initially, the previous samples $s_{-1}, s_{-2}, \ldots$ are taken to be zero; the filter is run for a length of time twice the length of the reconstruction frame desired in order to eliminate the transient effects at the beginning. The final half of the filter output is used in the resynthesis procedure. The excitation is taken to be a constant mixture of white noise and a glottal pulse (I have used the popular LF or Liljencrants-Fant glottal excitation model from [55]). The reconstruction frames were taken

to be the length of the frame shift between analysis frames and were simply concatenated together to form the final utterance. Throughout the remainder of the chapter I will use this basic reconstruction technique to illustrate the results of recovering audio from articulation. The reader is cautioned that the unnatural sound of these reconstructions is due largely to the fact that they are reconstructions from short-time spectra rather than to the fact that they are estimating acoustics from articulation.

NOTE: In principle it is possible to do better than naive excitation of the spectrogram filters when the analysis windows overlap in time. This is because if **overlapping** analysis windows are used, then the short-time spectrum of each is estimated with some samples in common with the previous window. This places restrictions not only on the magnitude of the short-time spectrum but also on its phase. So called *homomorphic* analysis techniques attempt to take advantage of this overlap to reconstruct a more complete signal that is consistent with the implicit phase constraints. However, these techniques are much more complex to implement and much more computationally demanding than direct excitation. Thus for the present work I have taken the most basic reconstruction approach.

6.2 Supervised learning: from articulators to audio

Below I present the results of a system which learns to synthesize utterances based only on the movements of the articulators. The system works by estimating the line spectral frequencies and signal energy from knowledge of the articulator bead positions. As will be shown, the resulting estimation of spectral shape, even when very simple time-independent models such as linear regression or piecewise linear regression are used, is quite good. The resynthesized utterances are intelligible and comparable in quality to the corresponding resynthesis using the true features measured from the original audio signal. These results allow a relatively strong *empirical* statement to be made about articulatory and acoustic data in normal English speech: *The instantaneous positions of the articulators alone contain sufficient information to predict the gross features of the instantaneous spectrum of the speech wave.* I will also present results of attempting to recover the energy of the signal based on articulator positions. This recovery is more difficult but can still be performed with some success—indicating that simple intuitions (when the mouth is open the

speech is loud, when the mouth is almost closed the speech is quiet) are to some extent supported in the real data. Using these two estimations (spectral shape and energy) together, it is possible to "add sound" to midsaggital "silent movies" without any reference to the original audio signal. I also present results of adding instantaneous derivative information to the time-invariant (position only) models. While a slight perceptual improvement in the quality of synthesized utterances is achieved, the models that also include velocity do not perform significantly better by any quantitative metric. This suggests that for this limited recovery task at hand, and for the limited power of the linear models being employed, a time-invariant (instantaneous) mapping is sufficient.

6.2.1 Why learn forward mappings from data?

Traditional articulatory synthesis has been based on physical models. Some models are extremely simple and use only two or three parameters to characterize the vocal tract while others involve extremely detailed three-dimensional finite element simulations of fluid flow in the vocal apparatus. In all cases, however, the approach to synthesizing audio from measured articulatory data is to first estimate the parameters of the articulatory model from the measured articulatory data and then to synthesize sound based on the articulatory model. An alternate approach is to dispense with an explicit physical model and use a "nonparametric" technique which learns to reconstruct audio directly from the measured articulatory information.

Such data-driven approaches to learning a forward mapping based only on example data are motivated by several goals. The first is to quantify the widely held belief that forward mapping for the vocal tract system is a well defined function. This can be analytically shown for a variety of *models* merely by invoking the causality of physics. However, it is an entirely different matter to make an empirical statement such as the one above about observed data from real human speech. The fact that such a statement has been made based on only *limited* articulator information is important to note but only makes the case stronger: if a system with access to only partial information can perform acceptable synthesis, then a system with access to more complete information will certainly do better. While this result is not surprising, it is certainly an important quantification to have made.

Another reason for using data-driven models is to compare with traditional physically based models. One advantage that data driven models have over physically based models is that they can potentially capture information that is present in the example data but may be excluded by the

assumptions of an explicit model. For example,[1] if the articulator bead positions, although they are measured on the midsaggital plane, contain some information about the three-dimensional shape of the mouth, then this information can be used in the forward mapping simply by the fact that it will cause useful correlations between observed spectra and bead positions. However, a physical model such as a uniform tube model which explicitly constructs a hypothesized shape based only on the cross-sectional area will never benefit from this feature of the data. It is possible to augment physical models with explicit functions that transform midsaggital areas to three-dimensional cross sections; and indeed this is in effect making them more data driven. On the other extreme, the success of a purely supervised learning approach allows one to make the important claim that all the *essential features* of a full physical model can be captured by looking only at example data.

A third motivation to develop an effective statistical forward model is that it may be a powerful research tool. Once we have a model that performs satisfactorily for the articulatory to acoustic mapping, we can use this model to investigate the acoustic to articulatory inversion process (see chapter 8). Inverting a simple model is an easier research problem to start with than working with the full dataset: certainly it is often simpler to understand why one can or cannot recover the underlying states of a mathematical model than to understand why one can or cannot predict one data stream from another. The forward model can also be used to generate a large amount of extra data or to generate data in a particular parameter regime not well represented in the original recordings. This synthetic data can be used to assist in training or initializing an inverse model.

Lastly, the result of successful forward synthesis system may be of interest to speech science and linguistics. Almost certainly a trained linguist could learn to watch a movie of articulator movements and then describe what the corresponding audio would sound like. After all, lipreaders (either deaf or not) clearly exhibit this skill. However, the extent of knowledge these humans employ in learning to do such a task is unknown. It sets an important benchmark to exhibit a simple model which can do the same thing. For example, if an audio track from one of the UBDB recordings was completely destroyed, the automated system outlined below would be able to reconstruct it sufficiently well that a human listening to the resynthesized audio would be able to understand what was spoken.

[1]Thanks to John Hopfield for suggesting this simple example.

6.2.2 Methods

I have trained simple linear regression models (see Appendix B) and also piecewise linear models to predict ℓ LSP coefficients and the log energy from only measured bead positions. In particular, these techniques estimate the LSP coefficient vector $\mathbf{y}$ and the log energy E from the articulatory positions $\mathbf{x}$ according to the relationship

$$\mathbf{y} = \mathbf{A}\mathbf{x} + \mathbf{y}_0 \tag{6.8}$$

$$E = \mathbf{e}\mathbf{x} + E_0 \tag{6.9}$$

where the matrix $\mathbf{A}$ is of size $\ell \times 16$, the vector $\mathbf{e}$ is of size 1×16 and $\mathbf{y}_0$ and E_0 are bias terms. The results presented below use models that do linear regression directly from the 16 bead coordinates to $\ell = 12$ LSP coefficients. A variety of other regression techniques were investigated, all with very similar results, for example models that take $\mathbf{x}$ as the first few coordinates in the eigenspace of bead positions (see chapter 5). These models perform slightly less well when the number of eigenvectors used is low. (And they are exactly equivalent when all eigenvectors are used since regression from the original axes and from the eigenbasis differs only by a rotation matrix which can be absorbed into $\mathbf{A}$ above). However, these reduced eigenvector models have the potential (as discussed in chapter 9) to be used in speaker independent synthesis systems.

Another variation that was investigated was to include the instantaneous first and second derivatives of the bead positions as extra coordinates in $\mathbf{x}$ for the linear regression. These derivatives can be computed using a cubic spline approximation to each bead coordinate time series in the vicinity of each articulatory frame. The cubic spline is an interpolant between points (x_i, y_i) which has continuous and smooth first derivative and continuous second derivative. Between any two known points (x_j, y_j) and (x_{j+1}, y_{j+1}), it has the form

$$y(x) = \alpha_j y_j + \beta_{j+1} y_{j+1} + \gamma_j y''_j + \delta_{j+1} y''_{j+1} \tag{6.10}$$

$$\alpha_j = (x_{j+1} - x)/(x_{j+1} - x_j) \tag{6.11}$$

$$\beta_{j+1} = 1 - \alpha_j \tag{6.12}$$

$$\gamma_j = (\alpha_j^3 - \alpha_j)(x_{j+1} - x_j)^2/6 \tag{6.13}$$

$$\delta_{j+1} = (\beta_{j+1}^3 - \beta_{j+1})(x_{j+1} - x_j)^2/6 \tag{6.14}$$

where y'' denotes the second derivative of y with respect to x. These second derivatives are computed as the solutions to the equations

$$(x_j - x_{j-1})y''_{j-1} + 2(x_{j+1} - x_{j-1})y''_j + (x_{j+1} - x_j)y''_{j+1} = \frac{6(y_{j+1} - y_j)}{(x_{j+1} - x_j)} - \frac{6(y_j - y_{j-1})}{(x_j - x_{j-1})} \tag{6.15}$$

except at the edges of the data ($j = 1$ and $j = N$) at which either the first or second derivative must be specified. A common choice which I have used is the "not-a-knot" condition which sets the end curvatures to zero.

Augmenting the articulatory position vector with derivatives is similar to the strategy of adding "delta" and "delta-delta" coefficients to an acoustic feature vector when performing speech analysis. The articulatory instantaneous derivatives were found to only slightly improve the accuracy of the estimation and the quality of the reconstructions. The addition of such derivatives makes the forward mapping not quite "instantaneous" since they are computed using position information from a few milliseconds in the past and future. In the audio examples below I have provided examples of synthesis both with and without instantaneous derivatives.

Finally, piecewise linear or locally linear regression models were trained in addition to global linear models. These models used the simple vector quantization algorithm described in chapter 5. Just as with the local PCA analysis, the articulator positions were vector quantized into 32 regions, but now a separate **regression** was performed within each VQ region from articulator positions to line spectral coefficients. Furthermore, the local models used for all the forward acoustic models use **the same** vector quantization regions learned in chapter 5 for modeling the possible articulatory configurations. Each articulatory frame was classified into one of 32 VQ codebooks and regression was performed independently within each codebook. Not surprisingly, using local models significantly improves the quality of the forward model. This is evident from the spectrogram displays and audio examples of section 6.2.3.

Finally, it is important to say a few words about the issue of **testing versus training data**. In all of the examples shown below and audio files presented, the utterance whose acoustics are being predicted was **not** included in the data used to estimate the linear regression matrices. However, the problem is extremely well conditioned and the simple linear models are so constrained (each one has only $16 \times 12 = 192$ parameters) that virtually identical results would be obtained if the test utterances had also been in the training data. Another way of saying this is that removing

a few hundred or one thousand random frames from a linear regression that is based on roughly 2×10^5 frames does not change the results of that regression. As described in chapter 5 only the normal speech tasks from the database were used; the oral motor and repeated sound tasks were excluded from both training and testing. This still leaves over 90% of the original data.

6.2.3 Results

The best way to present results of recovering acoustics is with audio examples. Sounds 6.2.1 to 6.2.11 are examples of utterances resynthesized using articulatory movements. As a complement to these sounds, the two colour plates in figures 6.3 and 6.4 show spectrograms of the original and recovered utterances. Finally, it is also possible to present the results in the form of plots of recovered parameters versus time although such plots are typically less informative than spectrograms and audio examples. Figure 6.2 shows such a plot, illustrating the estimation of two line spectral frequencies. It also shows the estimation of total signal power.

The recovery of signal power is certainly the weakest part of the resynthesis system. While the estimated power is generally correct, occasional periods of energy or silence are completely missed by the model reconstructions. For example at roughly 1.9sec in figure 6.2 there is a burst of energy in the true utterance which the model does not predict. Similarly at 4.2sec there is a silence in the true signal which is absent in the model estimation. Admitting this limitation, I have included a resynthesis example using estimated signal power in sound 6.2.1 only. All other resynthesis examples including the colour plates in figures 6.3 and 6.4 use the *true* energy signal as measured from the original utterance. This is so as not to confuse the difficulty of estimating power with the other aspects of system performance.

Audio file 6.2.1 (`track 8`) *A resynthesis example using estimated power.*

Audio file 6.2.2 (`track 9`) *A resynthesis example using global linear models.*

Audio file 6.2.3 (`track 10`) *Resynthesis of the same example using original acoustic parameters.*

Audio file 6.2.4 (`track 11`) *A resynthesis example using mixtures of local linear models.*

Audio file 6.2.5 (`track 12`) *Resynthesis of the same example using original acoustic parameters.*

Audio file 6.2.6 (`track 13`) *A resynthesis example using mixtures of local linear models.*

Audio file 6.2.7 (`track 14`) *Resynthesis of the same example using original acoustic parameters.*

Audio file 6.2.8 (`track 15`) *Another example with local models.*

Audio file 6.2.9 (`track 16`) *Resynthesis of the same example using original acoustic parameters.*

Audio file 6.2.10 (`track 17`) *A third example with local models.*

Audio file 6.2.11 (`track 18`) *Resynthesis of the same example using original acoustic parameters.*

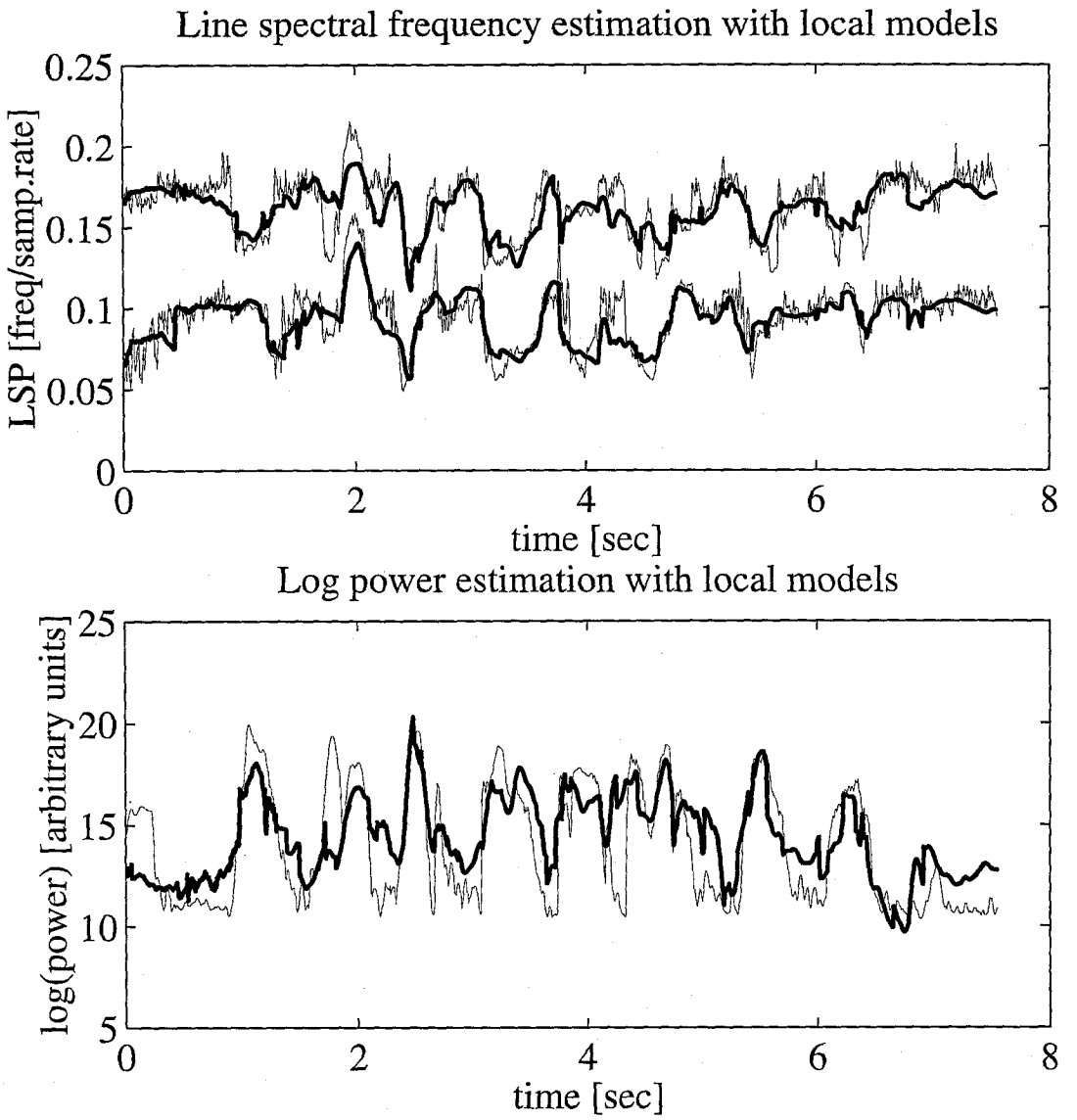


Figure 6.2: An example of estimation of line spectral frequencies and signal energy from articulatory movements. The thin lines show the actual measured LSP frequencies and energy while the thick lines show the predictions made by the models. The inputs to the models were the articulator positions alone. The top panel shows estimation of the third and fifth line spectral frequencies. The bottom panel shows estimation of log of signal energy for the same utterance. These plots show results using a mixture of 32 locally linear models that include instantaneous derivative information as well as instantaneous position information. Estimation of spectral shape parameters is consistently quite successful. Estimated power is usually correct; however, occasional periods of energy or silence are completely missed by the model reconstructions. For example, at roughly 1.9sec there is a burst of energy in the true utterance which the model does not predict; similarly at 4.2sec there is a silence in the true signal which is absent in the model estimation.

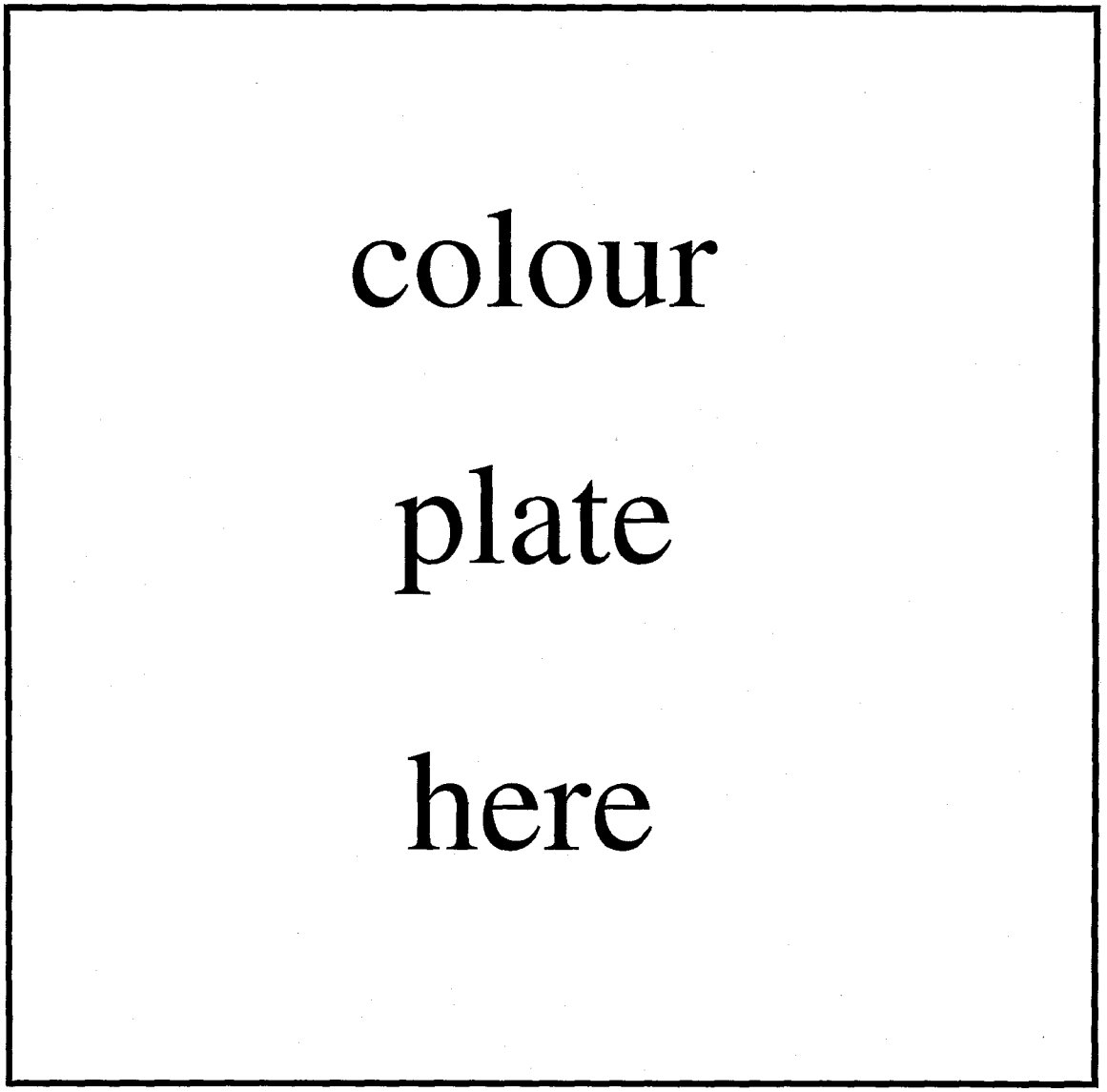


Figure 6.3: Spectrograms of an utterance resynthesized from articulatory movements. Each panel represents signal energy in the time frequency plane. Time runs across the horizontal axis and frequency up the vertical axis. Power in a certain frequency band over a certain time window is encoded using a colour scale which runs from blue (low power) through turquoise, green, yellow to red (high power). The original spectrogram is at the top, followed by spectrograms reconstructed from articulatory movements. The reconstructions use progressively less complex models towards the bottom of the page. All reconstructed spectrograms were generated using spectral shape information inferred from only articulator positions (and derivatives of those positions where indicated). The signal energy from the original utterance has been used. All panels use the same colour scale for power.

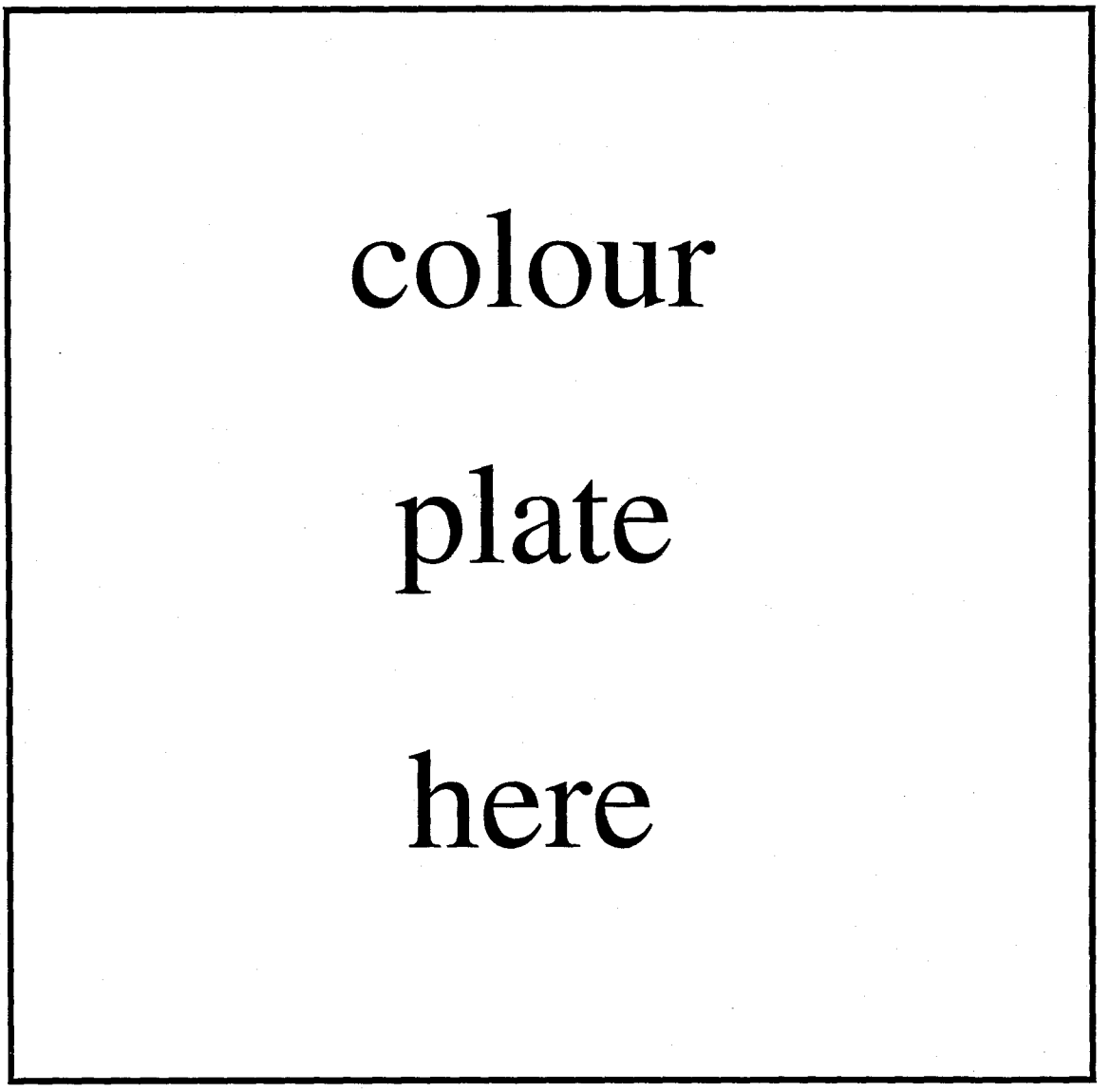


Figure 6.4: Spectrograms of a sequence of s/V/d syllables resynthesized from articulatory movements. Each panel represents signal energy in the time frequency plane. Time runs across the horizontal axis and frequency up the vertical axis. Power in a certain frequency band over a certain time window is encoded using a colour scale which runs from blue (low power) through turquoise, green, yellow to red (high power). The original spectrogram is at the top, followed by spectrograms reconstructed from articulatory movements. The reconstructions use progressively less complex models towards the bottom of the page. All reconstructed spectrograms were generated using spectral shape information inferred from only articulator positions (and derivatives of those positions where indicated). The signal energy from the original utterance has been used. All panels use the same colour scale for power.

6.2.4 Discussion and extensions: estimating voicing and other acoustic features

The models above use articulator positions to predict only two features of the speech analysis window: energy and spectral shape coefficients. Results show that they are relatively successful at predicting spectral shape but sometimes poor at predicting signal energy. In order to increase the utility of the models, improved estimation of the energy signal from the articulatory parameters is needed. The obvious approach is to use nonlinear regression techniques such as radial basis functions or multilayer perceptrons. To a certain extent there is some question about how much of the energy **could possibly be recovered** from the movements alone. However, visual inspection of the energy estimation (as in figure 6.2) indicates that in many cases the linear models are doing quite well. Only at certain brief times are they failing.

However, in speech, spectral shape and energy are certainly not everything. Other features of the audio signal are certainly of interest, for example the voicing state or the presence of plosive bursts. It would be interesting to investigate whether these acoustic features can also be predicted from the articulatory positions. Research along these lines faces an additional hurdle: unlike the energy signal, it is difficult to accurately measure parameters such as voicing from the acoustic waveform in the first place. Thus not only does one have to solve a difficult supervised learning problem—namely predicting this extra information from the articulatory positions—but one also has to worry about getting the information in the first place. In other words, the consistency and noise in the data used for training the system may be a concern.

Chapter 7 Self-Organizing Hidden Markov Models

7.1 Motivation: self-organizing maps

An extremely influential theoretical model in the field of neural computation has been the *self-organizing map* introduced by Kohonen and colleagues [110, 109, 111] in 1982. The first step in Kohonen's essential insight was the observation that *vector quantization*, a well known engineering technique for nonlinear data modeling, could be viewed as a simple competitive learning network. In this view, each prototype vector was associated with a single neuron; when an input pattern was to be encoded, the neuron with the nearest prototype had the highest activity and thus received the learning signal.

Kohonen's proposal was that the competing neurons be endowed with a **topology** defining which ones were close to each other, independent of the prototype vectors they encode. The learning signal was sent not only to the winning unit, but also passed on to its neighbours as defined by this topology. This has the effect of making neighbouring units encode similar prototypes for data, giving a self-organizing map which models in a very nonlinear way a low-dimensional manifold of high-dimensional data.

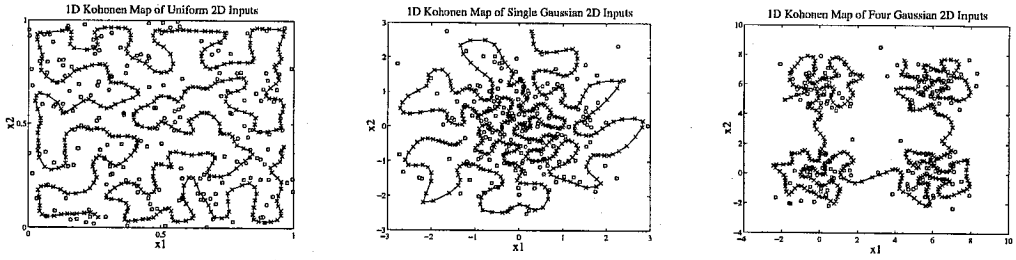


Figure 7.1: Kohonen map with one-dimensional topology (a "snake") trained on two-dimensional data. Circles indicate datapoints; crosses indicate nodes in the snake. Each node is connected to its two neighbours by solid lines.

Kohonen's model and extensions of it do not, however, include any notion of time. When trained on sequences of data, the ordering of the points has no effect (if the training is done carefully) on the outcome of the final map. Below, I introduce a model which is like a Kohonen map through time called, by analogy, a *self-organizing hidden Markov model* (SOHMM). It is a proba-

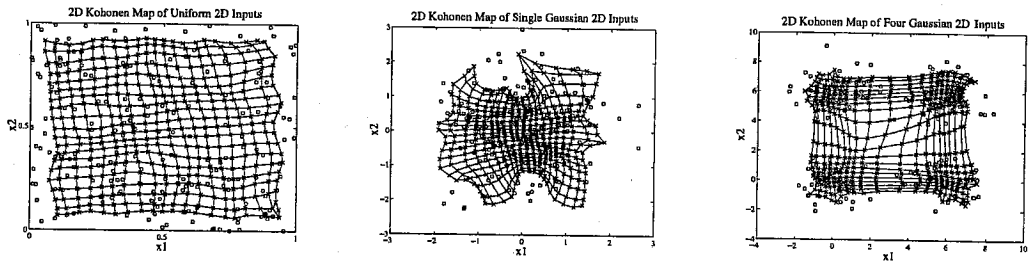


Figure 7.2: Kohonen map with two-dimensional topology (a "sheet") trained on two-dimensional data. Circles indicate datapoints; crosses indicate nodes in the snake. Each node is connected to its four "face-centred" neighbours by solid lines.

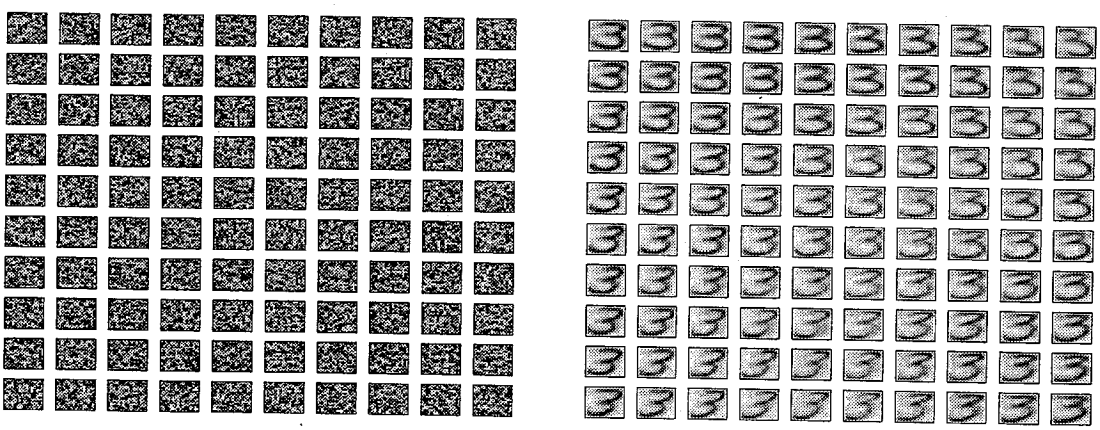


Figure 7.3: Kohonen map with two-dimensional topology (a 10 by 10 "sheet") trained on high-dimensional data. Each datapoint is a complete grayscale image of a handwritten digit 3. The left panel shows the initial condition of the map and the right panel shows the result after training. Similar looking digits occupy nearby cells in the map.

bilistic generative model for continuous or discrete data, similar to a regular hidden Markov model (HMM). However, in the SOHMM, the states of the underlying Markov chain have been endowed with a topology that defines neighbours. This is done by packing the states into a geometric arrangement in a fictitious space called the *topology space* exactly as was done with the neurons in Kohonen's competitive networks.

In the self-organizing map this topology was encoded as a modification to the learning rule of the system: the winning unit **and its neighbours** were updated. In SOHMMs the topology is encoded as a constraint on the system's parameters: the transition matrix of a regular HMM is constrained so that only transitions between neighbouring states are permitted. Thus the only possible state sequences for the underlying Markov chain correspond to connected paths through the topology space. This means **that outputs of the model which occur near each other in time must come from states that are near each other in topology space**. This creates a crucial difference between SOHMMs and Kohonen maps. In a Kohonen map topologically nearby units must encode similar patterns, whereas in a SOHMM nearby units may encode very different patterns—if such patterns often occur close together in time in the training data. Figure 7.4 shows a graphical representation of a state trajectory in a SOHMM with a three-dimensional topology space. Each cell in the large cube is a state in the underlying Markov chain. Transitions are only permitted between neighbouring states, for example "face-centred" neighbours. This means that all possible state sequences in the SOHMM correspond to connected trajectories in the topology cube. The dashed lines show an example state sequence corresponding to a "city-block" path through the cube.

7.1.1 A simple game

Consider playing the following game: divide a sheet of paper into several contiguous, non-overlapping regions which between them cover it entirely. In each region inscribe an integer, allowing integers to be repeated in different regions. Now place a pencil on the sheet and move it around, reading out (in order) the numbers in the regions through which it passes. Add some *noise* to the observation process so that some fraction of the time incorrect numbers are reported in the list instead of the correct ones. The goal of the game is to reconstruct the configuration of regions on the sheet from only such an ordered list of noisy numbers.

At first the task seems quite straightforward. Place the first region arbitrarily. Deduce its neighbours by observing which numbers occur adjacent in the observed sequence. Place them

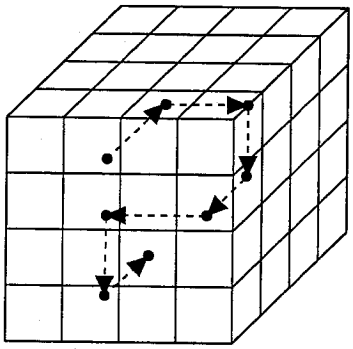


Figure 7.4: A graphical representation of a SOHMM with a three-dimensional topology space. Each cell in the large cube is a state in the underlying Markov chain. Transitions are only permitted between neighbouring states, for example "face-centred" neighbours. This means that all possible state sequences in the SOHMM correspond to connected trajectories in the topology cube. The dashed lines show an example state sequence corresponding to a "city-block" path through the cube.

around the first region, choosing their relative positions by their neighbour relations. This heuristic method can be formalized using *adjacency trees* or *adjacency graphs* from computer science. However, such algorithms fail in the presence of noise and repeated numbers. A possible method for mitigating the noise in the observations is to use some kind of statistical averaging. For example, one could attempt to use the average separation (in time) of two numbers to define a dissimilarity between them by observing which symbols occurred within a nearby window of each other. It then would be possible to use methods like multi-dimensional scaling [170, 206, 192, 193] or Kohonen mapping [111] to explicitly construct a configuration of points obeying these distance relations. However, these methods still cannot deal with repeated numbers in the sheet because by their nature they assign a unique spatial location to each symbol. Self-organizing hidden Markov models solve both of these difficulties and can "win" the game as described above.

Below I give a brief introduction to regular HMMs, and then describe the differences which distinguish SOHMMs. In section 7.4 I show illustrations of the SOHMM algorithm running on synthetic data. Application of the model to real speech production data is discussed in chapter 8.

7.1.2 Latent variable models for dynamic data: hidden Markov models versus linear dynamical systems

Hidden Markov models can be thought of as a dynamic generalization of discrete state models for static data such as vector quantization or Gaussian mixture models. They can also be though of as

discrete state versions of linear dynamical systems (Kalman filter models) which are themselves dynamic generalizations of continuous latent variable models for static data such as factor analysis or principal component analysis (PCA). HMMs have a long history in the speech processing community, while linear dynamical systems have been studied extensively in other engineering fields, especially control theory. Often there has been much parallel work on these topics, much of which is not well shared between the communities (although the review by Roweis and Ghahramani [163] and the book by Elliott [53] explore these and other connections in detail).

While both HMMs and linear dynamical systems provide probabilistic latent variable models for dynamic data, they both have important limitations in time-series modeling. Regular hidden Markov models have a very powerful model of the relationship between the underlying state and the observations because each state stores a private distribution over the output space. However, it is extremely difficult to capture reasonable dynamics on the latent variable because in principle any state is reachable from any other state at any time step and the next state depends only on the current state. Linear dynamical systems, on the other hand, have an extremely impoverished representation of the outputs as a function of the latent variables since this transformation is restricted to be linear. A single matrix captures the state to output mapping and it is applied uniformly regardless of location in the state space. But it is much easier to capture state dynamics since the state is a multidimensional vector of continuous variables on which a matrix "flow" is acting rather than a discrete variable for the state.

In order to improve upon these models, one can either attempt to endow linear dynamical systems with a richer representation of the mapping between states and outputs, or one can address the modeling of state dynamics in HMMs. In the first direction, various generalizations of linear dynamical systems known as *extended Kalman filters*, *switching state-space models* [72] or *switching Kalman filters* [133] have been proposed. (These models often generalize the simple first order linear dynamics as well.) The most basic of these extensions is to use a mixture of linear experts model as the state to observation mapping rather than a single linear transformation matrix. This is equivalent to having a collection of observation matrices as well as a gating function which decides when each one should be used based on the current state. The SOHMM is a step in the second direction. It constrains the HMM parameters so that the discrete state evolves as a discretized version of a continuous multivariate state variable, i.e. it inscribes only connected paths in a topology space. This lends a physical interpretation to the discrete state trajectories in the HMM.

7.2 Hidden Markov models: a brief review

Hidden Markov models (HMMs) are best understood in the context of other latent variable models such as linear-Gaussian dynamical systems (see [163]) or graphical models and belief networks (see [179]). However, I present a relatively self-contained introduction for the reader not familiar with the basics. Although the early journal articles can be confusing (and contain many typographical errors), the excellent review article by Portiz at IDA ([150]) is a notable exception to this generalization. Also, several good texts on speech processing present very readable introductions.

7.2.1 What are HMMs?

Hidden Markov Models are a class of models for mimicking the probability density of a sequence of observed symbols. They are essentially stochastic finite state machines which output a symbol each time they depart from a state. By specifying the state transition probabilities between states and the symbol generation model for each state, we can attempt to capture the underlying structure in a large set of symbol strings.

In general, the operational paradigm is as follows: select a starting state according to some fixed probability distribution. At each time step, generate an output symbol by invoking the generative model of the current state, and then transition to a new state according to a static transition probability matrix.

Let us define some notation for future convenience. Assume there are M possible states s_m in our model, and denote the set of all possible states by $\mathcal{S}$ where $\mathcal{S} = \{s_1, s_2, \ldots, s_M\}$. To this set, we will add a special state s_0 in which the model always starts as an implementational convenience – the use of this state will be explained later. We may choose to use one of the states (usually s_M) as an *end state* – in this case, whenever the model reaches this state, it stops. Let there be P possible output symbols a_p which comprise the output alphabet $\mathcal{A} = \{a_1, a_2, \ldots, a_P\}$. Each state s_j has an output distribution defined by the vector A_j such that the probability of emitting symbol a when we are in state s_j is given by $A_j(a)$. There is also a matrix T of *transition probabilities* defined so that T_{jk} is the probability of moving into state s_k if the model is currently in state s_j.

Of course we require that:

$$\sum_{a \in \mathcal{A}} A_j(a) = 1 \qquad \forall s_j \in \mathcal{S} \tag{7.1}$$

$$\sum_{s_k \in \mathcal{S}} T_{jk} = 1 \qquad \forall s_j \in \mathcal{S} \tag{7.2}$$

in other words, that the rows of T and A must sum to unity (although columns may not). Note also that the diagonal elements of T need not be zero – we allow self transitions.

In order to eliminate the need to store the distribution of starting states separately, we make use of the special state s_0. The model always starts in this state, and never returns there. Hence the entire first column of T is filled with zeros, and the first row of T is just the distribution over the starting states. That is, T_{0j} is the probability that the model will start in the "real" state s_j. Also, no symbol is output when we depart from this state. All this just makes it easier to incorporate the starting state probabilities directly into the workings of the model without having to store them separately and do a special start-up step when we run the model.

Formally, then, we operate the model as follows. Denote the current state by x. Set $x = 0$. Loop until x is the end state or until time runs out: { Generate a symbol a according to probabilities $A_x(a)$. Pick the next state y according to probabilities T_{xy}. Set x=y.}. Figure 7.5 illustrates this generative model.

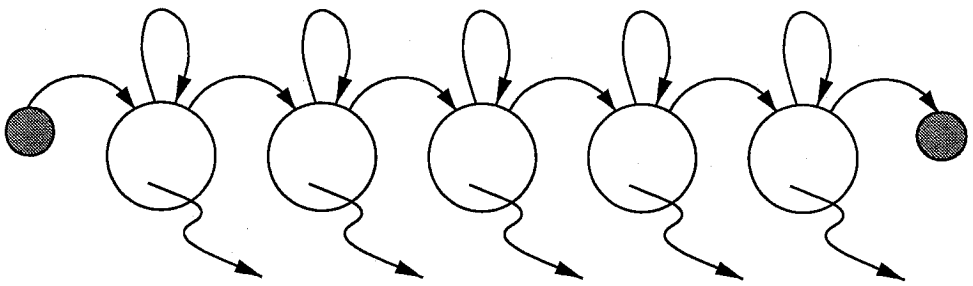


Figure 7.5: Schematic of a hidden Markov model. The model is a finite state machine with discrete states. Each state emits an output and then transitions to another state. In the diagram, states are indicated by circles. Squiggly arrows from the bottom of each state represent output emissions from each state. Arcs from the tops of states indicate possible transitions to other states. Notice that self transitions are permitted. The model shown here has a strictly "left-to-right" transition structure, although this is not in general required.

7.2.2 Using many models to do recognition: likelihood computations

Next, I will describe how to use several HMMs in conjunction to perform a classification or recognition task. Consider: given a large number of sequences each of which have been labelled as belonging to exactly one of several classes, how can we construct a system that tells us to which class an arbitrary new sequence (previously unseen) belongs. In other words, how can we use the examples of the various classes to extract salient features of those classes which we then make use of during recognition.

Hidden Markov models can be applied to this problem as follows: Use all of the sequences labelled as belonging to class C to train a generative HMM for class C. Do this for each class, so that we now have as many HMMs as classes. Now given an arbitrary sequence, one thing we could do would be to let all of the HMMs run freely for some long period of time, observing the sequences that they produced, and then ask which one produced our arbitrary sequence the greatest number of times. The class for which this HMM was the model would be the class into which we classified the arbitrary sequence. Conceptually this is fine, however, in practice this is usually unrealistic because it would take forever to get good estimates. We would like some way to answer the classification question without having to run all our models explicitly and observe them for long periods of time. The answer is to compute the *likelihood* that each model has of generating the given sequence and pick the class whose model has the largest computed likelihood. Fortunately, there is a simple method for doing this computation efficiently.

Consider some sequence of symbols σ which is τ time steps long:

$$\sigma : \sigma_1, \sigma_2, \ldots, \sigma_\tau \tag{7.3}$$

There are many possible state trajectories that could have caused this sequence; conceptually we wish to find all of them and add up all their probabilities. Denote the set of state trajectories which could possibly have caused σ by $I : \{i_1, i_2, \ldots, i_Q\}$. Here, each element i of I is a state trajectory τ states long which represents the states the system visited (in order) while producing the symbol sequence σ. The state visited at any time t in a trajectory i is denoted by $i(t)$. Now define the

likelihood that a particular model Z caused the sequence σ:

$$L^{\sigma}(Z) = Prob[\sigma_1, \sigma_2, \ldots, \sigma_\tau \mid model\ Z] \tag{7.4}$$

$$= \sum_{i \in I} T_{0i(1)} A_{i(1)}(\sigma_1) T_{i(1)i(2)} A_{i(2)}(\sigma_2) \ldots T_{i(\tau-1)i(\tau)} A_{i(\tau)}(\sigma_\tau) \tag{7.5}$$

This seems like it is going to be an extremely hard factorial computation, requiring $O(\tau M^\tau)$ multiplies and $O(M^\tau)$ additions, not including the (non-trivial) amount of work required to compute which sequences should be in the set I. The exponential nature of this cost would make computing many such likelihoods prohibitive, and thus severely limit the practicality of using HMMs as recognizers. Luckily, however, there is an excellent trick which greatly saves on the cost of working this out known as the *forward-backward algorithm.*

Define $\alpha_j(t)$ as the incremental probabilities of having reached state s_j at time t having emitted the correct symbols up till then (including σ_t):

$$\alpha_t^{\sigma}(j) = Prob[\sigma_1, \sigma_2, \ldots, \sigma_t \quad \text{and state at time } t = s_j] \tag{7.6}$$

Now induction comes to our rescue:

$$\alpha_1^{\sigma}(j) = T_{0j} A_j(\sigma_1) \tag{7.7}$$

$$\alpha_{t+1}^{\sigma}(k) = \left(\sum_{s_j \in \mathcal{S}} \alpha_t^{\sigma}(j) T_{jk} \right) A_k(\sigma_{t+1}) \tag{7.8}$$

which enables us to easily compute the desired likelihood $L^{\sigma}(Z)$ since we know we must end in some possible state:

$$L^{\sigma}(Z) = \sum_{s_k \in \mathcal{S}} \alpha_\tau^{\sigma}(k) \tag{7.9}$$

This can be done very cheaply, in only $O(M)$ multiplies and $O(M^2\tau)$ additions, and we do not need to compute the members of the set I.

7.2.3 Training a HMM

We have seen how if a single HMM represents a generative model for a class of sequences that we hope captures some of the structure inherent in the class, then a group of HMMs can be useful

in a recognition task. This involved doing state inference on the hidden states and computing the likelihood that each model generated the observed data. We now need to examine how to update the parameters of a single HMM given some example sequences in order to make it a better generative model.

The standard approach taken is to attempt to maximize the likelihood of the training data under the model; the idea is to update the parameters so that the model is more likely to generate the example sequences we have. This can either be done in an online way as we get each output in the time series, or in batch mode – processing the whole sequence at once. Batch training is much more common and easier to do. Usually in speech applications training occurs during the design of the system but not during its operation so online methods for parameter updating are not essential.

Ideally, we would like to find the *maximum likelihood* model parameters directly, but there is currently no know way to do this efficiently. (The problem of finding the optimal weights for a neural network has been proven by Judd [100, 101] to be NP complete, but in contrast it is still possible that there exists an efficient maximum likelihood algorithm for mixture models.) However, the well known EM algorithm [43] can be used to find the local maximum in likelihood closest to our initial parameters. In the context of hidden Markov models, the EM algorithm is called the *Baum-Welch* algorithm [12, 13] and it gives us a prescription for changing the parameters that always increases the likelihood of the model or leaves it constant.

The algorithm has an estimation step and a maximization step which are just the E and M steps in EM. The E step uses the forward-backward algorithm described above to estimate the hidden state occupations while the M step computes sufficient statistics for the transition and emission probabilities using the state estimates from the E step. The algorithm works as follows. For each example sequence σ, do the following:

- **E-step: forward-backward algorithm**
  - **Forward pass**:

    For each state s_j at each time t, compute the incremental probabilities $\alpha_t^\sigma(j)$ as explained above.
  - **Backward pass**:

    For each state s_j at each time t, compute $\beta_t^\sigma(j)$ as the probability of starting in state

s_j at time t and emitting the correct symbols after that point (including σ_t):

$$\beta_t^\sigma(j) = Prob[\sigma_t, \sigma_{t+1}, \ldots, \sigma_\tau \quad \text{and state at time } t = s_j] \tag{7.10}$$

which can once again be computed with a nice induction:

$$\beta_\tau^\sigma(j) = A_j(\sigma_\tau) \quad \forall s_j \in \mathcal{S} \tag{7.11}$$

$$\beta_t^\sigma(k) = A_k(\sigma_t) \sum_{s_j \in \mathcal{S}} T_{kj} \beta_{t+1}^\sigma(j) \tag{7.12}$$

- **M-step: update transition and emission parameters**

  - **Transition matrix**:

    First compute $\gamma_t^\sigma(i,j)$ which is the expected number of transitions from state s_i to s_j that begin at time t (given the model Z and the example sequence σ):

$$\gamma_t^\sigma(i,j) = \frac{\alpha_t^\sigma(i) T_{ij} \beta_{t+1}^\sigma(j)}{L^\sigma(Z)} \tag{7.13}$$

    The total expected number of state transitions from state s_i to state s_j, n_{ij} for the particular example σ is given by the sum over all starting times:

$$n_{ij}^\sigma = \sum_{t=1}^{\tau} \gamma_t^\sigma(i,j) \tag{7.14}$$

    Finally, compute $\gamma_t^\sigma(j)$ which is the probability that the model is in state s_j at time t (given the model Z and the example sequence σ):

$$\gamma_t^\sigma(j) = \frac{\alpha_t^\sigma(j) \beta_t^\sigma(j)}{A_j(\sigma_t) L^\sigma(Z)} \tag{7.15}$$

    The new transition parameters (including initial state occupation) are obtained by taking the averages over all example sequences σ of each of the following quantities:

$$T'_{0j} = \gamma_1(j) \tag{7.16}$$

$$T'_{ij} = n_{ij} \Big/ \sum_{s_j \in \mathcal{S}} n_{ij} \tag{7.17}$$

recalling that T_{0j} are the probabilities of beginning in state s_j.

- **Emission probabilities**:

  The emission probabilities can be updated by summing over all times at which a particular output symbol appeared. Again, averages over all example sequences σ should be taken:

$$A'_j(a) = \frac{\sum_{t|\sigma_t=a} \gamma_t(j)}{\sum_{t=1}^{\tau} \gamma_t(j)} \tag{7.18}$$

Each of these update rules has a nice interpretation. The new value for an initial state probability T_{0j} is just the average probability that the model will be in the state s_j at $t = 1$, the first time step. The new transition probability T_{ij} is simply the ratio of the expected number of state transitions from s_i to s_j to the expected total number of transitions out of state s_i. The new probability $A_j(a)$ of generating the symbol a when leaving state s_j is just the ratio of the probability that the model was in state s_j *and* generated the symbol a to the total probability that the model was in state s_j.

Baum, Pietrie, Soules and Weiss [13] have proven a theorem (originally proposed by Baum and Eagon in [11]) which guarantees that following this procedure will *increase* the likelihood of the model generating the example sequences (unless the current model defined a critical point of the likelihood function in which case the likelihood stays the same but does not decrease). That is, $\langle L^\sigma(Z') \rangle$ is guaranteed to be greater than $\langle L^\sigma(Z) \rangle$ where the angled brackets denote averages over all example sequences.

7.2.4 State inference

Notice that the above algorithm computes, as an intermediate in the likelihood computation, quantities which estimate the probability of being in each hidden state at each timestep. This is known as *state inference* and is an essential step in computations with all hidden variable models. The sequence of states defined by choosing the maximum value of γ is the sequence which maximizes the expected number of correct states and corresponds roughly to Kalman smoothing (see Appendix B) in a continuous state linear system. However, as mentioned in that same chapter this "maximum-at-each-time" sequence may not itself be a very probable state trajectory. In fact, it

may correspond to an impossible trajectory because it may move between states that are linked with zero transition probabilities. A dynamic program [18] can be used to find the single state sequence most likely to have generated given observations. This dynamic programming is also known as *Viterbi decoding* [199] and is often used to explain observations although is is rarely used to train models.

7.2.5 Extending output models

In all of the discussion above, I have assumed that the values of $A_j(a)$ were the actual probabilities of generating the discrete symbol a when leaving state s_j. However, it is possible for each state to use a more complex output generation model. For example, continuous values can be used. This means that each state needs to have a probability distribution over the scalar or vector space in which the outputs live. The simplest such model is a single Gaussian density function. In fact, we can think of a mixture of Gaussians model as a very simple HMM that has as many states as Gaussians in the model. The model starts in state 0, goes to one of the states, generates a data point according to the private Gaussian parameters of that state, and then returns to the start state. The initial state probabilities T_{0j} are just the mixing coefficients of the Gaussian submodels, while the mean and variance of each Gaussian are just the model parameters A for each state. More complex distributions for each state are also possible. For example, a common practice in speech processing [] is to have an entire feed-forward neural network as the generation model in each HMM state. Another common choice is to have a mixture of Gaussians for each state. (Do not confuse this with the analogy drawn above.) Each state has its own private values for the parameters of these model (for example, the means and variances or the network weights) which are used whenever that state must generate a symbol. While this makes HMMs potentially much more powerful, it also makes them much more difficult to work with. To answer the classification question: "Which model is most likely to have generated this sequence?," we still must be able to compute the individual symbol generation probabilities $A_j(a)$ from these more complex models. Even worse than this, updating our model parameters given some example sequences now becomes very difficult since we must backpropagate through each state's generative model or fit a mixture during each step of Baum-Welch.

7.3 Self-organizing hidden Markov models

7.3.1 Model definition: state topologies from cell packings

The first step in defining a SOHMM is to identify each state of the underlying Markov chain with a cell in a packing of a Euclidean space called the *topology space*. This involves (1) selecting a dimensionality d for the topology space and (2) choosing a packing rule for that space (such as hexagonal or cubic). (The second step is unnecessary if the topology space is one dimensional since there is only a single possible packing.) The number of cells in the packing is determined by the number of states M in the Markov model. Cells are taken to be of equal size. Furthermore, since the scale of the topology space is completely arbitrary the cells are taken to be of unit size; this means that the packing covers a volume M in topology space with a side length of

$$\ell = M^{1/d} \tag{7.19}$$

The dimensionality and packing together define a vector valued function $\mathbf{x}[m]$, $m = 1 \ldots M$ which gives the location of the centre of cell m in the packing. For example, a cubic packing of d dimensional space defines $\mathbf{x}[m]$ to be:

$$\mathbf{x}[m] = \left(m-1, (m-1)/\ell, (m-1)/\ell^2, \ldots, (m-1)/\ell^{d-1}\right) \mod \ell \tag{7.20}$$

where $a \mod b$ denotes the remainder after dividing a by b.

The next step is to choose a neighbourhood rule in the topology space. This rule defines the neighbours of cell m based on $\mathbf{x}[m]$. For example, the neighbours of cell m might be all connected cells (those which "touch" m in topology space). These are all cells ν for which $\mathbf{x}_i[\nu] = \mathbf{x}_i[m] + \alpha_i$ with $\alpha_i \in -1, 0, +1$. A more restrictive rule would be to consider only face-centred neighbours. This can be achieved by adding the restriction that exactly one of the α_i above is nonzero; excluding edge and corner neighbours from the connected set above. For cubic packings, there are $3^d - 1$ connected neighbours and $2d$ face-centred neighbours in a d dimensional topology space. The neighbourhood rule also defines the boundary conditions of the space—periodic boundary conditions make cells on opposite extreme faces of the space neighbours with each other.

We assign state m in the Markov model to cell m in the packing, thus giving it a location $\mathbf{x}[m]$ in the topology space. The transition matrix of the SOHMM is **preprogrammed** to only allow

transitions between neighbours as dictated by the neighbourhood rule. Transitions to neighbours may be uniformly likely, or other rules (e.g. the momentum rule described below) may be used. All other transition probabilities are set to zero. This means that all valid state sequences in the underlying Markov model represent connected ("city block") paths through the topology space.

7.3.2 State inference and learning rule

The SOHMM has exactly the same inference procedures as a regular HMM: the *forward-backward algorithm* for computing the state probabilities at any single time and the *Viterbi decoder* (described above and in Appendix B) for finding the single best state sequence remain unchanged. Once these discrete state codings have been performed, they can be transformed to yield probability distributions over the topology space (in the case of forward-backward) or paths through the topology space (in the case of Viterbi decoding) using the state position function $\mathbf{x}[m]$. This transformation makes the outputs of state decoding in SOHMMs equivalent to the outputs of inference procedures for dynamical systems such as Kalman filtering or smoothing.

The learning procedure for SOHMMs is also almost identical to that for HMMs. In particular, the EM algorithm (called the Baum-Welch algorithm in the context of HMMs) is used to update model parameters. The crucial difference is that the transition probabilities T_{ij} which were set by the topology and packing are never updated during learning. (In particular, this means that the update in equation (7.17) should not be performed.) In fact, the preprogrammed transition matrix makes the learning much easier in some cases. (This statement is quantified in the following sections by comparing structure discovery using SOHMMs with regular HMMs on data which is generated from systems having spatial dynamics.) Not only do the transition probabilities not have to be learned, but their structure constrains the state sequences in such a way as to make the learning of the output parameters much more efficient in cases where the underlying data really does come from a spatially structured generative model. But even with these advantages learning can be tricky. Because the Baum-Welch training guarantees convergence only to a local maximum of likelihood, initialization of the model's output parameters has a significant effect on the final model even when training with a fixed dataset. Figure 7.6 illustrates this local maximum problem. In section 7.5 below I will discuss a hierarchical training approach to initializing model parameters and show that it can lead to much faster learning and avoidance of local minima.

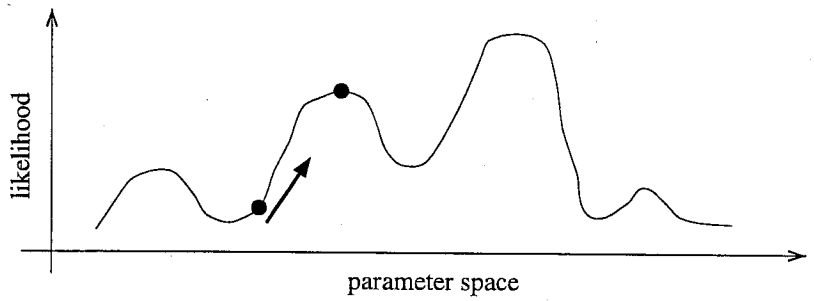
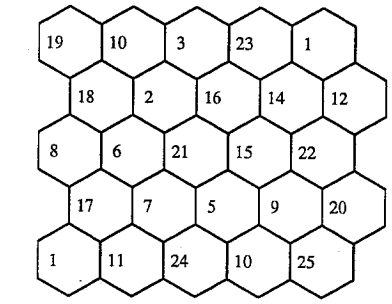


Figure 7.6: Local maxima in model learning. The Baum-Welch re-estimation procedure, like all EM algorithms, only finds parameter settings corresponding to a local maximum in likelihood. The final model is thus highly dependent on the initial condition chosen. Several heuristics are discussed in the text for selecting good initializations for SOHMMs.

7.4 Synthetic data examples

To illustrate learning and inference in self-organizing hidden Markov models and to demonstrate the training heuristics described in section 7.5, I have generated data from synthetic maps in two dimensions. The generative maps are either square or hexagonal griddings of the plane. In each grid cell there is a number which represents the output produced by that cell in the map. Data is generated by random walks in the maps, moving only to adjacent cells. (In the square grids I have allowed transitions to face and corner neighbours.) Self-transitions are also possible. Notice also that the output numbers are repeated in the maps so that more than one cell may generate the same output. Furthermore, in all of the experiments shown there is a 15% noise level in the output symbols. This means that 15% of the output symbols are destroyed and replaced with random noise (a symbol chosen uniformly from the set of all symbols). Figure 7.7 shows a hexagonal map and a sequence of data generated from it. Symbols which represent noise (errors) are circled for clarity, although of course the SOHMM algorithm does not have this information during training. Figure 7.8 shows the training of this hexagonal grid based on a noisy sequence of 1000 symbols (15% noise level). Each panel represents a snapshot of the model during training. The model parameters are displayed using the following convection: the numbers of the three most probable symbols in each state's output histogram are typed in the cell corresponding to that state. Font size is proportional to square root of probability so that the area of each number roughly corresponds to its probability mass in the histogram. Iterations and log likelihoods are reported at the bottom of each panel. For synthetic data the algorithm works quite well—the final models are all almost exactly rotation of the true maps.



1, 11, 1, 11,...
24, (2), 21, 2,...
18, 19, 10, 3,...
2, (2), 16, 16,...
15, 15, 2, (3),...

Figure 7.7: A generating map and an example sequence of data produced by the map. Symbols which represent noise (errors) are circled for clarity, although the SOHMM algorithm does not have this information during training.

7.5 An hierarchical training strategy: upsample training

The learning rule for SOHMMs climbs to a local maximum of likelihood in parameter space. The final model is thus highly dependent on the initial condition chosen as well as on the training data. One possible strategy for combating this difficulty is to embed the training procedure in a loop that samples the model landscape by restarting training at many random initial conditions. The model that ends up in the best maximum (as measured by likelihood of the training data) is the result of this iterative compound procedure. Unfortunately this search can be extremely slow and may not yield good solutions in a reasonable time. Several strategies are possible to overcome this difficulty. One is to use extra information or computation to generate an initialization that is significantly better than random parameter selection. For example, such an initialization may come from prior knowledge about the problem structure or from training a simpler or approximate model. This idea of training a simpler model has led to a heuristic hierarchical training procedure which I employ when training these models. The procedure, called *upsample training* involves first training a SOHMM with a small number of states and then gradually increasing the size of the model. At each iteration, the trained model from the previous stage is *upsampled* onto the larger grid of the new model. This upsampled model is taken to be the initial condition for training the larger model. Once convergence at a particular size has occurred, the size is increased in the same way. This iteration of upsampling and retraining continues until the final desired size of SOHMM is achieved. The training is always initialized with a single-state model. Training this single-state model is trivial: it merely involves constructing a normalized histogram of the output

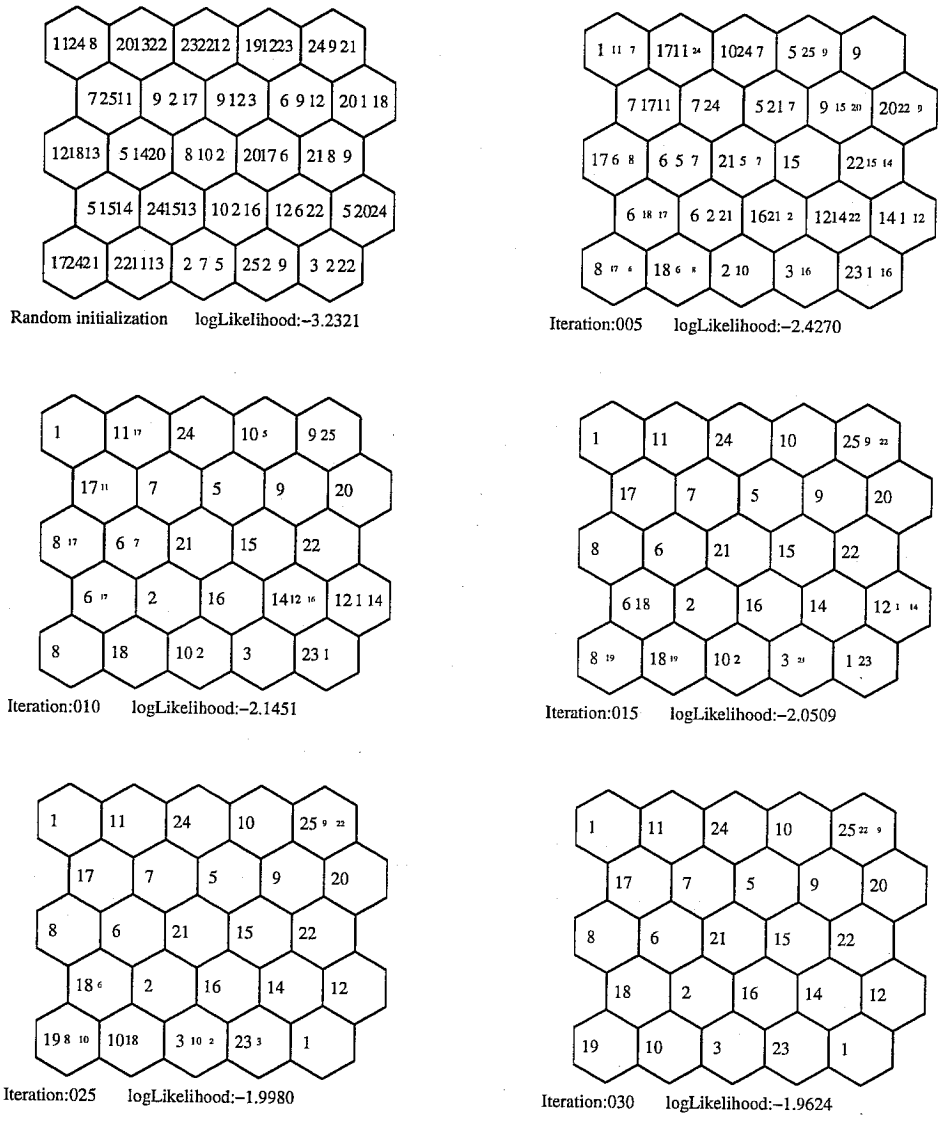


Figure 7.8: Training a hexagonal grid SOHMM. Each panel represents a snapshot of the model during training. The numbers of the three most probable symbols in each state's output histogram are typed in the cell corresponding to that state. Font size is proportional to square root of probability so that the area of each number roughly corresponds to its probability mass in the histogram. Iterations and log likelihoods are reported at the bottom of each panel. The algorithm was trained using a sequence of 250 points with 15% noise. The final model is almost exactly a rotation of the true map shown in figure 7.7.

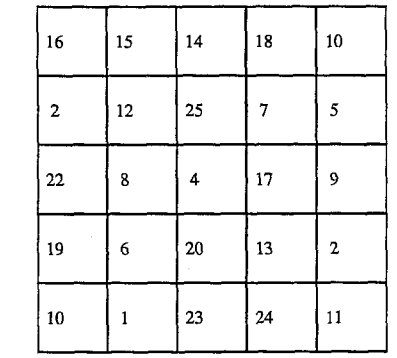


Figure 7.9: A square grid generating map.

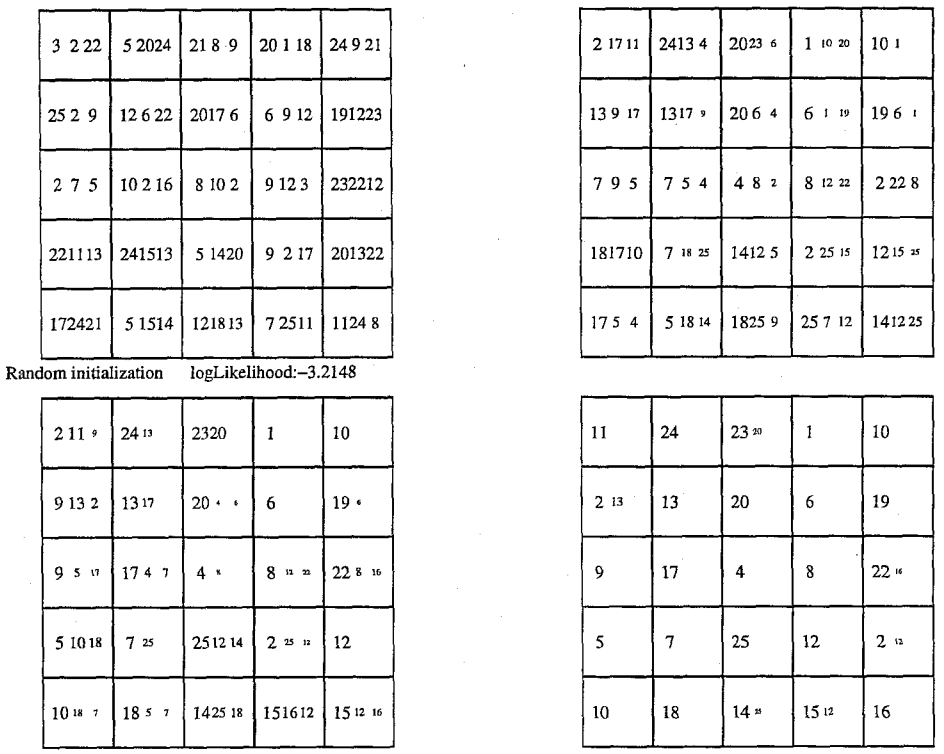


Figure 7.10: Training a square grid SOHMM. Each panel represents a snapshot of the model during training. The numbers of the three most probable symbols in each state's output histogram are typed in the cell corresponding to that state. Font size is proportional to square root of probability so that the area of each number roughly corresponds to its probability mass in the histogram. Iterations and log likelihoods are reported at the bottom of each panel. The algorithm was trained using a sequence of 250 points with 15% noise. The final model is almost exactly a rotation of the true map shown in figure 7.9.

symbols and assigning this distribution to the output parameters of the model. In the continuous output case, the single state must just do a global fit of its emission distribution (for example a Gaussian or mixture of Gaussians) to all the available data.

As an example, consider a SOHMM with a two-dimensional topology; i.e. , the states are arranged in a sheet. The outputs of the model are taken to be discrete symbols. Upsample training begins with a single state. A histogram of the output symbols is collected, and this distribution is assigned to the state. This initial model is *upsampled* or *shattered* into a 2×2 SOHMM sheet with four states. Each of the four daughter states inherits its initial output distribution from its parent state (in this case the single initial state). The transition probabilities are assigned by the topology and packing as usual. This new, larger model is then trained on the available data using the standard training procedure—namely Baum-Welch except without updates to the transition parameters. Once training has converged, the model is again upsampled or shattered. Fast growth can be achieved by quadrupling the number of states at each stage. In this case each parent state generates exactly four daughters which inherit its output distribution parameters exactly. However, slower growth may be required in which case each daughter cell in the new model has several *partial parents* determined by its overlap with those cells in topology space. The output distribution for the daughter cell is set to be a (renormalized) weighted combination of the distributions of its parents. Figure 7.11 illustrates this hierarchical training on the same sequence as was used in figure 7.10. The model quickly learns the correct structure. The total iterations needed are slightly more than a single run of direct training but notice that there is no random initialization present in this procedure. Every upsample training run will yield this same result, whereas with direct training we must try many initial conditions until a satisfactory local maximum is found.

This hierarchical approach can also be used across dimension. Typically it begins with a two-dimensional SOHMM "sheet" of the same edge size as the desired final map. Once the sheet has been trained, it is copied and stacked into a "cube" (three dimensional) SOHMM. This SOHMM is then retrained until convergence and then stacked to yield the model of next higher dimension. The procedure is repeated until the desired dimension is achieved.

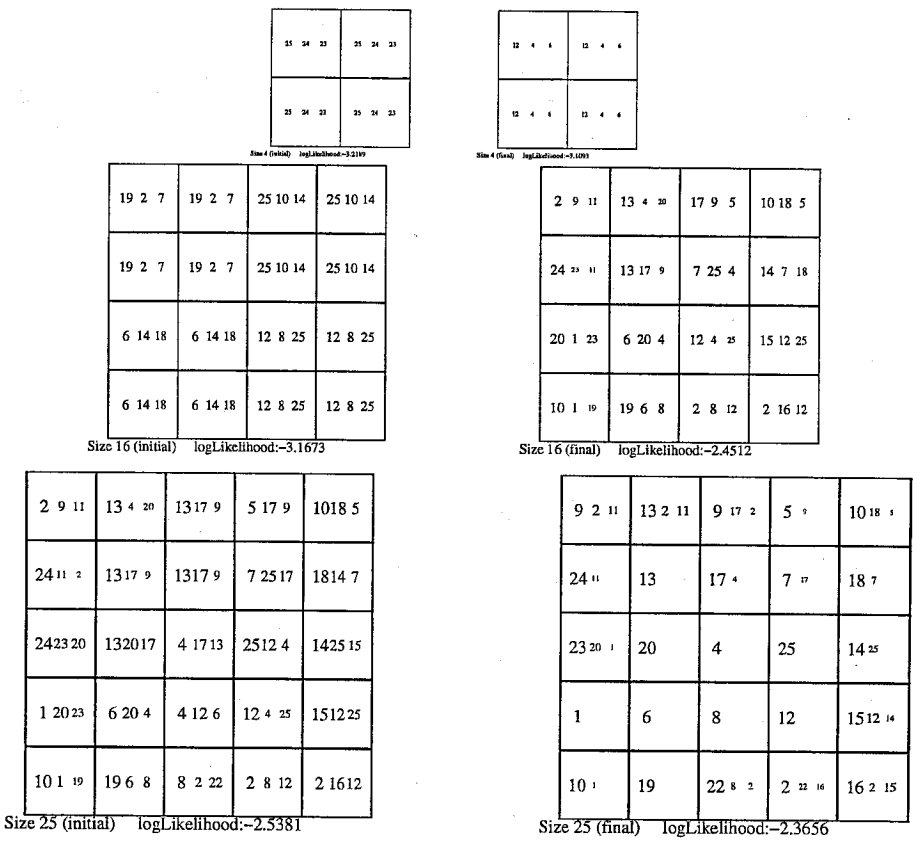


Figure 7.11: Hierarchical training of a SOHMM using *upsample training*. Upsample training begins with a single state. A histogram of the output symbols is collected, and this distribution is assigned to the state. This initial model is *upsampled* or *shattered* into a 2×2 SOHMM sheet with four states. Each of the four daughter states inherits its initial output distribution from its parent state (in this case the single initial state). The transition probabilities are assigned by the topology and packing as usual. This new, larger model is then trained on the available data using the standard training procedure—namely Baum-Welch except without updates to the transition parameters. Once training has converged, the model is again upsampled or shattered. This hierarchical approach can also be used across dimension. Although all four states in the top right panel appear to have the same output distribution they are in fact slightly different.

7.6 Application to articulatory speech processing

How can the SOHMM model be applied to articulatory speech processing? Firstly, the low-dimensional topology space in which we are learning trajectories (recall the sheet of paper in our game) represents an abstract articulatory space—a state space for a speech production system. The coordinates in this state space need not be directly equivalent to the measured positions of any specific articulators. (In fact the model is completely invariant to all orthogonal transformations such as rotations and axis aligned scalings.) Rather, the state (topology) space ought to span the same manifold in acoustic space as a physically based articulatory model would. For example, the topology space might implicitly encode the first few principal components of articulator positions or some other small-degrees-of-freedom representation. Secondly, the output symbols over which each cell in the sheet has a probability distribution represent the spectral shapes of the sounds that are produced when the articulators are in the configuration specified by a cell's location. (In our game this corresponded to the choice of number in each region.) These outputs might be discrete symbols (if we vector quantize the spectral features) or continuous valued vectors (such as the line spectral frequencies).

From this viewpoint, playing the game outlined above is akin to pursuing an *acoustic to articulatory mapping*: we perform short-time spectral analysis on an incoming acoustic signal (and possibly classify each short-time spectrum into one of a finite number of categories) and then attempt to reconstruct the organization of the articulatory space based only on the observed sequences of acoustics. Once we have learned a model we can perform state inference to decode the state trajectories associated with an individual utterance. Finally, we can attempt to relate this decoded state trajectory to the actual articulator movements. Alternately, if articulatory recovery is not the goal, the decoded state trajectories themselves may be used directly for recognition or speaker identification.

By allowing symbols to be repeated in the map, we are recognizing that there may be several *different* articulatory configurations all of which produce very *similar* spectral patterns. This ambiguity is an important feature of any model since the results of chapter 8 show that this phenomenon is a common occurrence in normal English speech.

Chapter 8 discusses acoustic to articulatory inversion in detail and in particular the application of SOHMMs to this problem. As discussed above, initialization is a serious difficulty and was addressed by application of the upsample training heuristic.

7.7 Other applications of SOHMMs: compositional coding, surface mapping, motion tracking

7.7.1 Compositional coding

Consider a new representation of a continuous signal $y(t)$ in the form:

$$y(t) = a(s(t)) + \text{noise} \tag{7.21}$$

where the function $a(s)$ is the state-to-output function or merely the *output function* and the signal $s(t)$ is the *state trajectory*. I will call this representation *compositional coding*. The idea behind compositional coding is that $y(t)$ is "generated" by the trajectory of a hidden state as observed through a fixed output function. As with other generative probabilistic models, two questions of interest are *state inference* and *learning*.

It turns out that the state inference problem is tractable for almost all reasonable output functions $a(s)$ and noise sources. Given $y(t)$, $a(s)$, and a probability distribution on the noise signal, the optimal (maximum likelihood) trajectory $s(t)$ is given by the *Hamilton-Jacobi-Bellman flow* [19] which is a continuous time analog of Bellman's dynamic programming algorithm.

The central question is then one of learning the output function $a(s)$. The model is powerful enough that maximum likelihood learning will not suffice, since many solutions that fit the data exactly are possible and hence overfitting will be a serious problem. For example, consider the trivial "memorization" solutions:

$$s(t) = y(t), \quad a(s) = s \tag{7.22}$$

$$s(t) = t, \quad a(s) = s(t) \tag{7.23}$$

in which $y(t)$ is encoded directly into either the state trajectory or the output function. Other memorizations are also possible, for example[1] in the frequency domain:

$$a(s) = \sum_{k=0}^{\infty} a_k s^k \qquad s(t) = e^{-jt} \tag{7.24}$$

where we have set the state trajectory to the unit velocity helix in the complex-plane vs. time space

[1] Thanks to John Hopfield for suggesting this example.

and where the output function $a(s)$ has a polynomial representation. The polynomial coefficients are just the complex Fourier series coefficients for the signal $y(t)$.

The goal of learning, then, must be to optimize some cost functions on $a(s)$ and $s(t)$. One approach which motivates the term "coding" is to search for a minimum description length (MDL) [160, 161] model of the original signal $y(t)$. That is, given prior probability distributions over the functions $a(s)$ and $s(t)$ and the noise, minimize the total number of bits required to reconstruct $y(t)$. This can be formalized in several ways. One possibility is to discretize both time and the output signal amplitude. In this case the continuous signal $y(t)$ becomes a discrete sequence y_t of integers representing the quantization level at each discrete time. If there are K amplitude levels and T time steps, we can trivially encode y_t using $T \log K$ bits. We can discretize the range of the output function using the same discretization as was used for the range of y and also discretize the state trajectory (domain of a) so that the output function $a(s)$ now becomes another sequence a_s of integers. The signal can then be transmitted by first sending a_s then sending the discrete state sequence s_t and finally sending the noise terms n_t which correct any outputs that the compositional coder did not properly produce. The goal of learning is to minimize the total number of bits required to communicate y_t in this way. This goal encourages the state trajectory to be "smooth" (so that the difference sequence has low entropy) and the output function to be "short"(compact support) to keep coding costs down. Another way to formalize the communication argument[2] is to consider a *sampling* perspective. Imagine that the original continuous signal $y(t)$ had a spectrum bandlimited below some cutoff frequency ω_{max}. The signal can be exactly reconstructed by the values of discrete amplitude samples taken evenly in time as long as the samples are taken above the Nyquist rate of $2\omega_{max}$. Thus, for a signal of total length τ in time, a naive sampling approach requires $2\omega_{max}\tau$ samples. However, it might be possible to design $a(s)$ and $s(t)$ so that $s(t)$ is bandlimited at a much lower frequency than $y(t)$. The goal of learning in this case would be to minimize the total number of samples required to send $a(s)$, $s(t)$ and the error signal or noise.

This problem can be approached directly using a SOHMM. Construct a SOHMM with a one-dimensional state topology and continuous outputs. Each state should have an output distribution equal to the distribution of the noise n but with an arbitrary mean. This is equivalent to discretizing the state s and allowing only transitions up or down by one quantization level. In other words, s_{t+1} must be either at the same level as s_t or at one level higher or lower. The signal y_t (discretized in time but continuous in amplitude) becomes the observation sequence used for training. The

[2]Thanks to Erik Winfree for a discussion which lead to this idea.

SOHMM learning procedure attempts to set the means of each state's output distribution so as to maximize the likelihood of the observed data given the constraints. This has the effect of discovering an output function a_s such that the observation can be coded in the form $y_t = a_{s_t} + n_t$. Furthermore, s_t is constrained to being a sequence with consecutive differences no greater than 1.

7.7.2 Surface learning

Consider discrete time data produced by an altitude sensor that repeatedly traverses an unknown surface at roughly constant speed. Such data may come from, for example, a mobile robot exploring a limited environment that samples its height at regular time intervals. The path (map trajectory) taken by the sensor is unknown; all that is given is the one dimensional signal of its height over time. Assume that the unknown path crosses back upon itself many times so that the surface has been well covered. The goal is to generate a map of the surface and also to recover the map trajectory of the sensor (robot).

The intuition for why this problem might be solvable is the following: if the altitude signal decreases (increases) from one time step to the next and then decreases (increases) **again** in the next time step it is certain that the sensor did not move backwards (otherwise the altitude would have returned to the same value). It is this weak constraint of not having moved backwards that over time can might be used to infer the trajectory and surface structure.

To solve this problem using a SOHMM, we would construct a two-dimensional state topology. Each state would have a univariate continuous output representing the height of the surface at that point. Since the sensor is constrained to move continuously over the surface, its path corresponds to some valid state sequence in the two-dimensional SOHMM. The learning and inference procedures described above can be applied with minor modifications to account for the continuous valued rather than discrete valued output at each state.

7.7.3 Motion tracking

As a final example application of SOHMMs, consider the problem of motion tracking in a computer vision system. In a typical motion tracking system, an articulated body (for example a human being) is instrumented with a few discrete sensors. The sensors are usually brightly coloured or light emitting dots attached to key points such as joints. During motion of the body a camera tracks the planar projections of these sensors. The goal is to estimate the motion of the underlying

body using only the images recorded by the camera.

Since most articulated bodies actually have a small number of degrees of freedom, this problem can be directly mapped into SOHMM learning. The underlying states represent configurations of the articulated bodies. Each state has an output distribution which describes what the camera will see when the body is in that state. The state evolution sequences must correspond to a smooth connected path in some topology space because of the physical structure of the system. The camera images are like the output sequences of the model. Learning can be performed directly on a large database of motion sequences using the techniques described above. There is no need for hand segmented training data. Once the model has been learned, motion tracking simply involves state inference in the SOHMM given the observation sequence.

Chapter 8 Acoustic to Articulatory Inversion

8.1 Can one hear the shape of the mouth?

One of the most famous and beautifully written papers in mathematical physics is the 1966 work by Mark Kac[1] entitled: *Can one hear the shape of a drum?* Kac considers the question of whether the shape of a vibrating membrane clamped at its boundary can be deduced from the spectra of the eigenmodes of vibrations that it supports. As L. Bers pointed out, this is, loosely speaking, equivalent to asking if it is possible to hear the shape of a drum. As it turns out, the answer to this question **in the absence of prior knowledge about the shape of the drumhead** is negative. Although it is possible to hear the perimeter and area of a drumhead, it was shown many years later (in for example [78]) that there exist drum shapes which are isospectral but not isometric. An example of two such shapes is shown in figure 8.1.

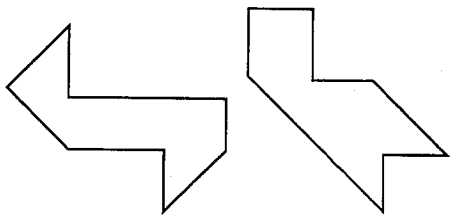


Figure 8.1: Two membrane shapes which are isospectral in the eigenmodes of vibrations they support but which are not isometric. This is an example of a case in which one cannot "hear the shape of a drum." (After Gordon et al.)

Kac's monograph is concerned mainly with mathematical questions about the eigenvalues of the Laplacian of certain domains in the Euclidean plane. Nonetheless, this intriguing investigation into the relationship between geometry and spectral information is the intellectual precursor to a question central to the present work: *Can one hear the shape of the mouth?* In other words, can the movements of a speaker's articulators be inferred from the speech wave alone? That is the topic which will be addressed in this chapter.

[1]Pronounced "Katz."

8.1.1 The inverse mapping problem: possible but not trivial

The problem of acoustic to articulatory inversion is a long standing one in the field of speech science. The fundamental question is whether the movements of the mouth, lips, tongue and jaw can be deduced from an analysis of the acoustic signal produced **during normal speech**. The final caveat is a crucial one since it is well known that there are several situations in which this inversion is impossible. Some examples:

- Articulator movements during silences cannot be tracked since the acoustic signal provides no information during these periods.
- Ventrilloquism effects (see for example [26]) indicate that with enough practice and effort, speakers can learn to produce almost normal sounding speech for a wide repertoire of sounds with little or no lip and jaw movement.
- A sufficiently long and richly varying section of speech is required for inversion to be possible. "Instantaneous" spectra (based on less than about a hundred milliseconds of speech) and certain steady state sounds cannot be inverted to a unique articulatory position (see section 8.2 below).

However, for normal speech it is widely believed that articulator movements *can* be recovered from speech. A thought experiment which strongly motivates but does not prove this belief is the **Million Dollar Recovery Machine**. The machine (shown in figure 8.2) takes as input an acoustic signal and one million dollars in cash and produces as output the movements of the speaker's articulators. It produces, in almost every case when the acoustic signal consists of normal speech, excellent recovery of the articulator movements. Unfortunately, the machine is an elaborate ruse. Inside the machine is the same speaker who spoke the original utterance (flown to the site if necessary). The speaker is given the one million dollars in cash and is told to listen very carefully to the acoustic signal and to repeat as exactly as possible what was said in the original recording. While they are repeating the utterance a second time, a measurement mechanism (such as the X-ray microbeam facility described in section 3.1.1) records the movements of their articulators and sends these as the output of the machine.

The machine is subject to two important criticisms. First, it only performs *speaker-dependent* inversion in the sense that it requires specialized knowledge about the speaker (in the form of the speaker herself) to recover movements from acoustics. Second, it might be argued that the machine

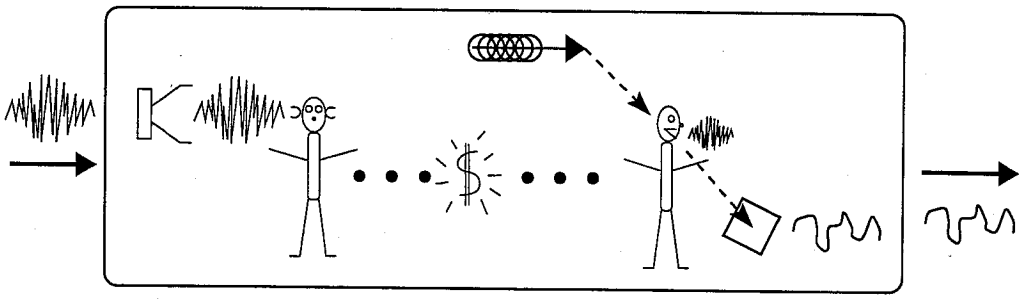


Figure 8.2: The Million Dollar Recovery Machine for solving the acoustic to articulatory inversion problem. Inside the machine is the same speaker who spoke the original utterance along with a measurement mechanism for tracking their articulators and a million dollars in cash. The speaker is given the money and asked to repeat as carefully as possible the utterance from the acoustic signal. As they repeat the utterance a second time, the movements of their articulators are recorded and output.

solves the recovery problem by first getting the human to do speech recognition and prosody estimation and subsequently doing recovery-by-synthesis. While this may be true to a certain extent, several studies (for example see [2]) have shown that humans are very good at repeating nonsense syllable strings (that do not form real words or phrases) so long as the syllables are drawn from those present in their native language. Such results suggest that humans possess a more low-level ability to infer articulator movements than simply the ability to do speech recognition. Another indication of this low level ability is the fact that babies can learn to speak—when they begin listening to speech sounds and mimicking them, they have no high level knowledge to guide them.

Nonetheless, a careful definition of the colourful but imprecise terms "hear" and "shape" in the question above is crucial to any discussion of the inversion problem. In this chapter I will define these terms with reference to simultaneously collected speech and movement data such as the X-ray microbeam data used in this thesis. By "hear" I will mean receiving a short-time spectral analysis of an entire utterance such as a spectrogram or sequence of features that model the smoothed short-time spectra (such as all-pole models). By "shape" I will mean recovering the simultaneous (midsaggital) configuration of the articulators as recorded in the database. What does "recover" mean? Clearly it denotes the reconstruction of the movements, but how do we judge how well we have done? I believe that this question can only reasonably be answered with reference to the **application** of recovered movements to a recognition, compression or speaker identification task.

By computing the short-time spectra of the real speech data in the UBDB (as described in section 6.1) it is possible to investigate the empirical analog of the question "Can you hear the shape of the mouth?" One important initial investigation is whether the *instantaneous mapping* from short-time spectrum back to mouth shape is well defined. In other words, for *observed* short time spectra that are very similar, how similar are the simultaneously *observed* positions of the mouth? Such an investigation is presented in detail in section 8.2 below. The answer is that there exist many cases of very similar short time spectra corresponding to very different instantaneous mouth shapes. This in turn means that it is not possible to infer the shape of the mouth (or, more correctly, its midsaggital projection) from a *single* short-time spectrum taken from normal English speech. Recovery, then, must be performed on **sequences** of short-time spectra to yield **sequences** of movements. This investigation is presented in section 8.3 in which it is shown that limited recovery of movements from complete utterances is possible in several cases.

8.1.2 Previous work

Acoustic to articulatory inversion is a problem that has always attracted research by engineers interested in speech science. It has developed into a major subfield of speech. Most techniques involve an initial inversion based on instantaneous mappings followed by some refinement procedure to enforce continuity in the articulatory domain. Some researchers ignore the one-to-many nature of the instantaneous inversion in the hopes that correct information from those times at which inversion is not ill-posed coupled with articulatory continuity will be sufficient to reconstruct movements. Others explicitly deal with the one-to-many problem by selecting several candidate articulatory positions at each time during the original instantaneous inversion and then choosing between candidates using some kind of path selection based on continuity. Work by Atal, Shondi and Schroeter, Levinson and others at Bell labs, by Papcun, Zacks and others at MIT and by Flanagan and his group at Rutgers [5, 171, 144, 32, 31, 30, 157, 207, 114] has employed various codebook-lookup or neural-network strategies to estimate an original articulatory trace followed by refinement. Work by Hogden, Nix and colleagues at Los Alamos labs is closest in spirit to the present work and takes a different approach of learning a probabilistic model of acoustics based on an internal state which explicitly only moves slowly and smoothly [89, 88, 86, 85, 84, 90]. This work, however, is based on the crucial assumption that to each sound type there corresponds only one unique articulatory position. Section 8.2 presents a study showing that this is not a good assumption in normal spoken English.

8.2 Instantaneous inverse mapping is ill-posed

Instantaneous inversion of acoustics into articulation is the problem of trying to estimate mouth shape from a single short-time spectrum of speech (typically estimated over about 25ms). Many simplified physical models of the speech production process have the feature that several different internal configurations can all cause the identical acoustic spectrum at the output. These include lossless tube approximations (see for example [59]) as well as more complex fluid flow models [175, 174]. Thus, in such **model** systems the instantaneous problem is clearly *ill-posed*, i.e. no unique and correct answer can be given.

In this section, I show that the same is true of actual speech production data. I present a simple empirical study. The X-ray microbeam data was processed as described in chapters 5 and 6 to produce two simultaneous vector time series: one containing the midsaggital positions of the eight tracking beads and another containing line spectral pairs (LSP) representing the short-time spectra. As in chapter 5, each short-time spectrum was calculated over a 23.5ms window (512 samples) centred on the middle time of the corresponding articulatory frame. Thus, the frame rate for spectral analysis was exactly the same as for the articulatory sampling, namely 6.866ms or roughly 146Hz.

The data was then sorted according to the following procedure. For each frame f, the K nearest frames to f in articulatory space were located across all available data. Denote these frames $N_K(f)^{artic}$. The K nearest frames to f in spectral space were similarly located across all data; call these $N_K(f)^{lsp}$. "Nearest" in each case was defined using a Mahalanobis distance metric based on the global covariance matrix of the articulatory data or the line spectral pair data. Specifically, the distance between frames f_1 and f_2 in articulatory space was:

$$d(f1, f2)^{artic} = (\mathbf{x}_1 - \mathbf{x}_2)^T C_{artic}^{-1} (\mathbf{x}_1 - \mathbf{x}_2) \tag{8.1}$$

where $\mathbf{x}_1$ and $\mathbf{x}_2$ are the 16 x and y coordinates of the eight tracking beads in each frame and C_{artic} is the global covariance matrix of $\mathbf{x}$ across all the data. Similarly, the distance between frames f_1 and f_2 in spectral space was:

$$d(f1, f2)^{lsp} = (\ell_1 - \ell_2)^T C_{lsp}^{-1} (\ell_1 - \ell_2) \tag{8.2}$$

where ℓ_1 and ℓ_2 are the LSP frequencies of the short-time spectra in each frame and C_{lsp} is the

global covariance matrix of ℓ across all the data. The frames $N_K(f_0)^x$ are then the K frames f_k for which $d(f_0, f_k)^x$ is smallest.

The K articulator positions corresponding to the two sets $N_K(f)^{artic}$ and $N_K(f)^{lsp}$ can then be considered with reference to the position $\mathbf{x}_f$ of the original frame f. The set $N_K(f)^{artic}$ will certainly fall near $\mathbf{x}_f$ – this is how the set was chosen. But what about the set $N_K(f)^{lsp}$? If similar sounds always have similar articulation, then this set should also be close to $\mathbf{x}_f$. If on the other hand there are many disjoint articulatory configurations which all cause the same sound, then the set of $N_K(f)^{lsp}$ will be spread out away from $\mathbf{x}_f$ or even multi-modal. These sets can be plotted graphically and compared. If the plots of $N_K(f)^{lsp}$ are in fact much more delocalized than those for $N_K(f)^{artic}$ then such plots would constitute direct evidence for the ill-posed nature of instantaneous inversion in real speech data.

Figures 8.3 and 8.4 show exactly such plots made from the Wisconsin X-ray microbeam data. A value of $K = 1000$ was used, chosen so that there would be ample data for the displays but also to be a small fraction of the entire database (about 2×10^5 frames in all).

The results clearly show that the instantaneous inversion problem *is* ill-posed in real speech production data from spoken English. This one-to-many aspect of the acoustic to articulatory mapping has led many researchers to declare the inverse mapping problem to be impossible or at least ill-posed. However, this objection ignores the crucial continuity property that articulators possess. Thus, while we may not be able to specify a *single* articulatory configuration given a *single* spectrum, we may indeed be able to reconstruct articulatory *traces* from *sequences* of spectra. This use of temporal continuity and smoothness information should allow better recovery of articulator information than a simple "inverse-lookup" procedure (see for example [171]). The remainder of this chapter (starting with section 8.3) discusses an approach for such recovery using the full time trajectory of spectral features.

8.2.1 How do we know that "close" is really close?

A natural concern when looking at the plots in figures 8.3 and 8.4 is the nature of the acoustic-spectral distance metric. Comparing LSP frequencies using a Mahalanobis distance (which is a Euclidean-like metric although warped by the covariance matrix) may seem dangerous. The reader may wonder if this method of measuring distance invalidates the arguments made above. While this is an important question, it turns out not to be a concern. The origin of the confusion is the somewhat unusual nature of the LSP parameterization. A dubious reader may reason as follows:

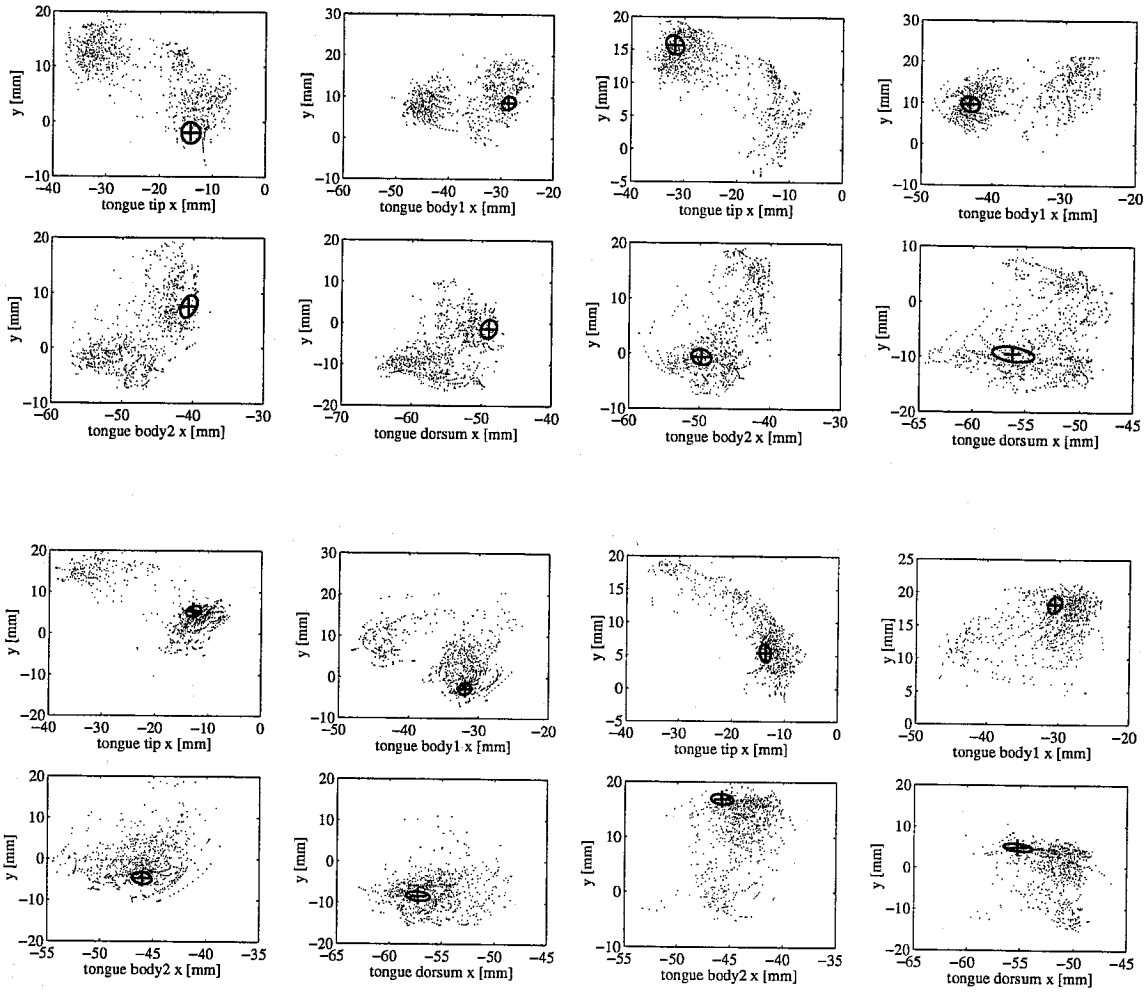


Figure 8.3: Instantaneous mappings from acoustics to articulation are ill-posed in real production data. Each plot shows the 1000 frames from the entire database which are nearest to one key frame (indicated with the thick cross) by two measures. The thick line is the two-standard deviation contour of the 1000 nearest frames using an articulatory position distance metric. The dots are the 1000 nearest frames using an acoustic-spectral measure based on LSP frequencies. Spread or multimodality in the dots indicates that many different articulatory configurations are used to generate very similar sounds. These plots offer direct empirical evidence for the ill-posed nature of instantaneous acoustic to articulatory inversion. See also figure 8.4.

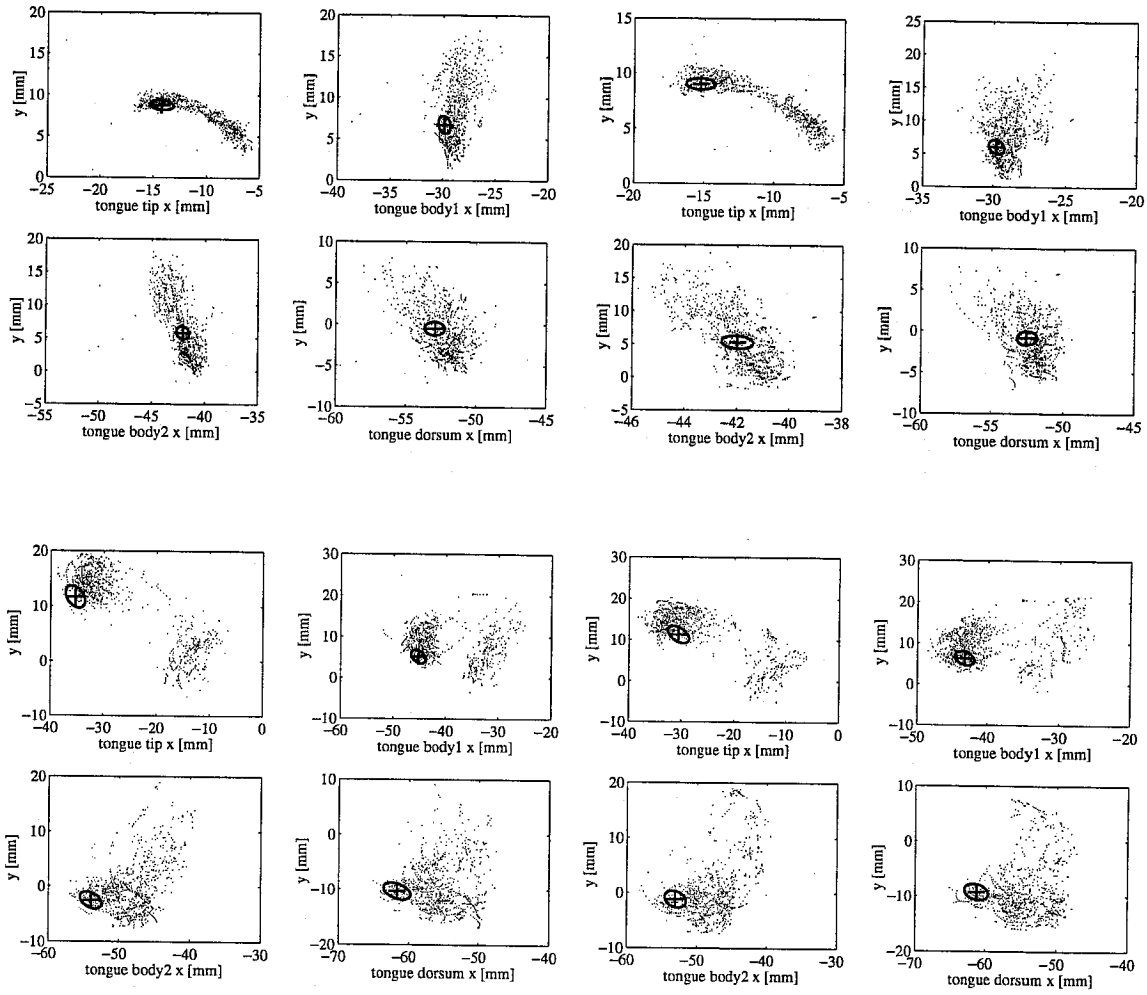


Figure 8.4: Instantaneous mappings from acoustics to articulation are ill-posed in real production data. Each plot shows the 1000 frames from the entire database which are nearest to one key frame (indicated with the thick cross) by two measures. The thick line is the two-standard deviation contour of the 1000 nearest frames using an articulatory position distance metric. The dots are the 1000 nearest frames using an acoustic-spectral measure based on LSP frequencies. Spread or multimodality in the dots indicates that many different articulatory configurations are used to generate very similar sounds. These plots offer direct empirical evidence for the ill-posed nature of instantaneous acoustic to articulatory inversion. See also figure 8.3.

Imagine two spectra which have almost identical peaks except that one has a small extra peak between *two peaks of the other. This will cause the line spectral representations to be very different because all LSP frequencies after the extra peak in the one will be* renumbered *relative to the other. Clearly comparing LSP coefficients directly is not a good way to compare spectra.*

This argument is perceptive and correct but not relevant to the plots in figures 8.3 and 8.4 above. While it *is* true that spectra may be similar and yet have very different LSP frequencies, it is *not* true that spectra may have very similar LSP frequencies and yet be different. Since a particular set of LSP frequencies implies a unique spectrum, if we are close in LSP space, then we *know* the corresponding spectra are similar.

To address this concern empirically, I have made a plot of the 1000 nearest *spectra* to the key frame **by spectral measure**. In other words, I have plotted all 1000 spectra for the same frames whose articulator positions are represented by the **dots** in figures 8.3 and 8.4 above. Each thin solid line in figure 8.5 corresponds to a set of four dots in the plots of figures 8.3 and 8.4 and shows the power across frequency for one of the frames chosen to be near the key frame by the spectral measure. The thick solid line shows the spectrum of the key frame itself.

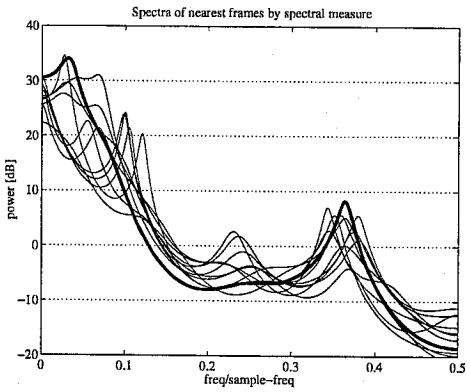


Figure 8.5: The 1000 nearest spectra by the **spectral distance measure** to the single key frame in figures 8.3 and 8.4. Each thin solid line corresponds to a set of four dots in the plots of figures 8.3 and 8.4 and shows the power across frequency for one of the frames chosen to be near the key frame by the spectral measure. The thick solid line shows the spectrum of the key frame.

For the interested reader I have also included a corresponding plot of the 1000 spectra for the frames whose positions are closest to the key frame in **articulatory** space. Each thin solid line in figure 8.6 corresponds to one of the points whose positions are summarized by the contour shown

by four thick lines in the plots of figures 8.3 and 8.4 and shows the power across frequency for one of the frames chosen to be near the key frame by the articulatory measure. The thick solid line again shows the spectrum of the key frame.

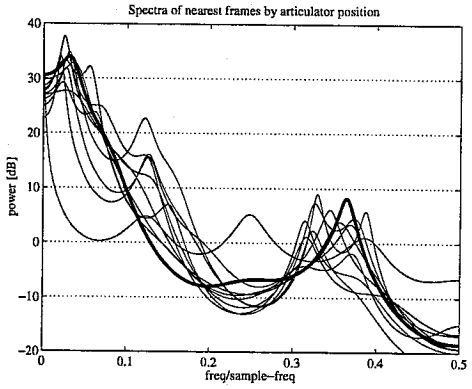


Figure 8.6: The 1000 nearest spectra by the **articulatory distance measure** to the single key frame in figure 8.3 and 8.4. Each thin solid line corresponds to one of the points whose contour is shown by four thick lines in the plots of figures 8.3 and 8.4 and shows the power across frequency for one of the frames chosen to be near the key frame by the articulatory measure. The thick solid line again shows the spectrum of the key frame.

Figures 8.7 and 8.8 contain the same plots as figures 8.5 and 8.6 except that they show the 1000 *furthest* frames using the spectral and articulatory distance measures.

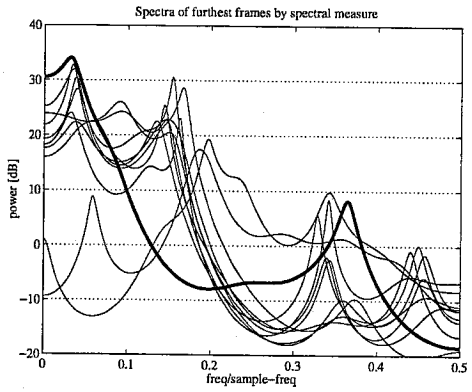


Figure 8.7: The 1000 **furthest** spectra by the **spectral distance measure** to one key frame in figures 8.3 and 8.4. Each thin solid line corresponds to a set of four dots in the plots of figures 8.3 and 8.4 and shows the power across frequency for one of the frames chosen to be near the key frame by the spectral measure. The thick solid line shows the spectrum of the key frame.

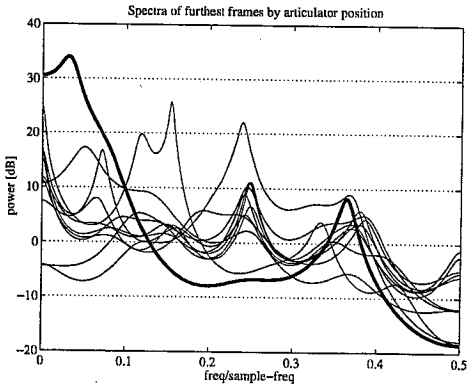


Figure 8.8: The 1000 **furthest** spectra by the **articulatory distance measure** to one key frame in figure 8.3 and 8.4. Each thin solid line corresponds to one of the points whose contour is shown by four thick lines in the plots of figures 8.3 and 8.4 and shows the power across frequency for one of the frames chosen to be near the key frame by the articulatory measure. The thick solid line again shows the spectrum of the key frame.

8.3 Recovering articulatory movements for complete utterances

Having demonstrated that the instantaneous inverse problem is ill-posed, I devote the remainder of this chapter to studying the problem of recovering articulatory movements from complete utterances. In other words, I investigate attempting to recover sequences of midsaggital positions from sequences of spectral shapes. Two earlier chapters in this dissertation form the basis of the approach used to recover articulatory movements in this section.

On the one hand, the self-organizing hidden Markov models (introduced in chapter 7) in principle have the potential to be applied directly to the inversion problem. The SOHMM models could be trained directly on sequences of acoustic observations. This approach would represent a purely unsupervised philosophy in which the articulatory data was not used at all during training. In practice I was only able to make this approach work somewhat well on very limited data (`/s/V/d/` utterances only) and was unable to make it work when large maps were trained on the entire database. I believe the reason for the failure of SOHMMs on the larger problems is that local minima in the learning cause enormous problems, even when the hierarchical training techniques described in chapter 7 are used. The SOHMMs are capable of modeling the speech production process well, but it is extremely difficult to find an initialization that leads learning into a good minimum. The SOHMM model is in some sense *too powerful* to allow effective learning.

On the other hand, the global linear forward mapping from articulator positions to acoustics (discussed in chapter 6) can be used as the observation matrices in a simple linear dynamical

system model. Given a time series of acoustic observations, state inference could be performed on such a system (as discussed in Appendix B) to attempt to recover articulator movements. This would represent inversion of a purely supervised linear model which was trained both on acoustics and corresponding articulator positions. This technique suffers from the opposite difficulty than do the SOHMMs. The global linear model lacks the power to model the production process properly. This was seen in chapter 6 when it was noted that the global models performed much less well than the mixtures of local models for synthesizing utterances. Thus, while no further learning needs to be performed and inference is exact and efficient, the recovered movements match the actual movements very poorly.

The approach I have taken is a synthesis of these two techniques. Ideally, we would like to be able to perform exact state inference on the mixture of local linear models which model the production process much better. However, exact inference for these models is intractable (see for example [133]). The hybrid algorithm I describe below solves this problem by using a SOHMM to perform state inference on the discrete variable indicating which local model is active at any time. Another way to think of the algorithm is that it uses the supervised training for the local linear forward models to set an initialization for the SOHMM. When taken in combination these two techniques yield an algorithm which does reasonable recovery of articulatory movements as shown in section 8.4.4.

8.3.1 An unsatisfactory unsupervised approach: direct application of self-organizing hidden Markov models

In the direct application of SOHMMs to this inversion problem, only the acoustic data are used during training. The acoustic data are processed by computing short-time spectral estimates (as described in chapter 6). These short-time spectra are then vector quantized into a finite codebook of discrete symbols. Each utterance is thus converted into a sequence of integers. These integer sequences are used to train discrete output SOHMMs of varying state space dimension, as detailed in chapter 7. The hierarchical training techniques across scale and dimension are applied when training to attempt to find good local maxima of learning.

The training procedure is *unsupervised*: articulatory data are not used to estimate any parameters of the system. However, in order to evaluate how well the algorithm has captured the underlying structure of the data, after it had been run, the true articulatory data are used to compare with its internal state trajectories. The best *single* linear transformation between all the learned state

trajectories in the SOHMM map and all the actual observed articulator movement sequences is computed. This single transform can then be applied to any individual state trajectory to generate a recovered articulator trace. In the experiments presented below, all available utterances were used to fit the transform, but the results are similar if a random subset of half are left out of this last fitting step. Notice that some such normalization will always be necessary after running an unsupervised algorithm since it is *impossible* to recover anything better than an orthogonal transformation (rotation plus axis aligned scaling) of the true information. (Because the coordinate axes and measurement units used to specify the articulator movements are both arbitrary.)

Figures 8.9 and 8.10 show the results of applying this procedure to only a very limited set of sounds from the microbeam speech data. In particular I used 56 noiseless utterances of vowels, consonant-vowel-consonant triples (/s/V/d/) and vowel-consonant-vowel triples (/uh/C/a/) all from a single male speaker. Acoustic features consisted of 12 mel-frequency cepstral coefficients based on 23.5 ms windows at a frame rate of 6.866 ms (designed to match the articulatory sampling). These cepstral coefficients were then vector-quantized using a codebook of 64 symbols. The codebook was trained using a batch version of the *k-means* algorithm. The SOHMM algorithms were then applied to train models with state spaces of various dimensions from 1 to 10.

The results are somewhat satisfactory on this limited dataset. Unfortunately, however, this method does not scale at all to larger datasets, bigger maps or more vector quantization symbols. Learning the SOHMM parameters is simply too difficult–local maxima and overfitting problems are paramount. These limitations mean that this purely unsupervised approach cannot ultimately be used to recover articulator movements from entire continuous speech utterances.

8.3.2 An unsatisfactory supervised approach: direct inversion of the global forward model by Kalman smoothing

Recall (from Appendix B) the form of a linear dynamical system:

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{w}_t = \mathbf{A}\mathbf{x}_t + \mathbf{w}_\bullet \qquad \mathbf{w}_\bullet \sim \mathcal{N}(0, \mathbf{Q}) \tag{8.3}$$

$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v}_\bullet \qquad \mathbf{v}_\bullet \sim \mathcal{N}(0, \mathbf{R}) \tag{8.4}$$

where $\mathbf{A}$ is the $k \times k$ *state transition matrix*, $\mathbf{C}$ is the $p \times k$ *observation, measurement,* or *generative* matrix, $\mathbf{x}$ is the hidden state and $\mathbf{y}$ are the observations.

If we identify the internal states $\mathbf{x}$ with midsaggital articulator positions and the outputs $\mathbf{y}$ with

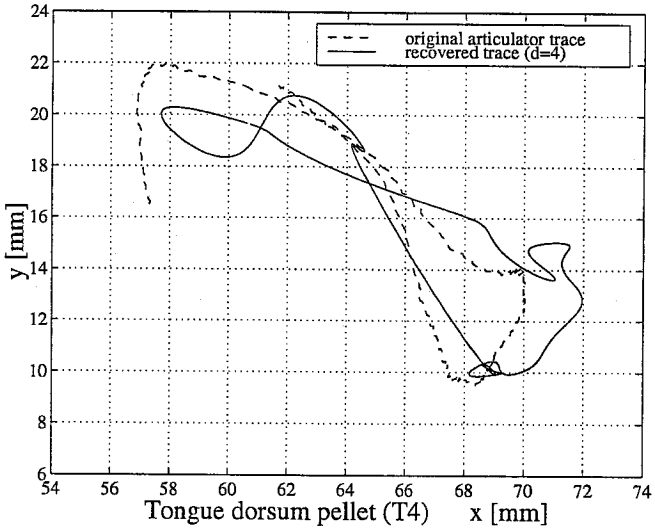


Figure 8.9: A measured trace of articulator data and its reconstruction after direct application of the SOHMM algorithms to acoustic data only. The horizontal and vertical axes are the x and y coordinates of the pellet. For this example, a four-dimensional map containing 1296 cells and 64 output symbols was learned. The recovered trace shown here is a linear transformation of a trajectory learned from only acoustic, not articulator data. A single linear transformation was used to project all the map trajectories onto the actual articulator traces, as opposed to a different transform for each pair.

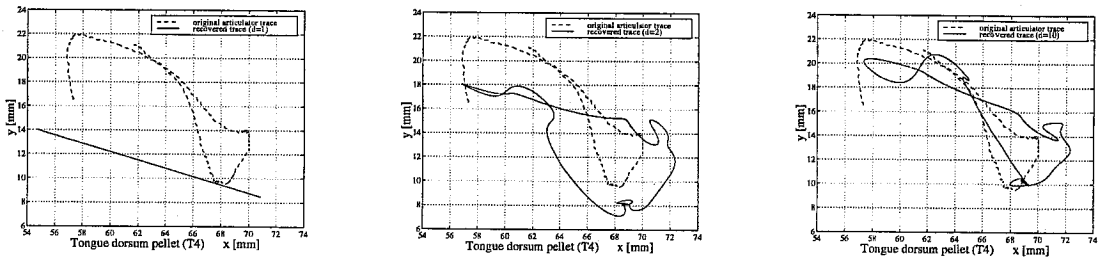


Figure 8.10: Similar reconstruction for a one-dimensional, two-dimensional and ten-dimensional map. Notice that there is little improvement after four dimensions although this may be a data scarcity problem. The hierarchical training techniques described in chapter 7 were used to train higher-dimensional maps. Regularization was also applied to higher-dimensional maps to avoid overfitting. The recovered trace shown here is a linear transformation of a trajectory learned from only acoustic, not articulator data.

spectral measurements (such as LSP frequencies), then we can see that the global linear forward model from chapter 6 is nothing more than the observation matrix $\mathbf{C}$. It generates spectral shapes from articulator positions. Setting the dynamics matrix $\mathbf{A}$ to the identity matrix and using a driving noise $\mathbf{w}_\bullet$ of the appropriate scale, we can model the articulator movements as a random walk in articulatory space. (For the matrices $\mathbf{Q}$ and $\mathbf{R}$, we may take the noises $\mathbf{w}_\bullet$ and $\mathbf{v}_\bullet$ to be distributed according to scaled down Gaussians with the same shape as the global covariances of articulator positions and LSP frequencies but smaller variances.)

Given this model setup, we are in a position to take a sequence of spectral-acoustic features $\mathbf{y}_t$ and perform *state inference* to attempt to recover the articulatory movement trajectories $\mathbf{x}_t$. This state inference can be performed using the Kalman smoother discussed in Appendix B. This represents another approach to doing acoustic to articulatory inversion based on the supervised forward mapping learned from both acoustics and articulatory data.

After reviewing the details of the smoothing algorithm, I will present the results of trying this approach on X-ray microbeam data and conclude that, like the purely unsupervised SOHMM approach, it too is unsatisfactory.

Inference in linear dynamical systems by Kalman smoothing

Appendix B described the inference problem for linear dynamical systems and its solution by Kalman filtering and the RTS smoothing algorithm (also known informally as Kalman smoothing). Here I review the actual formulas used for smoothing since they can be somewhat complicated, but refer to the discussion in Appendix B for more context and description. Inference in a linear dynamical system involves computing the posterior distributions of the hidden state variables given the sequence of observations. With Gaussian noise assumptions and Gaussian probabilities for the initial states, all the hidden state variables are Gaussian distributed, and are therefore fully described by their means and covariance matrices. The algorithm for computing the posterior means and covariances consists of two parts: a forward recursion which uses the observations from $\mathbf{y}_1$ to $\mathbf{y}_t$, known as the *Kalman filter* [103], and a backward recursion which uses the observations from $\mathbf{y}_T$ to $\mathbf{y}_{t+1}$ [155]. The combined forward and backward recursions are known as the Kalman or Rauch-Tung-Streibel (RTS) smoother.

To describe the smoothing algorithm, it will be useful to define the following quantities: $\mathbf{x}_t^s$ and $\mathbf{V}_t^s$ are respectively the mean and covariance matrix of $\mathbf{x}_t$ given observations $\{\mathbf{y}_1, \ldots \mathbf{y}_s\}$; $\hat{\mathbf{x}}_t \equiv \mathbf{x}_t^T$ and $\hat{\mathbf{V}}_t \equiv \mathbf{V}_t^T$ are the "full smoother" estimates. To learn the $\mathbf{A}$ matrix using *EM*it

is also necessary to compute the covariance across time between $\mathbf{x}_t$ and $\mathbf{x}_{t-1}$ although this is currently not being done. For this section only to preserve clarity of notation, the transpose of a vector or matrix is written as $\mathbf{x}'$ not $\mathbf{x}^T$. The symbol T is used instead to denote the length of a time series.

```
LDSInference(Y, A, C, Q, R, x_1^0, V_1^0)  % Kalman smoother
  for t = 1 to T  % Kalman filter (forward pass)
      x_t^{t-1} ← A x_{t-1}^{t-1}  if t > 1
      V_t^{t-1} ← A V_{t-1}^{t-1} A' + Q  if t > 1
      K_t ← V_t^{t-1} C' (C V_t^{t-1} C' + R)^{-1}
      x_t^t ← x_t^{t-1} + K_t (y_t − C x_t^{t-1})
      V_t^t ← V_t^{t-1} − K_t C V_t^{t-1}
  end
  initialize V̂_{T,T-1} = (I − K_T C) A V_{T-1}^{T-1}
  for t = T to 2  % Rauch recursions (backward pass)
      J_{t-1} ← V_{t-1}^{t-1} A' (V_t^{t-1})^{-1}
      x̂_{t-1} ← x_{t-1}^{t-1} + J_{t-1} (x̂_t − A x_{t-1}^{t-1})
      V̂_{t-1} ← V_{t-1}^{t-1} + J_{t-1} (V̂_t − V_t^{t-1}) J'_{t-1}
      V̂_{t,t-1} ← V_t^t J'_{t-1} + J_t (V̂_{t+1,t} − A V_t^t) J'_{t-1}  if t < T
  end
  return x̂_t, V̂_t, V̂_{t,t-1} for all t
```

Global Kalman smoothing on X-ray microbeam data

Figure 8.11 shows the results of performing Kalman smoothing on acoustic sequences from the X-ray microbeam database. The state vector $\mathbf{x}$ is taken to be the 16 coordinates of the eight articulatory bead positions. The output vector $\mathbf{y}$ is taken to be the 12 LSP frequencies plus the log energy. The matrix $\mathbf{C}$ was taken to be the global linear model which was learned to do forward mapping from articulators in chapter 6. The noise covariances $\mathbf{Q}$ and $\mathbf{R}$ were each set to one-tenth of the global covariance of articulator positions and LSP frequencies respectively. The initial probability distribution over states was taken to be the global mean and covariance of bead positions.

It is clear that this approach is also unsatisfactory – the recovered movements match the actual movements very poorly. The global linear model is not a good model of the production process, and so we cannot expect that attempting to invert it will yield correct results. The difficulty in this case is not one of local maxima as with the purely unsupervised approach, since no learning needs

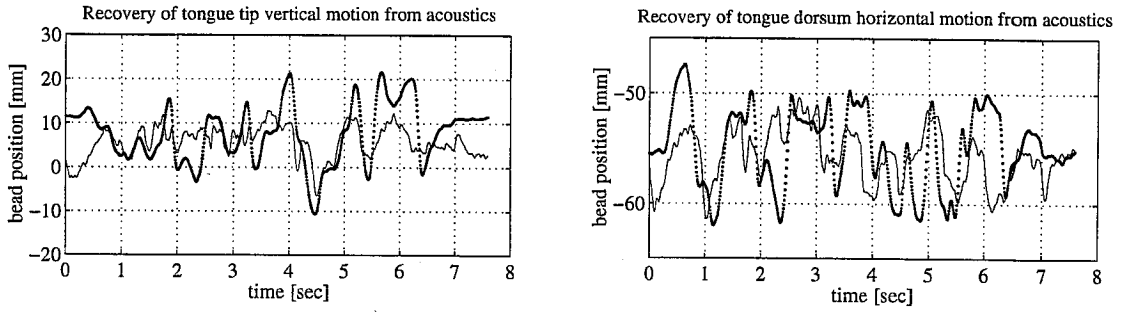


Figure 8.11: Recovered articulator movements using Kalman smoothing on a global linear model. Dots show the actual measured articulator movements; the thin line is the model estimation of movements from the corresponding acoustics. The panels show movements of a single bead coordinate (vertical position of the tongue tip on the left and horizontal position of the tongue dorsum on the right) versus time.

to be performed. Nor is it a problem of being unable to perform inference (which is what prevents us from doing this same thing with the superior local models) since inference is exact. The model simple is not powerful enough to capture the relationship between acoustics and articulation.

8.4 A combined approach: induced probabilities and coupled inference

The two (unsuccessful) techniques described above can be combined together in a successful way. The intuition behind the combined method is that we would like to be able to perform exact state inference on the mixture of local linear models since from the forward mapping studies of chapter 6 we know they model the production process quite well. However, inference is difficult because without knowing where the articulators are we cannot know which submodel to use. And we do not know where the articulators are because that is what we are trying to infer!

One approach is to do approximate inference using the global model as shown above and then use the results of this inference to identify which submodel is appropriate at each time. However, the quality of the global inference is too poor for this approach to be successful: the submodel or "mode" chosen by this method is too often incorrect. Fortunately, there is a way of using a SOHMM to discover these "modes" or submodel choices. Once we know the correct submodels we can apply Kalman smoothing exactly as described above in section 8.3.2 to recover the articulatory movements from acoustics.

8.4.1 A "mode SOHMM" for learning submodel mode sequences

First, I will describe how a "mode SOHMM" can be trained which identifies the correct submodel choice at each time given a sequence of acoustic features. The idea is as follows: if we look at the *actual* articulator movements we can trivially determine which local model is being used at any time by identifying the closest codebook vector in the vector quantizer. (In fact, this had to be done during the training of the mixture of local forward models to know which portions of each utterance each submodel should train on.) Thus, the articulatory movements can be easily converted into a sequence of discrete symbols (integers) representing the sequence of submodels activated as an utterance proceeds. The trick is **to train a SOHMM on this sequence of discrete symbols**. I will call the resulting SOHMM a "mode SOHMM". This training should be easy to do, since switching between submodels can only be caused by the articulators moving in a smooth way. In other words, there definitely *is* an underlying connected state space through which the system is moving to produce the sequence of submodel choices. Figure 8.12 shows the result of training a 5×5 square grid SOHMM on the sequence of VQ modes for all the articulator data from speaker `jw45`. It can be seen from the font sizes that almost all the cells have probability distributions dominated by a single VQ mode. This is an excellent result of learning—it means that the underlying SOHMM does not have to "cheat" by hedging its bets on which mode to output from each cell.

14	25 3	4	22	1020
3	28	29 1 16	323124	15
17	18	9 26	11 6	27
16	7 1 17	12	8 21	2
5	2319	1	30	13

Figure 8.12: The result of training a 5×5 square grid SOHMM on the sequence of VQ modes for all the articulator data from a single speaker. It can be seen from the font sizes that almost all the cells have probability distributions dominated by a single VQ mode. This is an excellent result of learning—it means that the underlying SOHMM does not have to "cheat" by hedging its bets on which mode to output from each cell.

The results can also be displayed in the style of figure 5.7. Instead of plotting a grid with

numbers in it, we can plot a grid with pictures of the modes. Figure 8.13 shows such a plot which displays the most likely VQ mode from each cell of the SOHMM map as a picture in that cell's position in the grid.

8.4.2 Converting a mode SOHMM into an acoustic model using induced probabilities

Once we have successfully trained a mode SOHMM on the sequence of submodel choices we can convert it into an acoustic model using a simple operation. Once again, we use the true articulatory movements to classify each frame in the database into one of the 32 submodels. We can then collect statistics on the **acoustic** features in each of these 32 modes separately. In other words, we look at all the spectra produced when the articulator positions were classified into a certain submode. These distributions of acoustic features in each mode can be used to *induce* a new "acoustic SOHMM" given the existing mode SOHMM as follows: create a new SOHMM with the same topology and connection matrix but with outputs in acoustic space rather than over submodel numbers. Call this new SOHMM the "acoustic SOHMM". In each state of the acoustic SOHMM, set the output distribution to be a mixture of the empirical output distributions in each mode according to the submodel probabilities in the corresponding state of the mode SOHMM. This new acoustic SOHMM now represents a low-dimensional map that generates sequences of acoustic features. This was exactly our goal in section 8.3.1. However, previously the articulator data was not being used in the training of the acoustic SOHMM. In this case we have used the articulatory data to train a SOHMM for VQ modes and then converted this mode SOHMM into an acoustic SOHMM. Another way to think of this algorithm is that it uses the supervised training for the local linear forward models to set an initialization for an acoustic SOHMM which is far better than a random initialization or one obtained from hierarchical training.

Notice that it is also possible to retrain the acoustic SOHMM slightly once it has been induced using the sequences of acoustic features from the database. This was not done in any of the experiments presented below, but has the potential to further improve the quality of the model.

8.4.3 Coupled inference for mode estimation

Once the acoustic SOHMM has been induced it can be used to do mode identification in a two stage *coupled inference* procedure. First, the incoming sequence of acoustic features is used to do

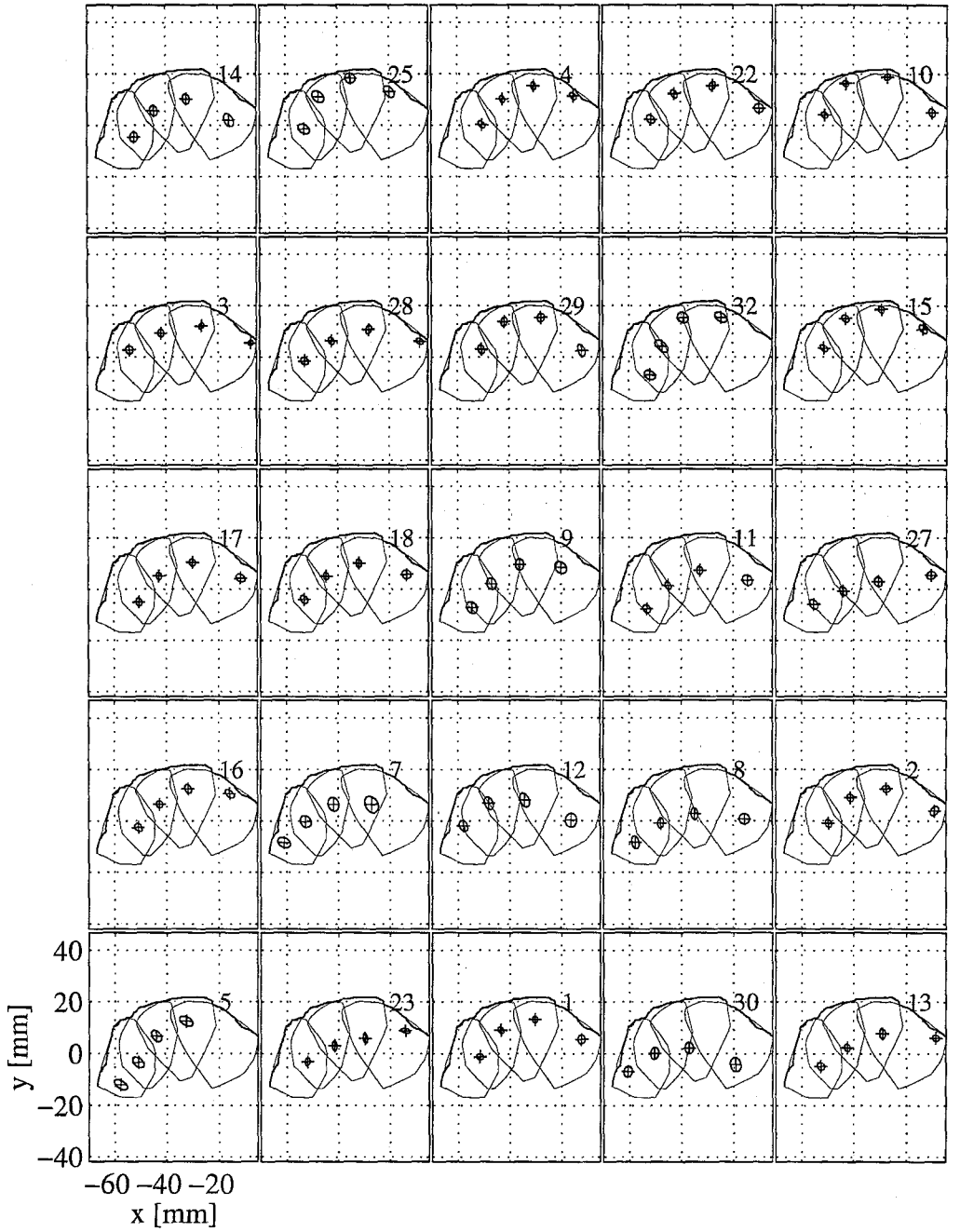


Figure 8.13: The trained SOHMM map of figure 8.12 plotted with pictures of each mode instead of numbers of each mode. Compare with figure 5.7. This shows result of training a 5×5 square grid SOHMM on the sequence of VQ modes for all the articulator data from a single speaker. The most likely VQ mode from each cell's histogram has been plotted.

state inference in the acoustic SOHMM. Then, the corresponding state sequence is run *generatively* through the mode SOHMM to determine the probability of occupation of each VQ mode at every time.

The result of the coupled inference procedure above is a discrete probability distribution over modes at each time step. Luckily, however, as shown in figure 8.12 the distributions in the original mode SOHMM have extremely low entropy. In fact, most states have all of their probability mass on a single mode. This means that for practical purposes we can make an approximation and take only the most likely mode from the coupled inference at each time without expecting too many errors. This most-likely-mode sequence can then be used to do Kalman smoothing on the local linear models. Such an approximation is very similar to a Viterbi decoding in a regular SOHMM.

Once the Kalman smoothing has been performed using the original most-likely-mode sequence, we can use the inferred articulatory trajectories to re-estimate the mode sequence. If the mode has changed at any time steps, Kalman smoothing can be redone. This cycle can be repeated if necessary.

8.4.4 Using inferred modes to do local Kalman smoothing

I will now show the results of doing local Kalman smoothing using the modes inferred by coupled inference. A 25 state (5×5 square grid) SOHMM with a two-dimensional topology space was trained on the true sequences of modes as determined by the measured articulatory positions. (The same local models were used here as in chapters 5 and 6 so there are 32 modes, or VQ clusters, in all.) Figures 8.14 and 8.15 show the results of the coupled inference followed by local Kalman smoothing for recovering articulator movements. In contrast to the purely unsupervised SOHMM approach and the direct Kalman smoothing of global models, the results are quite good.

8.5 Limitations: states of the mouth and smoothness constraints

What are the states of the speech production system? While it is certainly true that the positions and velocities of the movable structures—the *articulators*—are an important part of this state, they by no means completely specify it. The excitation to the system in the form of airflow from the lungs and glottis is also very important. Voicing (oscillation of the vocal cords) is another crucial factor. Pressure built up behind the lips before a plosive burst is an important state variable which distinguishes the burst of sound made when the lips are opened from the lack of sound made when

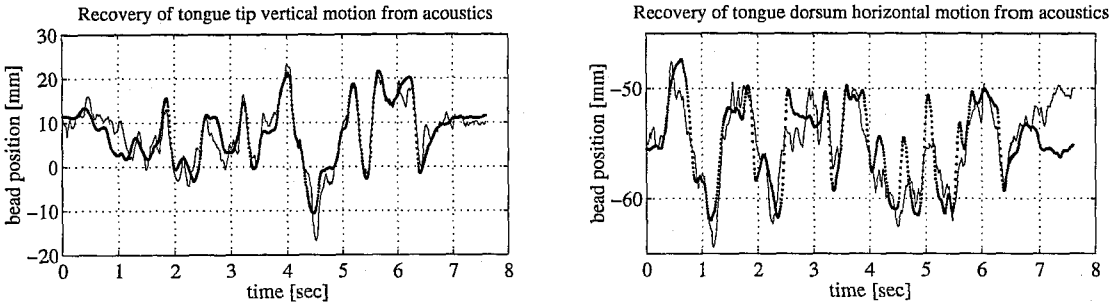


Figure 8.14: Recovered articulator movements using Kalman smoothing on local linear models. The *coupled inference* procedure described in the text is used to do state inference through an acoustic SOHMM which has been generated by *probability induction.* The coupled inference extracts probabilities across the different submodels at each time, which allows Kalman smoothing to be performed using the local models. Dots show the actual measured articulator movements; the thin line is the model estimation of movements from the corresponding acoustics. The panels show movements of a single bead coordinate (vertical position of the tongue tip on the left and horizontal position of the tongue dorsum on the right) versus time.

they are closing again.

Because they are inertial objects controlled by muscles, the positions and velocities of articulators change slowly and smoothly. However, many of these other important physical quantities such as voicing or built up pressure can change extremely rapidly. Thus, the assumption of slowly and smoothly changing state variables is only true for a subset of the factors controlling speech production.

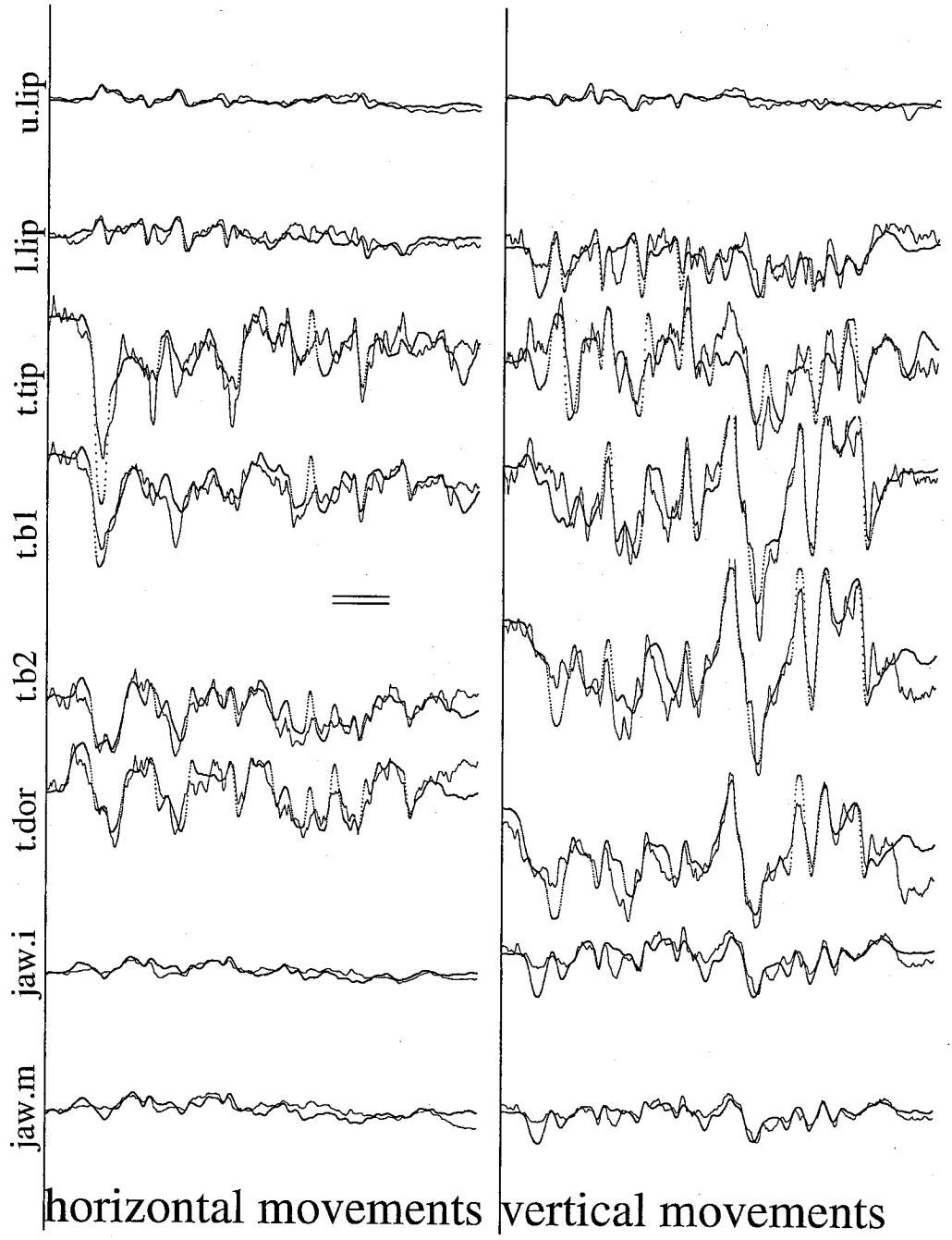


Figure 8.15: Recovered articulator movements using Kalman smoothing on local linear models. See caption of figure 8.14 for details. Each panel shows the movements of a single bead coordinate versus time. All vertical axes are the same scale. The bead names are shown on the far left. Horizontal movements are plotted in the left-hand column and vertical movements in the right-hand column. The separation between the two horizontal lines in the centre of the plot is the machine measurement error.

Chapter 9 Conclusions: From Analysis to Application

The investigations described in the preceding chapters of this dissertation have answered several empirical questions about the relationship between articulator movements and acoustics in normal spoken English speech. The results have all been derived from real speech production data consisting of simultaneous audio and movement recordings as measured in the Wisconsin X-ray microbeam database (described in detail in chapter 3). I have presented results describing the constraints on articulator positions (chapter 5), the forward mapping between articulators and acoustics (chapter 6) and the inverse mapping from acoustics back to articulation (chapter 8). I have also introduced new machine learning algorithms (chapters 4 and 7) which were developed in the course of those investigations.

In this final chapter, I present some preliminary investigations and ideas on the subject of how articulatory methods might be brought to bear on practical speech applications such as recognition, synthesis, speaker identification and coding. The reader is cautioned that research in all of these application areas is extremely advanced. The following discussion should in no way be interpreted as presenting comparisons with existing systems on any aspect of performance. I have certainly not constructed a speech recognizer nor a coder that will perform on par with any state-of-the-art system. Furthermore, the amount of data I have been able to analyze, while quite large from a speech science perspective, is quite small on the scale of tuning large engineering systems. Rather, I will attempt to outline ideas about how these problems could be tackled using an articulatory approach. In many cases I will actually perform preliminary experiments showing that **some** sort of articulatory system can be constructed to solve a certain speech processing task. The hope is to stimulate thought and excitement about future research directions and decidedly **not** to make strong quantitative claims about the performance promise of articulatory methods.

9.1 Articulatory speech recognition

The obvious paradigm for articulatory speech recognition is *inversion-then-matching*. An acoustic signal (the same input as to a traditional recognizer) would be analyzed using an articulatory model to infer the gestures underlying its production. The inferred gestures would then form an

intermediate representation between the original acoustics and the word or phone level transcript sought. Recovered movements would be matched against stored templates to perform recognition in this intermediate space.

For such an approach to succeed, two things need to be simultaneously true. First, it must be possible to construct systems that **reliably** recover internal states of models from acoustics alone. Second, it must be possible to do reliable speech recognition by matching such internal state trajectories with reference templates. While the actual physical movements of the speech articulators are one obvious choice of internal state variables they are by no means the only possibility. Ultimately, what is important is for the internal states of the model—whatever they may be—to reliably capture information useful for the recognition task. In fact, the actual physical movements of mouth parts are certainly *not* the best choice for internal states. However, they form a reasonable starting point from which it is possible to make a preliminary evaluation of articulatory methods.

In chapter 8 a system was described that recovered to a limited degree articulator movements from acoustics. Those results indicated that the recovery problem, while still not fully solved, is not insurmountable. In what follows I will try to give a similar indication about the second requirement for successful articulatory recognition: the usefulness of matching articulatory movements.

9.1.1 A brief review of previous work

A few very rudimentary versions of true articulatory speech recognition systems have been described in the past by a small handful of researchers. Each system is slightly different is philosophy and in implementation, but each has contributed an important piece of knowledge to this fledgling field. Zlokarnik (Los Alamos) [208, 209] showed that if true articulator movements were added to a simple recognition system, they improved recognition results. Furthermore, he trained a multi-layer perceptron to estimate articulator positions from the short-time spectrum of the signal and showed that even such estimated articulatory information can give a small improvement in system performance. Zlokarnik's experiments, however, involved only very limited data. Also, the nature of his estimation system made it inherently impossible for him to address the one-to-many nature of the inverse problem. Papcun and colleagues (Los Alamos) also used a neural network to infer articulatory movements on a somewhat larger database [144, 139]. Hogden, Nix and colleagues (Los Alamos and UC Boulder) have a much more complex system for recovering articulatory movements from acoustics based on a model known as *continuity mapping*

[89, 88, 86, 85, 84, 90, 138, 87]. However, their system still uses only limited data (three minutes in the most recent reports). It is also not yet mature enough to be tested on a recognition task using out-of-sample data, so all of their recognition experiments thus far have involved evaluating the system on training data it had previously seen. Finally, their system is based on the crucial assumption that to each sound type there corresponds only one unique articulatory position. The results of chapter 8 clearly show that this assumption is regularly violated during the production of normal English speech. Nix has just completed a thesis on recovering articulation using a slightly different model, which may avert this difficulty, but at the present time the document is not available. Sinder and his colleagues in Flanagan's group at Rutgers have developed a "voice-mimic" system which inverts the states of a simple articulatory model [158, 32, 31, 30, 157]. This system uses a fixed mapping to generate an initial guess of articulatory model parameters. This initial guess is then refined to take into account continuity of articulatory parameters and the many-to-one nature of the inversion. Simon Blackburn (Cambridge) described [22, 23, 21] a self-organizing speech production model trained on the same X-ray microbeam data from Wisconsin as has been used in this work.[1] His system, however, does not attempt to infer articulatory movements from acoustics. Rather it only models the forward mapping from phonetics to articulation and then from articulation to acoustics. It is used as an "analysis-by-synthesis" system to *rescore* possible recognition strings suggested by a conventional HMM speech recognizer. The system does this by converting the phonetics of the proposed string into articulatory movements and then into acoustic-spectral features. These features can then be compared with the observed features to produce a score from the articulatory module. If this score is then carefully combined with the score from the original HMM recognizer, the resulting compound system can show some slight improvements in error rates. Dan Fain and Al Barr (Caltech) are working on an articulatory recognition system that uses hierarchical, interpolating HMMs. A top level state variable, driven by a Markov chain generates the phoneme sequence for an utterance. A second level of state variables representing articulator configurations is driven by this top level. The second level variables change slowly and smoothly over the course of the utterance and capture coarticulation effects explicitly. Both acoustic parameters and articulatory movements are generated by the second level HMMs. During recognition, the acoustic observations are used to infer the articulatory parameters which are then used to deduce the phoneme sequence. Fain and Barr are working with a copy of the Wisconsin X-ray data

[1] Many thanks to Simon for his encouragement when I was starting off on this project and for providing the occasional data file or set of model parameters.

that I have given to them, but have not yet published the results of their investigations.

9.1.2 Cheating experiments

As an important first step towards articulatory speech recognition, one may consider the paradigm of doing recognition with the **true** articulatory movements rather than recovered movements. The purpose of such "cheating experiments" (cheating because during real world recognition tasks we will never have access to the true articulatory movements) is to show that simple pattern recognition systems *can* be constructed which work on only articulatory domain information. Armed with this initial confidence, we can later go on to attempt true articulatory recognition.

I have constructed a very simple template matching system that uses *dynamic time warping* to compare incoming articulatory movements to a stored pattern. Dynamic time warping is a standard application of dynamic programming [18] to the problem of aligning two sequences in time. It was commonly used in speech processing before hidden Markov models became standard. In particular, given two sequences of feature vectors, $\{q_1, q_2, \ldots, q_t, \ldots, q_{Tq}\}$ and $\{r_1, r_2, \ldots, r_t, \ldots, r_{Tr}\}$, as well as a distance function $d(q, r)$ which compares feature vectors, dynamic time warping finds a reparametrization of the time axis $\tau(t)$ such that:

$$\sum_{i=1}^{Tq} d(q_i, r_{\tau(i)}) \tag{9.1}$$

is minimized subject to certain constraints on $\tau(t)$. Common constraints on the reparametrization of time include endpoint constraints and limits on how much speeding up or slowing down can occur. I have used the endpoint constraints $\tau(1) = 1$ and $\tau(Tq) = Tr$ and the rate constraints and $0.5 < \tau' < 3$ in my experiments. I have used Mahanalobis distance in the bead position covariances as the distance function $d(q, r)$ in order to compare articulatory feature vectors.

The pattern recognition experiment I have run is a simple single word recognizer. The recognizer consists of a single template of the word (for example, in the figures below I have shown results for the word "children") which was taken to be the actual articulator movements from one instance of that word in the database. This template is slid across the entire database of articulatory movements and the dynamic time warping algorithm is run in order to find the best match at each position given the warping constraints. The matching score at each time is the inverse of the total distance between the best warping of the articulatory movements and the template. The matching scores are then uniformly rescaled to set a threshold for recognition of the word at 100.

Figures 9.1 and 9.2 show the results of this word spotting experiment in isolated word speech and in continuous speech.

Using this simple single template dynamic time warping system and the true articulatory movements, it was possible to set a threshold for every word that I tried (six in all) so that the system gave perfect performance. That is, all occurrences of the word in the database gave a matching score above the detection threshold while at the same time no other words ever gave a score above threshold. Typically the templates were a few hundred milliseconds in length, and they were compared to roughly 1200 seconds of data giving no false positives or false negatives.

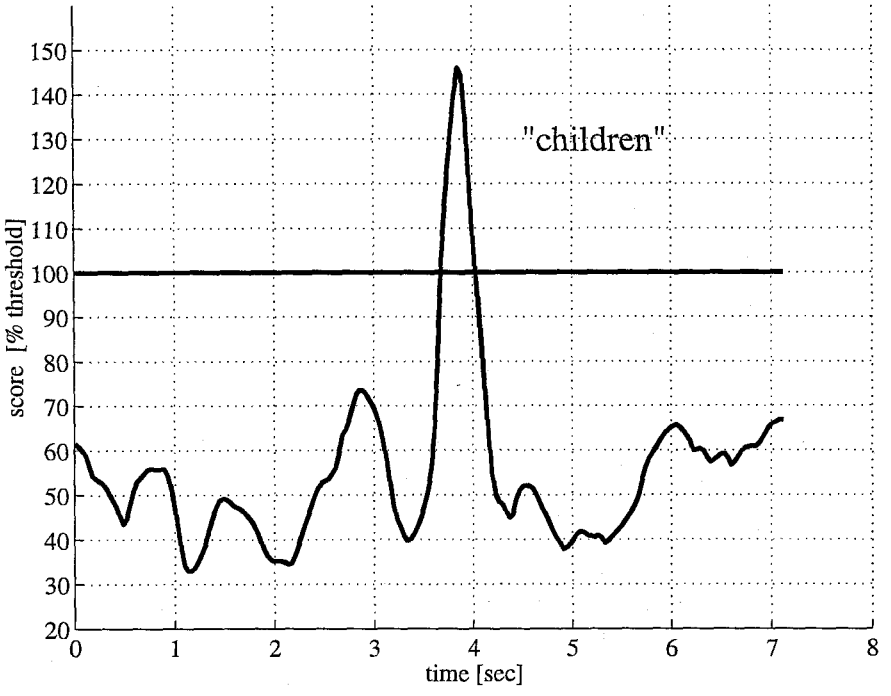


Figure 9.1: Articulatory speech recognition of isolated words using the true (measured) articulatory movements. A single template (the same as in figures 9.2 and reffig:wrartic1 below) of the articulator movements during an isolated utterance of the word "children" was matched across the entire database for one speaker. Shown is the matching score over time for an utterance of isolated words that contains another occurrence of "children" (at about 4sec). The matching threshold for detection pf the word is shown as a horizontal line at score 100.

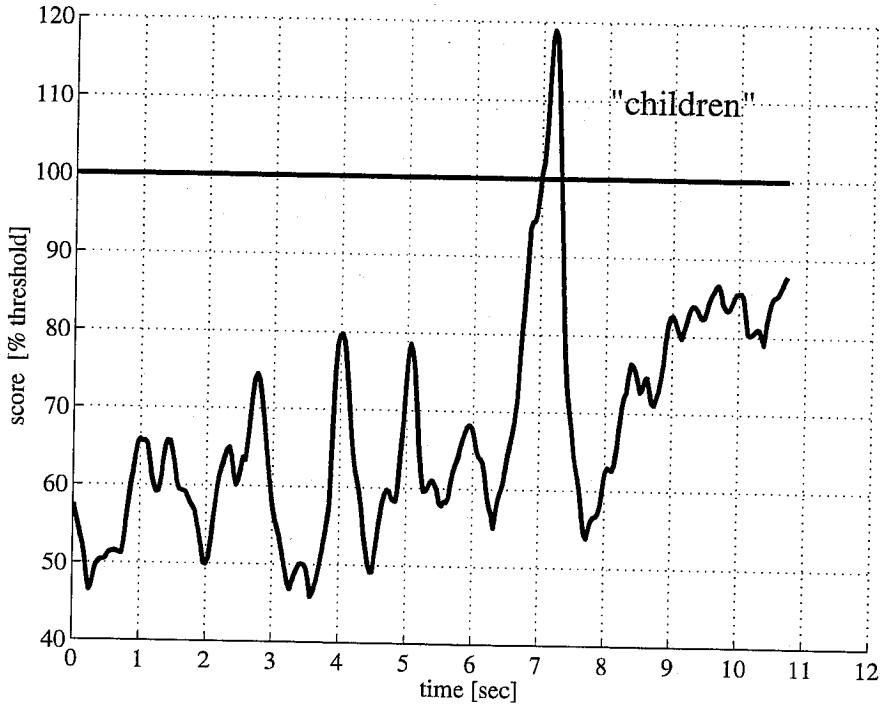


Figure 9.2: Articulatory speech recognition of individual words in continuous speech using the true (measured) articulatory movements. A single template (the same as in figure 9.1 above and figure 9.3 below) of the articulator movements during an isolated utterance of the word "children" was matched across the entire database for one speaker. Shown is the matching score over time during a continuous speech utterance that contains another occurrence of "children" (at about 7sec). The matching threshold for detection of the word is shown as a horizontal line at score 100.

9.1.3 True articulatory speech recognition—initial results

The above recognition experiments show that even a very simple single template system with a primitive pattern recognition algorithm can achieve perfect performance on a basic word spotting task in the articulatory domain. The practical culmination of the work of this thesis is to try the same experiment using estimated articulatory movements that were recovered from acoustics alone. Care has to be taken, however, to ensure that the utterance being tested has never been seen before by the system. In order to do this, I followed the following procedure: for each utterance in the database, I removed it temporarily from the database and retrained all of the forward models (global and mixtures of local models) as well as all of the mode SOHMMs and acoustic SOHMMs for doing articulatory recovery. I then applied the algorithms of chapter 8 to estimate the articulatory movements for that utterance which I had left out. In this way, the articulatory recovery is always being done on out of sample data previously unseen by the system. I then moved to the next utterance, temporarily removed it from the database and repeated all of the retraining procedures before doing articulatory recovery. Doing this for all utterances in the database gives recovered movement information for all of them which in each case was achieved by a system that had never seen that utterance before.

Armed with the recovered movements, I then applied exactly the same word spotting system described above to these signals. The template I used was still taken from the *true* articulatory movements from one utterance of the word being recognized.

Figure 9.3 shows the results of this word spotting experiment in isolated word speech. For all of the isolated word utterances in the database, it was again possible to set a threshold that gave perfect performance. However, this was not true if the entire database of continuous speech and vowel sounds was included. (In that case a few false positives appeared.) Nonetheless, this word spotting experiment represents a proof of concept example of a true articulatory speech recognition system that first recovers movements from acoustics and then does pattern matching based on those movements.

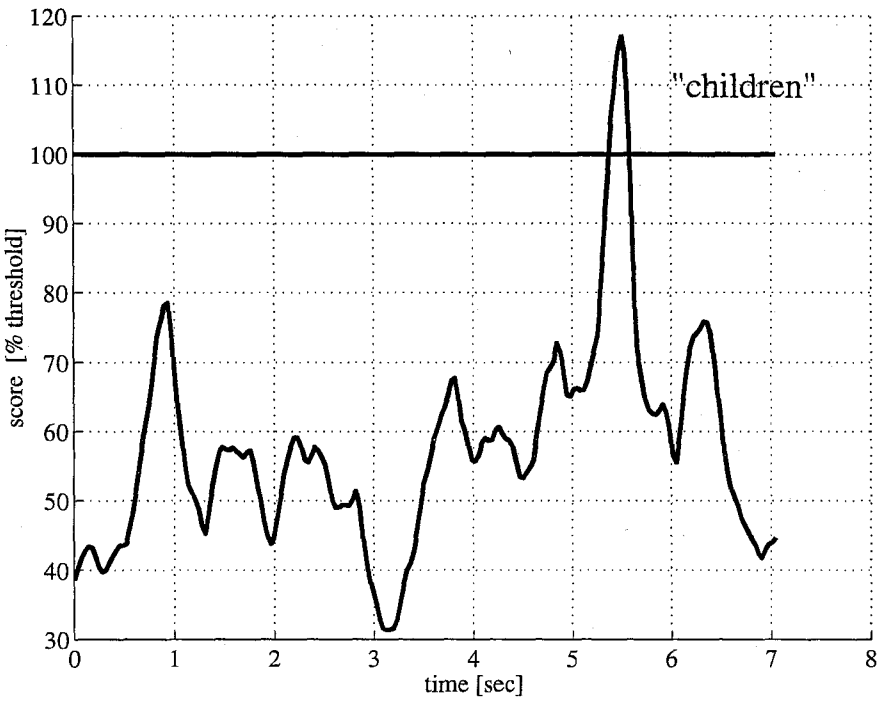


Figure 9.3: Articulatory speech recognition of isolated words using recovered movements derived from acoustic information only. The coupled inference procedure described in chapter 8 was used to estimate articulatory movements from only acoustic information on utterances that were not part of the training set for the inversion model. These out-of-sample recovered movements were matched to a single template (the same as in figures 9.1 and reffig:cheat2 above) of the true articulator movements during an isolated utterance of the word "children". Shown is the matching score over time for an utterance of isolated words that contains another occurrence of "children" (at about 5.5sec). The matching threshold for detection of the word is shown as a horizontal line at score 100. This represents successful wordspotting on out-of-sample data using recovered articulatory information starting with only acoustics.

9.2 Towards speaker independence: Several quantitative comparisons between speakers

All of the data analysis in this dissertation has been speaker dependent. Ultimately the hope is that the production modeling techniques described in this work can be extended to be speaker independent. In this section, I will present the results of several comparisons across speakers which suggest that this may be possible.

9.2.1 Bead covariances

Although speakers have different sized and shaped mouths, it appears that the scale on which movements of the beads occur is very similar. One way to see this is to plot many bead covariances together. In order for such a plot to have comparative value, all of the covariance ellipses must have their centres aligned. Figure 9.4 shows such a plot for the first 16 speakers in the database. The mean positions of the beads have been aligned across speakers and placed at the mean positions for one speaker. The covariance ellipses are then plotted without modification for all speakers. One can see that the scale and general alignment of each bead's covariance is similar across speakers. Thus, a model that attempted to make use of these individual bead covariances would not do poorly to assume a single model across all speakers. However, this use of only covariance information does depend on the means being properly aligned across speakers.

9.2.2 Comparison of global eigenmodes across speakers

Examining the individual bead covariances across multiple speakers is a first step towards discovering how similar the constraints on mouth position are from one speaker to another. It would be better, of course, to examine the full covariance across all beads, i.e. to compare the 16–dimensional eigenvectors. Figures 9.5, 9.6 and 9.7 show the first three global eigenvectors for several speakers.

How do the eigenvectors of various speakers compare to each other? As it turns out, the first few eigenmodes exhibit a surprising amount of regularity across different speakers in the database. The reader is strongly encouraged to look at the accompanying plots and make her own evaluation. Figures 9.5, 9.6 and 9.7 show the leading, second and third eigenmodes for 12 speakers from the database. In each case the eigenvector in question has been plotted exactly as described in figure 5.2 above. Notice that the symbols that mark the ends of the lines could all be flipped (at

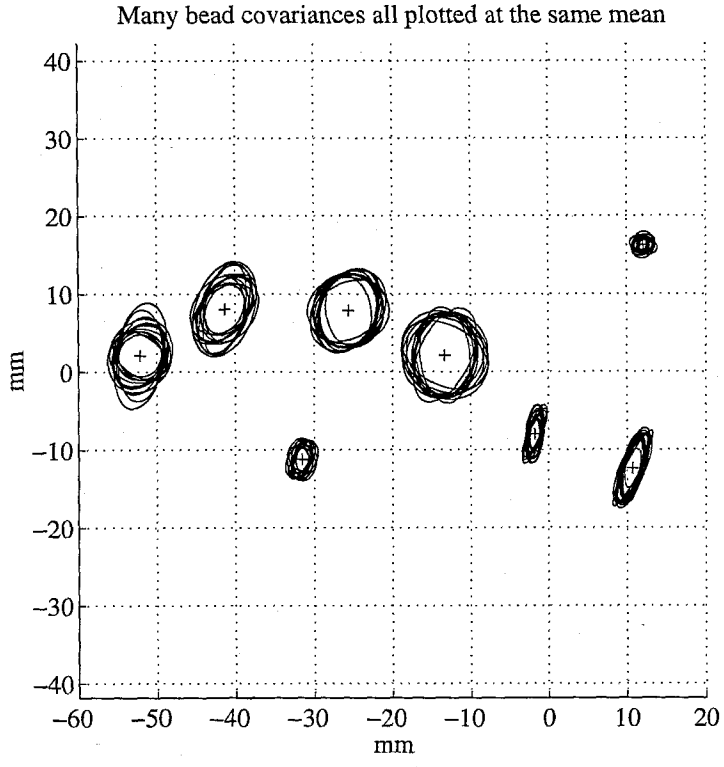


Figure 9.4: A comparison of bead covariances for several speakers. The mean positions of the beads have been aligned across speakers. The covariance ellipses are plotted without modification for the first 16 speakers in the database. Even though the speakers have mouths of different sizes and shapes, the scale and orientation of bead deviations from mean positions are similar.

once) to the opposite polarity without changing the communicated information. For ease of visual display I have kept this arbitrary degeneracy constant across all speakers. I have also plotted only the four tongue beads but left out the jaw and lip beads.

While linguistic interpretation of computational results is dangerous at best, it is nonetheless useful to provide qualitative descriptions of the modes as observed, keeping in mind the cautionary note just mentioned. The first eigenmode directs each bead roughly perpendicular to the palate wall. It moves the tongue towards and away from the roof of the mouth. The second eigenmode directs each bead roughly perpendicular[2] to the first mode, in other words roughly parallel to the palate. This moves the tongue out of the mouth or back into the mouth. The third eigenvector is a "rocking" mode in which the front of the tongue moves up or down (towards or away from the palate) and the back moves in the opposite direction. These verbal descriptions may be further motivated by viewing a sequence of eigenvector movies which displays, in turn, the leading eigenmode followed by the second and finally third eigenmodes for the same 12 speakers shown in the static plots 9.5, 9.6 and 9.7.

A few words about mode ordering and regularity

For two of the speakers in the above figures, the second and third eigenmodes **as ranked by their eigenvalues** were reversed from the pattern in all other speakers. This is merely an ordering question, and has to do with the fact that some speakers adopt mouth configurations that place more power in certain modes than most other speakers. At first it may seem strange to the reader that the eigenvectors exhibit strong regularity in their directions even though they may be misordered by eigenvalue. One often thinks of datasets in which two eigenvalues are close in value as having corresponding eigenvectors that are not well defined. (Consider for example a circular distribution for which all rotations of orthogonal axes are equivalent.)

This observation is true for Gaussian data, but it can be far from true for non-Gaussian distributions. Imagine for example a vibrating string pinned at its two ends. If the string is excited it will vibrate only with the well known sinusoidal modes that it supports. These modes are present as a consequence of the physical structure of the system. Imagine doing several experiments in which we excite such a string and compute the eigenmodes of its vibrations. We will always obtain the Fourier modes no matter what the excitation in each experiment (so long as the excitation

[2]Although the eigenvectors must be orthogonal in the full 16–dimensional space, this does not require the corresponding bead lines to be at right angles in the x-y plane.

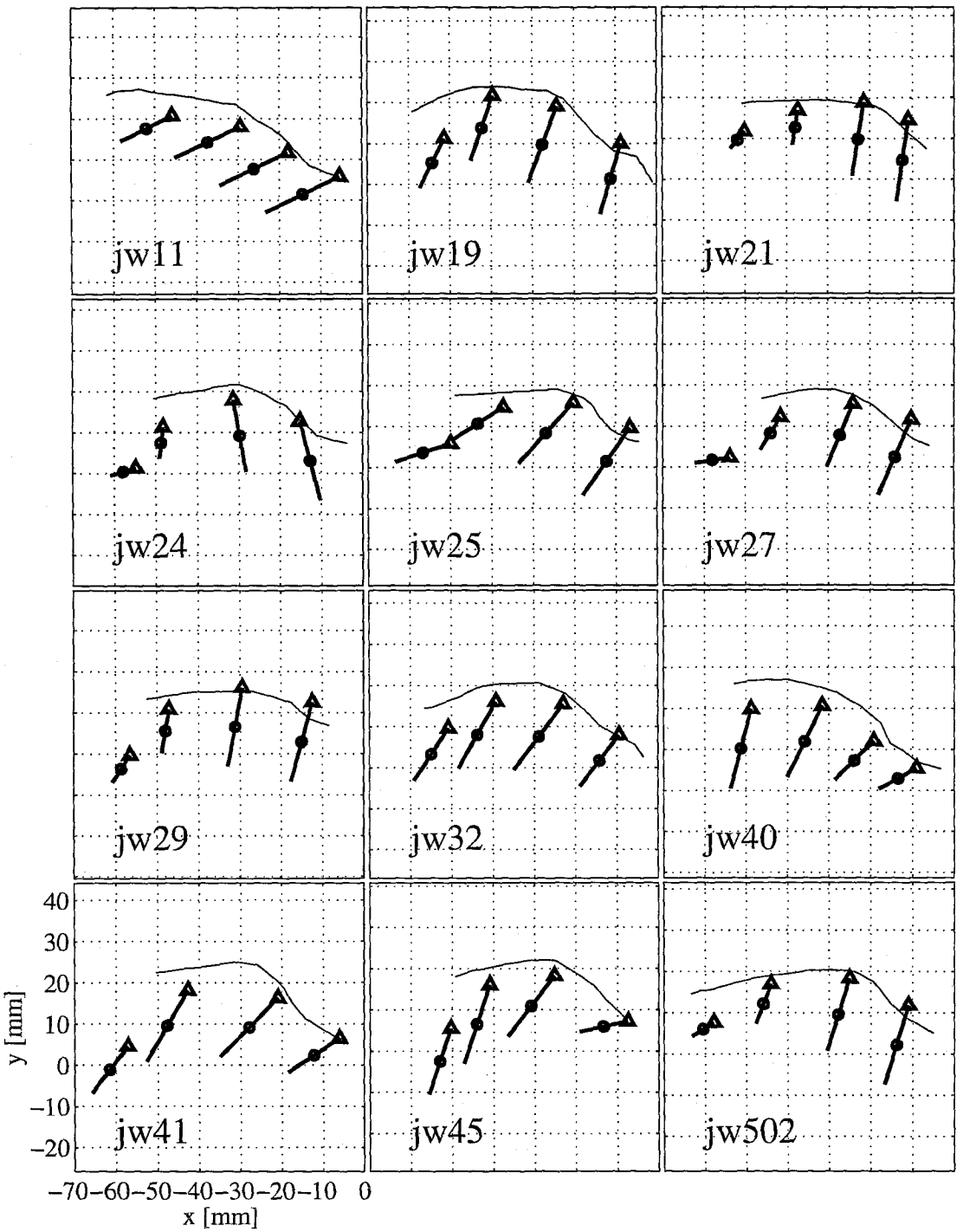


Figure 9.5: Leading eigenmode for 12 speakers in the database. This mode moves the tongue towards and away from the palate. The eight lines on each plot above together with their polarities as indicated by the symbols at one end of each line represent a single 16–dimensional eigenvector in articulatory space. Eigenvectors are computed by diagonalizing the sample covariance of all the bead data for a single speaker. Shown is the leading eigenvector (the mode with the greatest power) for 12 speakers.

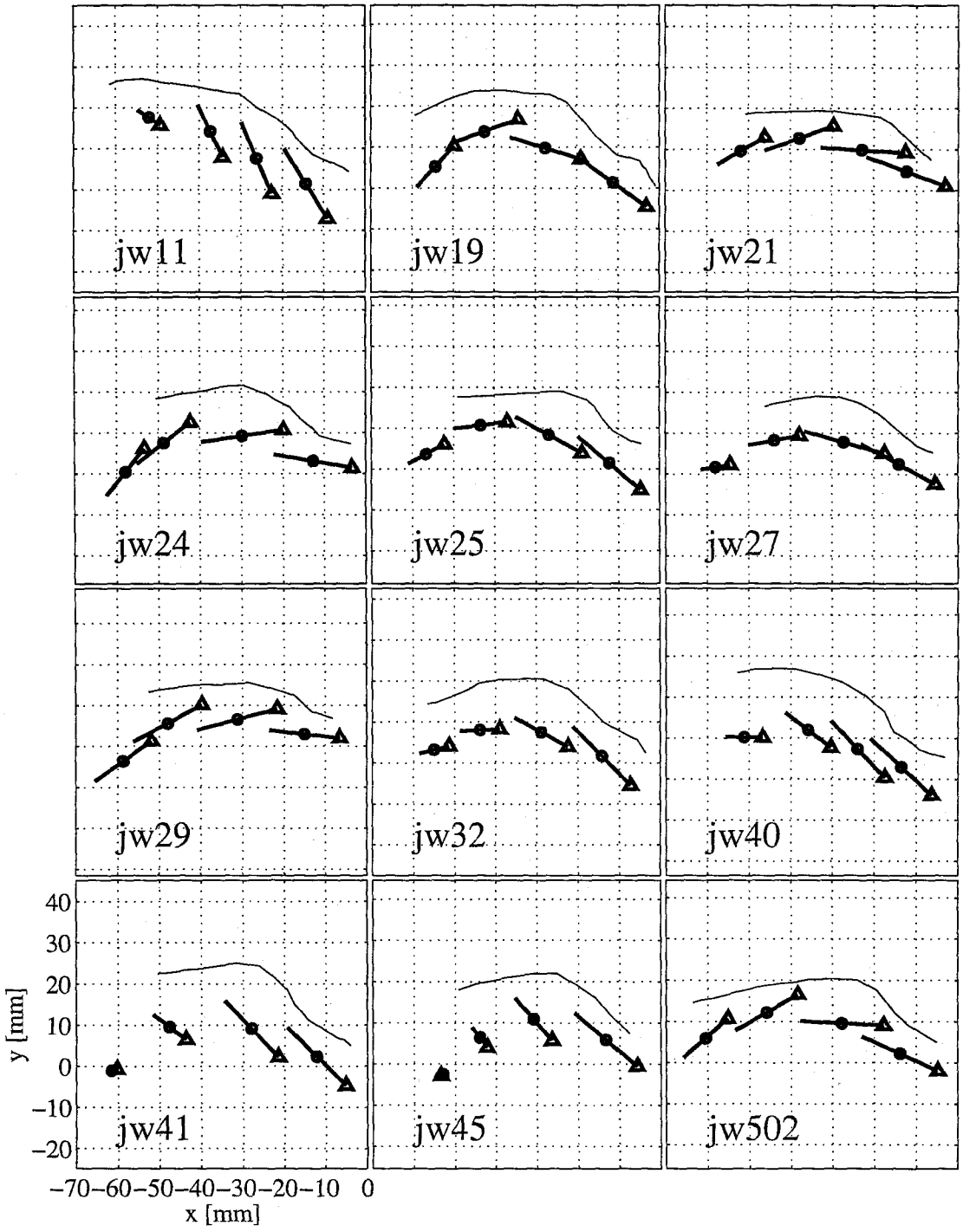


Figure 9.6: Second eigenmode for 12 speakers in the database. This mode moves the tongue in and out of the mouth. The eight lines on each plot above together with their polarities as indicated by the symbols at one end of each line represent a single 16–dimensional eigenvector in articulatory space. Eigenvectors are computed by diagonalizing the sample covariance of all the bead data for a single speaker. Shown is the second eigenvector for 12 speakers.

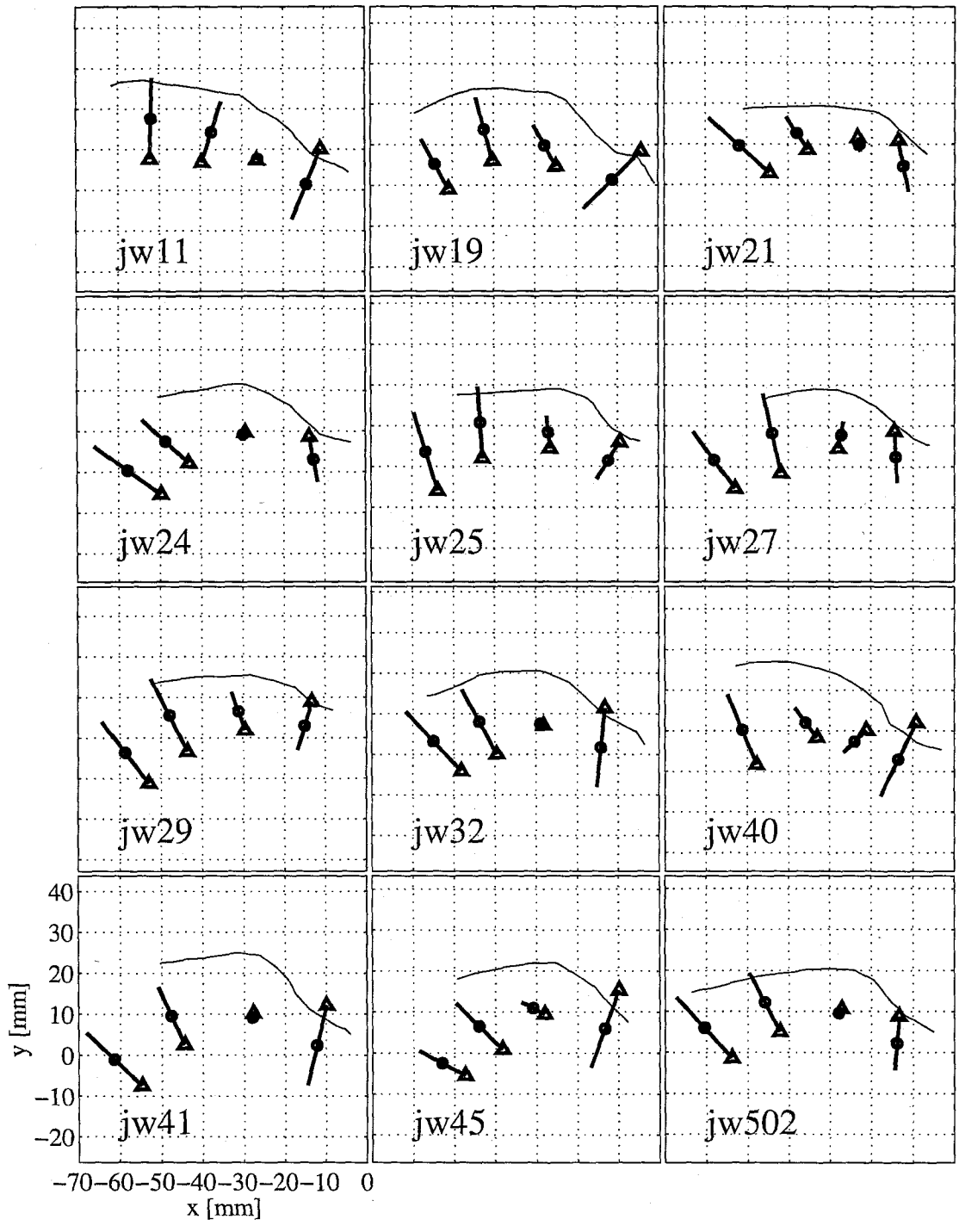


Figure 9.7: Third eigenmode for 12 speakers in the database. This is a "rocking" mode which tilts the tongue front relative to the back. The eight lines on each plot above together with their polarities as indicated by the symbols at one end of each line represent a single 16–dimensional eigenvector in articulatory space. Eigenvectors are computed by diagonalizing the sample covariance of all the bead data for a single speaker. Shown is the third eigenvector for 12 speakers.

is "sufficiently interesting"). However, if the excitation is random from one experiment to the next, the ordering of the modes may very well change because some experiments may inject more energy into certain modes than others. Yet it would certainly not be fair to say that the modes in the string are not well defined.

As with the string, so with the configurations of the mouth. The mechanical structure of human articulators may enforce configurations that have a regular decomposition. This regularity will persist even if one speaker chooses to use certain configurations more than another speaker thus causing a difference in the ranking of eigenmodes.

9.3 Speaker identification

Speaker identification using the articulatory information as measured in the UBDB is a trivial task. The mean positions of beads are sufficiently different for all speakers (as are the palatal traces, which can be extracted from the extrema of bead movements) to allow identification using only a few seconds of bead data. However, it is difficult to imagine how these properties of the articulatory data might be extracted directly from the acoustic signal. (Although some aspects of the palate outline such as its overall size do have well known acoustic manifestations.) Thus, it is interesting to investigate how much speaker-specific information an algorithm can discard while still retaining the ability to distinguish between talkers. Such experiments are done with a view to future applications which do not have access to the true articulatory data. Figure 9.8 shows that even when the mean positions of beads are removed and each bead covariance is whitened to the identity matrix, it is still possible to identify speakers using only a few seconds of movement data. This is because the **correlations** between how individual articulators move are unique to individuals. The plot shows the integrated error over time for compressing incoming movement data onto the eigenbases of many speakers. Each curve is the integrated error using a different speaker's eigenbasis for compression; all curves are trying to compress the same incoming data. After a few seconds the error for the correct eigenbasis is clearly much lower than all others.

9.4 Synthesis and compression

Articulatory models can also potentially offer something to the problems of speech synthesis and speech coding. Full *articulatory synthesis* systems attempt to generate speech waveforms by in-

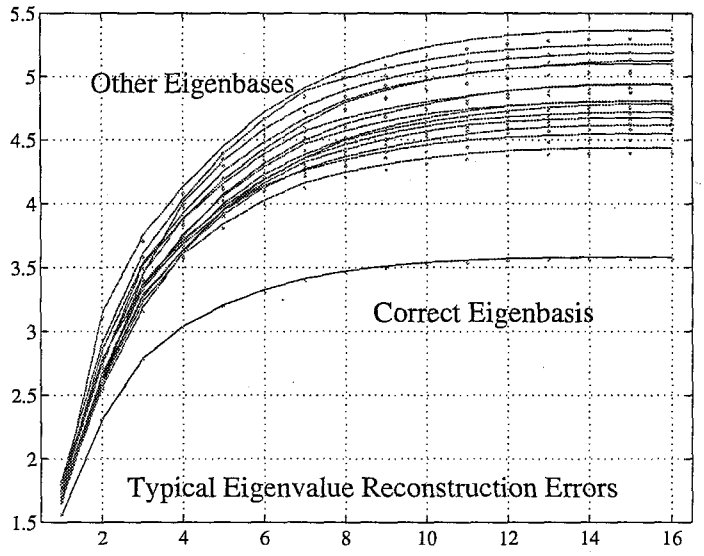


Figure 9.8: Speaker recognition using bead movement data. Even when the means of beads are removed and each bead covariance is whitened to the identity matrix, it is still possible to identify speakers using only a few seconds of true movement data. This is because the correlations between how the articulators are moved is unique to individuals. The plot shows the integrated error over time for compressing incoming movement data onto the eigenbases of many speakers. Each curve is the integrated error using a different speaker's eigenbasis for compression; all curves are trying to compress the same incoming data. After a few seconds the error for the correct eigenbasis is clearly much lower than all others.

voking a mapping between text (phonemes) and articulatory primitives. Such mappings generally attempt to model co-articulation and other such dynamic effect. The articulatory control signals are then rendered into a time-domain signal using a (usually highly complex) forward physical model. Such systems are generally extremely computationally demanding and produce very low quality utterances, because it is extremely difficult to model the various flow and turbulence effects of the vocal tract. However, another paradigm of interest is *resynthesis*. An articulatory resynthesis system attempts to infer articulator movements from an incoming acoustic signal and then does some manipulation of the data at the articulator level, followed by resynthesis into acoustics. The manipulations in question might involve modification of the articulatory control parameters, such as the introduction or removal of certain speech defects or disfluencies. Alternately, other articulatory parameters used in the resynthesis could be made different from those in the analysis system, for example, conversion from a man's voice into a woman's voice by changing the vocal tract length.

The idea behind *articulatory compression* is that if the internal states of a system are changing slowly and smoothly, then it is much easier to transmit or encode these internal economically than the outputs of the system. If an incoming acoustic signal could be converted into an articulatory representation, then this representation can be sent or stored compactly and then resynthesized elsewhere using a forward model. In fact, this is a generalization of the *compositional coding* idea presented in chapter 7 except that the state vector s is now a vector instead of a scalar. Many standard speech compression technologies use some variation of this idea to one degree or another. For example, code-excited linear prediction (CELP) is based loosely on the idea that the vocal tract acts as a slowly time-varying filter which is driven by stereotyped glottal pulses.

References

[1] J.H. Abbs, R.D. Nadler, and O. Fujimura. X-ray microbeams track the shape of speech. *SOMA*, pages 29–34, 1988.

[2] J. B. Allen. How do humans process and recognize speech? *IEEE Transactions on Speech and Audio Processing*, 2(4):567–577, 1994.

[3] A. Alwan, S. Narayanan, and K. Haker. Toward articulatory-acoustic models for liquid approximants based on MRI and EPG data. 2. the rhotics. *Journal of the Acoustical Society of America*, 101(2):1078–1089, 1997.

[4] J. A. Anderson and E. Rosenfeld, editors. *Neurocomputing: Foundations of Research.* MIT Press, Cambridge, 1988.

[5] B.S. Atal, J.J. Chang, M.V. Mathews, and J.W. Tukey. Inversion of articulatory-to-acoustic transformation in the vocal tract by a computer-sorting technique. *Journal of the Acoustical Society of America*, 63(5):1535–1555, 1978.

[6] B.S. Atal and S.L. Hanauer. Speech analysis and synthesis by linear prediction of the speech wave. *Journal of the Acoustical Society of America*, 50:637–655, 1972.

[7] H. Attias and C.E. Schreiner. Low-order temporal statistics of natural sounds. In M.C. Mozer et al., editor, *Advances in Neural Information Processing Systems*, volume 9, pages 27–33, MIT Press, 1997. Cambridge, MA.

[8] T. Baer, J.C. Gore, L.C. Gracco, and P.W. Nye. Analysis of vocal tract shape and dimension using magnetic resonance imaging: Vowels. *Journal of the Acoustical Society of America*, 90(2):799–828, August 1991.

[9] E. Bauer, D. Koller, and Y. Singer. Update rules for parameter estimation in bayesian networks. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence (UAI-97)*, pages 3–13. Morgan Kaufmann, 1997.

[10] L. E. Baum. An inequality and associated maximization technique in statistical estimation of probabilistic functions of a markov process. *Inequalities*, 3:1–8, 1972.

[11] L. E. Baum and J. A. Eagon. An inequality with applications to statistical estimation for probabilistic functions of markov processes and to a model for ecology. *Bulletin of American Mathematical Society*, 73:360–363, 1967.

[12] L. E. Baum and T. Petrie. Statistical inference for probabilistic functions of finite state Markov chains. *Annals of Mathematical Statistics*, 37:1554–1563, 1966.

[13] L. E. Baum, T. Petrie, G. Soules, and N. Weiss. A maximization technique occurring in the statistical analysis of probabilistic functions of Markov chains. *Annals of Mathematical Statistics*, 41(1):164–171, 1970.

[14] R. H. Baumann, J. C. R. Licklider, and B. Howland. Electronic word recognizer. *Journal of the Acoustical Society of America*, 26:137(A), 1954.

[15] A. G. Bell. Patent no. 174,465. U.S. Patent Office, February 14 1876.

[16] A. G. Bell. Prehistoric telephone days. *National Geographic Magazine*, 41:223–242, 1922.

[17] A. J. Bell and T. J. Sejnowski. Learning the higher-order structure of a natural sound. *Network: Computation in Neural Systems*, 7:261–266, 1996.

[18] R. Bellman. *Dynamic programming*. Princeton University Press, Princeton, New Jersey, 1957.

[19] Dimitri P. Bertsekas. *Dynamic Programming and Optimal Control*, volume 1. Athena Scientific, Bellmont, MA, 1995.

[20] Christopher Bishop. *Neural Networks for Pattern Recognition*. Clarendon Press, Oxford, 1995.

[21] Charles Simon Blackburn. *Articulatory Methods for Speech Production and Recognition*. PhD thesis, Cambridge University, Engineering Department and Trinity College Cambridge, December 1996.

[22] Simon Blackburn and Steve Young. Towards improved speech recognition using a speech production model. In *Proceedings of Eurospeech'95 Conference*, volume 3, pages 1623–1626, 1995.

[23] Simon Blackburn and Steve Young. Pseudo-articulatory speech synthesis for recognition using automatic feature extraction from x-ray data. In *Proceedings of International Conference on Speech and Language Processing (ICSLP'96)*, volume 2, pages 969–972, 1996.

[24] H. Bourlard, H. Hermansky, and N. Morgan. Towards increasing speech recognition error rates. *Speech Communication*, 18(3):205–231, 1996.

[25] H. Bourlard and N. Morgan. *Connectionist Speech Recognition – A Hybrid Approach.* Kluwer Academic Publishers, 1994.

[26] A. S. Bregman. *Auditory Scene Analysis: The Perceptual Organization of Sound.* MIT Press, 1990.

[27] C.P. Browman and L. Goldstein. Towards an articulatory phonology. *Phonology Yearbook*, 3:219–252, 1986.

[28] C.P. Browman and L. Goldstein. Articulatory gestures as phonological units. *Phonology*, 6:201–251, 1989.

[29] D. Byrd. Palatogram reading as a phonetic skill: a short tutorial. *Journal of the International Phonetic Association*, 23(2):59–72, 1994.

[30] S. Chennoukh, D. Sinder, G. Richard, and J. Flanagan. Methods for acoustic-to-articulatory mapping and voice mimic systems. *Journal of the Acoustical Society of America*, 101(5), May 1997.

[31] S. Chennoukh, D. Sinder, G. Richard, and J. Flanagan. Voice mimic system using an articulatory codebook for estimation of vocal tract shape. In *Eurospeech 1997*, Rhodes, Greece, September 1997.

[32] Samir Chennoukh, Daniel Sinder, Gael Richard, and James Flanagan. Articulatory based low bit-rate speech coding. *Journal of the Acoustical Society of America*, 102(5):3163, November 1997.

[33] T. Chiba and M. Kajiyama. *The vowel, its nature and structure.* Tokyo-Kaiseikan Publishing Company, Tokyo, 1941.

[34] Kenneth Church. *Phonological Parsing in Speech Recognition.* Kluwer Academic Publishers, 1987.

[35] Arthur C. Clarke. *2001: A Space Odyssey.* The New American Library Inc., New York, 1968. Based on a screenplay by Stanley Kubrick and Arthur C. Clarke.

[36] United States Congress. Communications act of 1934. Technical Report 47 USCA – 151, Federal Government, 1934.

[37] F. S. Cooper, P. C. Delattre, A. M. Liberman, J. M. Borst, and L. J. Gerstman. Some experiments on the perception of synthetic speech sounds. *Journal of the Acoustical Society of America*, 24:597–606, 1952.

[38] Thomas M. Cover. An algorithm for maximizing expected log investment return. *IEEE Transactions on Information Theory*, IT-30:369–373, 1984.

[39] E. David and O. Selfridge. Eyes and ears for computers. *Proceedings of the IRE*, pages 1093–1101, May 1962.

[40] H. Davis, R. Biddulph, and S. Balashek. Automatic recognition of spoken digits. *Journal of the Acoustical Society of America*, 24(6):637–642, November 1952.

[41] J. Deller, J. Proakis, and J. Hansen. *Discrete Time Processing of Speech Signals.* Macmillan, 1993.

[42] Bernard Delyon. Remarks on filtering of semi-Markov data. Technical Report 733, Institute de Recherche en Informatique et Systems Aleatiores, Campus de Beaulieu – 35042 Rennes Cedex – France, May 1993.

[43] A. P. Dempster, N. M. Laird, and D. B. Rubin. Maximum likelihood from incomplete data via the EM algorithm (with discussion). *Journal of the Royal Statistical Society series B*, 39:1–38, 1977.

[44] P. Denes and M. V. Mathews. Spoken digit recognition using time-frequency pattern matching. *Journal of the Acoustical Society of America*, 32:1450–1455, 1960.

[45] P. B. Denes. On the statistics of spoken English. *Journal of the Acoustical Society of America*, 35(6):892–905, 1963.

[46] Peter Denes and Elliot Pinson. *The Speech Chain: The Physics and Biology of Spoken Language.* W. H. Freeman and Company, New York, 2 edition, 1993.

[47] V. Digalakis, J. R. Rohlicek, and M. Ostendorf. ML estimation of a stochastic linear system with the EM algorithm and its application to speech recognition. *IEEE Transactions on Speech and Audio Processing*, 1(4):431–442, October 1993.

[48] R. O. Duda and P. E. Hart. *Pattern Classification and Scene Analysis*. Wiley, New York, 1973.

[49] H. Dudley. The carrier nature of speech. *Bell Systems Technical Journal*, 19:495–515, 1940.

[50] H. Dudley and S. Balashek. Automatic recognition of phonetic patterns in speech. *Journal of the Acoustical Society of America*, 30:721–732, 1958.

[51] H. Dudley and T. H. Tarnoczy. The speaking machine of wilhelm von kempelen. *Journal of the Acoustical Society of America*, 22, 1950.

[52] R. Durbin, S. Eddy, A. Krogh, and G. Mitchison. *Biological sequence analysis: probabilistic models of proteins and nucleic acids*. Cambridge University Press, 1998.

[53] Robert J. Elliott, Lakhdar Aggoun, and John B. Moore. *Hidden Markov Models: Estimation and Control*, volume 29 of *Applications of Mathematics*. Springer-Verlag, New York Berlin Heidelberg, 1995.

[54] B. S. Everitt. *An Introducction to Latent Variable Models*. Chapman and Hill, London and New York, 1984.

[55] G. Fant, J. Liljencrants, and Q. Lin. A four-parameter model of glottal flow. Technical Report 4, Swedish Royal Institute of Technology (KTH), Speech Transmission Laboratory Quarterly Progress Status Report, Stockholm, Sweden, 1985.

[56] Gunnar Fant. *Acoustic Theory of Speech Production*. Mouton and Company, The Hague, 1970 (first published 1960).

[57] Gunnar Fant. *Speech Sounds and Features*. MIT Press, Cambridge, Mass, 1973.

[58] J.D. Ferguson, editor. *Proceedings of the Symposium on the Application of Hidden Markov Models to Text and Speech*, Princeton, NJ, 1980. IDA–CRD.

[59] J. L. Flanagan. *Speech analysis, synthesis, and perception.* Springer-Verlag, New York, 1972.

[60] H. Fletcher. *Speech and hearing in communication.* Van Nostrand, Princeton, NJ, 1953.

[61] H. Fletcher. *Speech and hearing in communication (Van Nostrand 1953), edited by Jont B. Allen.* Acoustical Society of America, New York, NY, ASA edition, 1995.

[62] R. Fletcher and M. J. D. Powell. A rapidly convergent descent method for minimization. *Computing Journal*, 2:163–168, 1963.

[63] NIST March 1997 Hub-5E Benchmark Test Results for Recognition of Conversational Speech over the Telephone in English. `ftp://jaguar.ncsl.nist.gov/lvcsr/...mar97/eval/lvcsr_mar97_scores.970410/Summary`. Government web site., April 4 1997.

[64] C.A. Fowler. An event driven approach to the study of speech perception from a direct-realist perspective. *Journal of Phonetics*, 14:3–28, 1986.

[65] C.A. Fowler. Listeners do hear sounds, not tongues. *Journal of the Acoustical Society of America*, 99(3):1730–1741, 1996.

[66] A. M. Fraser and A. Dimitriadis. Forecasting probability densities by using hidden markov models with mixed states. In A. S. Weigend and N. A. Gershenfeld, editors, *Time series prediction: Forecasting the future and understanding the past*, pages 265–282. Addison Wesley, Reading, MA, 1993.

[67] D. B. Fry and P. Denes. The solution of some fundamental problems in mechanical speech recognition. *Language and Speech*, 1:35–58, 1958.

[68] Dennis Fry. *The Physics of Speech.* Cambridge University Press, Cambridge, 1979.

[69] O. Fujimura, H. Ishida, and S. Kiritani. Computer-controlled dynamic cineradiography. *Annual Bulletin of the Research Institute of Logopedics and Phoniatrics, University of Tokyo*, 2:6–10, 1968.

[70] O. Fujimura, S. Kiritani, and H. Ishida. Computer-controlled radiography for observation of the movements of articulatory and other human organs. *Computers in Biology and Medciine*, 3:371–384, 1973.

[71] Zoubin Ghahramani and Geoffrey Hinton. Parameter estimation for linear dynamical systems. Technical Report CRG-TR-96-2 [ftp://ftp.cs.toronto.edu/pub/zoubin/], Dept. of Computer Science, University of Toronto, February 1996.

[72] Zoubin Ghahramani and Geoffrey Hinton. Switching state-space models. Technical Report CRG-TR-96-3, Dept. of Computer Science, University of Toronto (Submitted for Publication), July 1996.

[73] Zoubin Ghahramani and Geoffrey Hinton. The EM algorithm for mixtures of factor analyzers. Technical Report CRG-TR-96-1 [ftp://ftp.cs.toronto.edu/pub/zoubin/], Dept. of Computer Science, University of Toronto, May 1996 (revised Feb. 1997).

[74] Zoubin Ghahramani and Michael I. Jordan. Supervised learning from incomplete data via an EM approach. In Jack D. Cowan, Gerald Tesauro, and Joshua Alspector, editors, *Advances in Neural Information Processing Systems*, volume 6, pages 120–127. Morgan Kaufmann Publishers, Inc., 1994.

[75] Paul Goldberg, Chris Williams, and Chris Bishop. Regression with input-dependent noise: A gaussian process treatment. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10, pages 493–499. The MIT Press, 1998.

[76] Gene H. Golub and Charles F. Van Loan. *Matrix Computations*. The Johns Hopkins University Press, Baltimore, MD, USA, second edition, 1989.

[77] G. C. Goodwin and K. S. Sin. *Adaptive filtering prediction and control.* Prentice-Hall, 1984.

[78] C. Gordon, D. Webb, and S. Wolpert. Isospectral plane domains and surfaces via Riemannian orbifolds. *Inventiones Mathematicae*, 110(1):1–22, 1992.

[79] W.J. Hardcastle. The use of electropalatography in phonetic research. *Phonetica*, 25:197–215, 1972.

[80] H. Hassenein and M. Rudko. On the use of discrete cosine transform in cepstral analysis. *IEEE Trans. Acoustics, Speech, and Signal Processing*, ASSP-32(4):922, 1984.

[81] John Hertz, Anders Krogh, and Richard G. Palmer. *An Introduction to the Theory of Neural Computation.* Lecture Notes Volume I. Addison Wesley, 1991.

[82] Geoffrey Hinton and Zoubin Ghahramani. Generative models for discovering sparse distributed representations. *Philosophical Transactions of the Royal Society B*, 352:1177–1190, 1997.

[83] Geoffrey Hinton, Michael Revow, and Peter Dayan. Recognizing handwritten digits using mixtures of linear models. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 1015–1022. The MIT Press, 1995.

[84] J. Hogden, A. Lofqvist, V. Gracco, I. Zlokarnik, P. Rubin, and E. Saltzman. Accurate recovery of articulator positions from acoustics: New conclusions based on human data. *Journal of the Acoustical Society of America*, 100(3):1819–1834, 1996.

[85] John Hogden and Michelle Heard. Evaluating a topographical mapping from speech acoustics to tongue positions. *Journal of the Acoustical Society of America*, 97(5):3401, May 1995.

[86] John Hogden, Anders Lofquist, Vincent Gracco, Kiyoshi Oshima, Philip Rubin, and Elliot Saltzman. Inferring articulator positions from acoustics: An electromagnetic midsagittal articulometer experiment. *Journal of the Acoustical Society of America*, 94(3):1764, September 1993.

[87] John Hogden, David Nix, Vincent Gracco, and Philip Rubin. Stochastic word models for articulatorily constrained speech recognition and synthesis. *Journal of the Acoustical Society of America*, 103(5):2774, May 1998.

[88] John Hogden, Philip Rubin, and Elliot Saltzman. An unsupervised method for learning to track tongue position from an acoustic signal. *Journal of the Acoustical Society of America*, 91(4):2443, April 1992.

[89] John Hogden, Elliot Saltzman, and Philip Rubin. Unsupervised neural networks that use a continuity constraint to track articulators. *Journal of the Acoustical Society of America*, 92(4):2477, 1992.

[90] John E. Hogden. A maximum likelihood approach to estimating speech articulator positions from speech acoustics. *Journal of the Acoustical Society of America*, 100(4):2663–2664, October 1996.

[91] A. Holbrook and G. Fairbanks. Dipthong formants and their movements. *Journal of Speecha and Hearing Research*, 5(1):38–58, March 1962.

[92] P. Hoole. Issues in the acquisition, processing, reduction and parametrization of articulographic data. Technical Report 34, Institute for Phonetics and Speeech Communication, Universitat Munchen, 1996.

[93] H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24:417–441 and 498–520, 1933.

[94] R. A. Houde. *A study of tongue body motion during selected speech sounds*. PhD thesis, University of Michigan, 1967.

[95] F. Jelinek. Five speculations (and a divertimento) on the themes of H. Bourlard, H. Hermansky and N. Morgan. *Speech Communication*, 18(3):242–246, 1996.

[96] F. Jelinek. Speech recognition as a code breaking process. Technical Report no.5, Center for Language and Speech Processing, Johns Hopkins University, February 1996.

[97] M. Joos. Acoustic phonetics. *Language*, 24(supplement: Language monograph 23), 1948.

[98] Michael I. Jordan and Robert A. Jacobs. Hierarchical mixtures of experts and the EM algorithm. *Neural Computation*, 6(2):181–214, 1994.

[99] K. G. Jöreskog. Some contributions to maximum likelihood factor analysis. *Psychometrika*, 32:443–482, 1967.

[100] J. Stephen Judd. Learning in networks is hard. In M. Caudill and C. Butler, editors, *Proceedings of IEEE First International Conference on Neural Networks*, volume 2, pages 685–692, San Diego, CA, 1987.

[101] J. Stephen Judd. *Neural Network Design and Complexity of Learning*. The MIT Press, Cambridge, MA, 1990.

[102] Mark Kac. Can one hear the shape of a drum? *The American Mathematical Monthly*, 73:1–23, April 1966.

[103] R. E. Kalman. A new approach to linear filtering and prediction problems. *Trans. Am.Soc.Mech.Eng., Series D, Journal of Basic Engineering*, 82:35–45, March 1960.

[104] R. E. Kalman and R. S. Bucy. New results in linear filtering and prediction theory. *Trans. Am.Soc.Mech.Eng., Series D, Journal of Basic Engineering*, 83:95–108, March 1961.

[105] Nandakishore Kambhatla and Todd K. Leen. Dimension reduction by local principal component analysis. *Neural Computation*, 9(7):1493–1516, 1997.

[106] H. Karhunen. Uber Lineare Methoden in der Wahrscheinlichkeitsrechnung. *Ann. Acad. Sci. Fenn.*, Ser. A.I. 37, 1947. See translation by I. Selin, The Rand Corp., Doc. T-131, 1960.

[107] J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. *Journal of Information and Computation*, 132(1):1–64, 1997.

[108] R. Koenig, H. K. Dunn, and L. Y. Lacy. The sound spectograph. *Journal of Acoustic Society of America*, 18:19–49, 1946.

[109] Teuvo Kohonen. Analysis of a simple self-organizing process. *Biological Cybernetics*, 44:134–140, 1982.

[110] Teuvo Kohonen. Self-organized formation of topologically correct feature maps. *Biological Cybernetics*, 43:59–69, 1982.

[111] Teuvo Kohonen. *Self-Organization and Associative Memory*. Springer, 3rd edition, 1989.

[112] S. L. Lauritzen and D. J. Spiegelhalter. Local computations with probabilities on graphical structures and their application to expert systems. *J. Royal Statistical Society B*, 50(2):157–224, 1988.

[113] R. B. Lehoucq, D. C. Sorensen, and C. Yang. Arpack users' guide: Solution of large scale eigenvalue problems with implicitly restarted Arnoldi methods. Technical Report from `http://www.caam.rice.edu/software/ARPACK/`, Computational and Applied Mathematics, Rice University, October 1997.

[114] S.E. Levinson and C.E. Schmidt. Adaptive computation of articulatory parameters from the speech signal. *Journal of the Acoustical Society of America*, 74(4):1145–1154, 1983.

[115] A. M. Liberman, F. S. Cooper, D. P. Shankweiler, and M. Studdert-Kennedy. Perception of the speech code. *Psychological Review*, 74:431–461, 1967.

[116] A. M. Liberman, P. C. Delattre, and F. S. Cooper. The role of selected stimulus-variables in the perception of unvoiced stop consonants. *American Journal of Psychology*, 65:497–516, 1952.

[117] A. M. Liberman, P. C. Delattre, F. S. Cooper, and L. J. Gerstman. The role of consonant-vowel transitions in the perception of the stop and nasal consonants. *Psychological Monographs*, 1954.

[118] A. M. Liberman and I. G. Mattingly. The motor theory of speech perception revised. *Cognition*, 21:1–36, 1985.

[119] A. M. Liberman and M. Studdert-Kennedy. Phonetic perception. In R. Held, H. W. Leibowitz, and H. L. Teuber, editors, *Handbook of Sensory Physiology*, volume VIII: Perception. Springer-Verlag, New York, 1978.

[120] R. P. Lippmann. Speech recognition by humans and machines: Miles to go before we sleep. *Speech Communication*, 18(3):247–248, 1996.

[121] R. J. A. Little and D. B. Rubin. *Statistical Analysis with Missing Data.* Wiley, New York, 1987.

[122] L. Ljung and T. Söderström. *Theory and Practice of Recursive Identification.* MIT Press, Cambridge, MA, 1983.

[123] S. P. Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28:128–137, 1982.

[124] M. Loeve. Fonctions aleatoires du second ordre. In P. Levy, editor, *Processus Stochastic et Mouvement Brownien.* Gauthier Villars, Paris, 1948.

[125] E. Lyttkens. On the fixpoint property of Wold's iterative estimation method for principal components. In P. Krishnaiah, editor, *Paper in Multivariate Analysis.* Academic Press, New York, 1966.

[126] J. MacDonald and H. McGurk. Visual influence on speech perception process. *Perception and Psychophysics*, 24(3):253–257, 1978.

[127] J. Makhoul. Linear prediction: A tutorial review. *Proceedings of the IEEE*, 63(4), April 75.

[128] J. D. Markel and Jr. A. H. Gray. *Linear Prediction of Speech*. Springer-Verlag, Berlin, 1976.

[129] A. A. Markov. An example of statistical investigation in the text of 'eugene onyegin' illustrating coupling of 'tests' in chains. In *Proceedings of the Academy of Science, St. Petersburg*, volume 7, pages 153–162, 1913.

[130] H. McGurk and J. MacDonald. Hearing lips and seeing voices. *Nature*, 264:746–748, 1976.

[131] Sebastian Mika, Bernhard Scholkopf, Alex J. Smola, Klaus R. Muller, Matthias Scholz, and Gunnar Ratsch. Kernel pca and de-noising in feature spaces. In *Advances in neural information processing systems*, volume 11, Cambridge, MA, 1999. MIT Press.

[132] E. Muller and G. McLeod. Perioral biomechanics and its relation to labial motor control. *Journal of the Acoustical Society of America*, 78, 1982.

[133] Kevin P. Murphy. Switching kalman filters. Technical report, University of California, Berkeley; Dept. Computer Science, August 1998.

[134] R.D. Nadler, J.H. Abbs, and O. Fujimura. Speech movement research using the new x-ray microbeam system. In *XIth International Conference of Phonetic Sciences*, 1987.

[135] L. Nakatani and J. Schaffer. Hearing words without words: Prosodic cues for word perception. *Journal of the Acoustical Society of America*, 61(1):234–245, 1978.

[136] R. M. Neal and G. E. Hinton. A view of the EM algorithm that justifies incremental, sparse and other variants. In M. I. Jordan, editor, *Learning in Graphical Models*, pages 355–368. Kluwer Academic Press, 1998.

[137] W. Nelson. Articulatory feature analysis – I. Initial processing considerations. Technical report, AT&T Bell Laboratories, 1977.

[138] D.A. Nix, G. Papcun, J. Hogden, and I. Zlokarnik. Two cross-linguistic factors underlying tongue shapes for vowels. *Journal of the Acoustical Society of America*, 99(6):3707–3717, 1996.

[139] David A. Nix and George J. Papcun. Three acoustically predictable factors underlying vowel tongue shapes. *Journal of the Acoustical Society of America*, 97(5):3401, May 1995.

[140] S. J. Nowlan. Maximum likelihood competitive learning. In R. P. Lippmann, J. E. Moody, and D. S. Touretzky, editors, *Advances in Neural Information Processing Systems*, volume 3, pages 574–582. Morgan Kaufmann, San Mateo, CA, 1991.

[141] E. Oja. Neural networks, principal components, and subspaces. *International Journal Of Neural Systems*, 1(1):61–68, 1989.

[142] Harry F. Olson. *Music, Physics and Engineering*. Dover, 1967.

[143] Douglas O'Shaughnessy. *Speech Communication: Human and Machine*. Addison-Wesley, Reading, 1987.

[144] G. Papcun, J. Hochberg, T.R. Thomas, F. Laroche, J. Zacks, and S. Levy. Inferring articulation and recognizing gestures from acoustics with a neural network trained on x-ray microbeam data. *Journal of the Acoustical Society of America*, 92(2):688–700, 1992.

[145] J. Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Mateo, CA, 1988.

[146] J. S. Perkell. Cineradiographic studies of speech: implications of certain articulatory movements. In *5th International Congress on Acoustics*, Liege, Belgum, September 1965.

[147] J.S. Perkell, M.H. Cohen, M.A. Svirsky, M.L. Matthies, I. Garabieta, and M.T. Jackson. Electromagnetic midsagittal articulometer systems for transducing speech articulatory movements. *Journal of the Acoustical Society of America*, 92(6):3078–3096, 1992.

[148] G.E. Peterson and H.L. Barney. Control methods used in a study of the vowels. *Journal of the Acoustical Society of America*, 24:175–184, March 1952.

[149] J. Pierce. Whither speech recognition? *Journal of the Acoustical Society of America*, 46:1049–1051, 1969.

[150] Alan B. Poritz. Hidden markov models: A guided tour. In *IEEE Conference on Acoustics Speech and Signal Processing*, pages 7–13, 1988.

[151] Ralph Potter, George Kopp, and Harriet Kopp. *Visible Speech.* Dover Publications, New York, 1966.

[152] L. Rabiner and B. Juang. *Fundamentals of Speech Recognition.* Prentice Hall, 1993.

[153] L. Rabiner and R. Schafer. *Digital Processing of Speech Signals.* Prentice-Hall, 1978.

[154] L. R. Rabiner and B. H. Juang. An introduction to hidden Markov models. *IEEE ASSP Magazine*, 3(1):4–16, January 1986.

[155] H. E. Rauch. Solutions to the linear smoothing problem. *IEEE Transactions on Automatic Control*, 8:371–372, October 1963.

[156] H. E. Rauch, F. Tung, and C. T. Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450, August 1965.

[157] G. Richard, M. Goirand, D. Sinder, and J. Flanagan. Simulation and visualization of articulatory trajectories estimated from speech signals. In *International Symposium on Simulation, Visualization and Auralization for Acoustic Research and Education (ASVA97)*, Tokyo, Japan, April 1997.

[158] G. Richard, Q. Lin, F. Zussa, D. Sinder, C. Che, and J. Flanagan. Vowel recognition using an articulatory representation. *Journal of the Acoustical Society of America*, 98(5), November 1995.

[159] B. D. Ripley. *Pattern Recognition and Neural Networks.* Cambridge University Press, Cambridge, UK, 1996.

[160] J. Rissanen. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, 11:416–431, 1983.

[161] J. Rissanen. Minimum-description-length principle. *The Annals of Statistics*, 6:461–464, 1985.

[162] Sam Roweis. EM algorithms for PCA and SPCA. In Michael Kearns, Michael Jordan, and Sara Solla, editors, *Advances in neural information processing systems*, volume 10, pages 626–632, Cambridge, MA, 1998. MIT Press. Also Technical Report CNS-TR-97-02, Computation and Neural Systems, California Institute of Technology.

[163] Sam Roweis and Zoubin Ghahramani. A unifying review of linear Gaussian models. *Neural Computation*, 11(2), 1999.

[164] Donald B. Rubin and Dorothy T. Thayer. EM algorithms for ML factor analysis. *Psychometrika*, 47(1):69–76, March 1982.

[165] David E. Rumelhart, James L. McClelland, and the PDP research group. *Parallel Distributed Processing — Explorations in the Microstructure of Cognition (Volume 1: Foundations)*. MIT Press, Cambridge, Massachusetts, 1988.

[166] H. Sakoe and S. Chiba. Dynamic programming algorithm optimization for spoken word recognition. *IEEE Transactions on Acoustics Speech and Signal Processing*, 26:43–49, February 1978.

[167] Terence David Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural Networks*, 2:459–473, 1989.

[168] Terence David Sanger. An optimality principle for unsupervised learning. In David S. Touretzky, editor, *Advances in Neural Information Processing Systems 1*, pages 11–19. Morgan Kaufmann, 1989.

[169] Lawrence Saul and Mazin Rahim. Modeling acoustic correlations by factor analysis. In Michael I. Jordan, Michael J. Kearns, and Sara A. Solla, editors, *Advances in Neural Information Processing Systems*, volume 10, pages 749–755. The MIT Press, 1998.

[170] I. J. Schoenberg. Remarks to Maurice Fréchet's article "sur la definition axiomatique d'une classe d'espaces distanciés vectoriellement applicable sur l'espace de Hilbert". *Annals of Mathematics*, 36:724–732, 1935.

[171] J. Schroeter and M. Sondhi. Techniques for estimating vocal tract shapes from the speech signal. *IEEE Transactions on Speech and Audio Processing*, 2(1 p2):133–150, 1994.

[172] R. H. Shumway and D. S. Stoffer. An approach to time series smoothing and forecasting using the EM algorithm. *Journal of Time Series Analysis*, 3(4):253–264, 1982.

[173] R. H. Shumway and D. S. Stoffer. Dynamic linear models with switching. *Journal of the American Statistical Association*, 86(415):763–769, September 1991.

[174] D. Sinder, M. Krane, S. Chennoukh, G. Richard, J. Flanagan, S. Levinson, S. Slimon, and D. Davis. Fluid dynamic studies of speech production. In *Proceedings 133rd meeting of the Acoustical Society of America*, State College, Pennsylvania, June 1997.

[175] D. Sinder, G. Richard, H. Duncan, J. Flanagan, S. Slimon, D. Davis, M. Krane, and S. Levinson. Flow visualization in stylized vocal tracts. In *Proceedings of the International Symposium on Simulation, Visualization and Auralization for Acoustic Research and Education (ASVA97)*, Tokyo, Japan, April 1997.

[176] L. Sirovich. Turbulence and the dynamics of coherent structures. *Quarterly Applied Mathematics*, 45(3):561–590, 1987.

[177] C. P. Smith. A phoneme detector. *Journal of the Acoustical Society of America*, 23:446–451, 1951.

[178] Padhraic Smyth. Clustering sequences with hidden Markov models. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 9, pages 648–654. The MIT Press, 1997.

[179] Padhraic Smyth, David Heckerman, and Michael I. Jordan. Probabilistic independence networks for hidden markov probability models. *Neural Computation*, 9(2):227–269, 1997.

[180] R. H. Stetson. *Motor phonetics: a study of speech movements in action.* North Holland, Amsterdam, 1951.

[181] S.S. Stevens and J. Volkman. The relation of pitch to frequency. *American Journal of Psychology*, 53:329, 1940.

[182] M. Stone. A three-dimensional model of tongue movement based on ultrasound and x-ray microbeam data. *Journal of the Acoustical Society of America*, 87(5):2207–2217, 1990.

[183] M. Stone and E.P. Davis. A head and transducer support system for making ultrasound images of tongue jaw movement. *Journal of the Acoustical Society of America*, 98(6):3107–3112, 1995.

[184] M. Stone, M.H. Goldstein, and Y.Q. Zhang. Principal component analysis of cross sections of tongue shapes in vowel production. *Speech Communication*, 22(2-3):173–184, 1997.

[185] M. Stone and A. Lundberg. Three-dimensional tongue surface shapes of English consonants and vowels. *Journal of the Acoustical Society of America*, 99(6):3728–3737, 1996.

[186] M. Stone, T. H. Shawker, T. L. Talbot, and A. H. Rich. Cross-sectional tongue shape during the production of vowels. *Journal of the Acoustical Society of America*, 83(4):1586–1596, 1988.

[187] B.H. Story, I.R. Titze, and E.A. Hoffman. Vocal tract area functions from magnetic resonance imaging. *Journal of the Acoustical Society of America*, 100(1):537–554, 1996.

[188] M.A. Thompson and P.E. Robl. X-ray microbeam for speech research. *Nuclear instruments & methods in physics research*, 193(1-2):257–259, 1982.

[189] Michael Tipping and Christopher Bishop. Mixtures of probabilistic principal component analyzers. Technical Report NCRG/97/003, Neural Computing Research Group, Aston University, June 1997.

[190] Michael Tipping and Christopher Bishop. Probabilistic principal component analysis. Technical Report NCRG/97/010, Neural Computing Research Group, Aston University, September 1997.

[191] Michael Tipping and Christopher Bishop. Mixtures of probabilistic principal component analyzers. *Neural Computation*, 11(2):435–474, 1999. Also Technical Report TR-NCRG/97/003, Neural Computing Research Group, Aston University.

[192] W. S. Torgerson. Multidimensional scaling I. Theory and method. *Psychometrika*, 17:401–419, 1952.

[193] W. S. Torgerson. *Theory and Methods of Scaling*. Wiley, New York, 1958.

[194] V. Tresp, S. Ahmad, and R. Neuneier. Training neural networks with deficient data. In J. D. Cowan, G. Tesauro, and J. Alspector, editors, *Advances in Neural Information Processing Systems 6*, pages 128–135, San Francisco, CA, 1994. Morgan Kaufman Publishers.

[195] M. Unser and M. Stone. Automated detection of the tongue surface in sequences of ultrasound images. *Journal of the Acoustical Society of America*, 91(5):3001–3007, 1992.

[196] Y. Vardi, L. A. Shepp, and L. Kauffman. A statistical model for positron emission tomography. *Journal of the American Statistical Association*, 80:8–37, 1985.

[197] T. K. Vintsyuk. Speech discrimination by dynamic programming. *Kibernetka (Cybernetics)*, 4:81–88, January-February 1968.

[198] T. K. Vintsyuk. Element-wise recognition of continuous speech consisting of words from a specified vocabulary. *Kibernetka (Cybernetics)*, pages 133–143, 1971.

[199] Andrew J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Transactions on Information Theory*, IT-13:260–269, April 1967.

[200] Alex Waibel and Kai-Fu Lee. *Readings in Speech Recognition.* Morgan Kaufmann, San Mateo, California, 1990.

[201] John R. Westbury. X-ray microbeam speech production database user's handbook. Technical Report version 1.0, University of Wisconsin, Madison, June 1994.

[202] J.R. Westbury. The significance and measurement of head position during speech production experiments using the x-ray microbeam system. *Journal of the Acoustical Society of America*, 89(4):1782–1791, 1991.

[203] J. Whittaker. *Graphical Models in Applied Multivariate Statistics.* Wiley, Chichester, England, 1990.

[204] J. H. Wilkinson. *The Algebraic Eigenvalue Problem.* Claredon Press, Oxford, England, 1965.

[205] C. Yang and H. Kasuya. Accurate measurements of vocal tract shapes from magnetic resonance images of child, female and male subjects. In *Proceedings of International Conference on Speech and Language Processing (ICSLP'94)*, volume 2, pages 623–626, 1994.

[206] G. Young and A. S. Householder. Discussion of a set of points in terms of their mutual distances. *Psychometrika*, 3:19–22, 1938.

[207] J. Zacks and T. R. Thomas. A new neural-network for articulatory speech recognition and its application to vowel identification. *Computer Speech and Language*, 8(3):189–209, 1994.

[208] Igor Zlokarnik. Adding articulatory features to acoustic features for automatic speech recognition. *Journal of the Acoustical Society of America*, 97(5p2), May 1995.

[209] Igor Zlokarnik. Articulatory kinematics from the standpoint of automatic speech recognition. *Journal of the Acoustical Society of America*, 98(5):2930–2931, November 1995.

Appendix A Speech Processing Background

This chapter provides an extremely brief and necessarily incomplete introduction to speech processing by machines for those unfamiliar with the basics of the field. It is clearly beyond the scope of this thesis to give a comprehensive survey of computer speech processing methods. Below I provide a very general overview of the current paradigms used in speech processing so that the reader may conceptually compare these approaches with the direction outlined in this work. I do not, however, provide details of implementing a recognition system; although unfortunately most of the work in getting a system to actually function properly is in the details. I also give examples of state of the art performance for recognition, synthesis, speaker identification and compression systems. Three standard textbooks, one old but classic (by Rabiner and Schafer [153]) and two newer (by Rabiner and Juang [152] and Deller , Proakis and Hansen [41]), provide very comprehensive introductions to this material. The collection edited by Waibel [200] provides an excellent source of important early papers. Several very good books have been written on speech science in general with less of an engineering focus (see for example [59, 56, 57, 68, 60, 61, 151, 143, 46]).

A.1 A brief historical review

The history of modern speech processing genuinely begins after the second World War. Systems recognizable as the direct predecessors of current systems appeared in the early 1970's. A brief historical review[1] for cultural as much as academic benefit follows:

- **1875** Edison invents the **phonograph**.
- **1876** Bell's **invention of the telephone** [15, 16] inspired by the attempts of Sir Charles Wheatstone to reproduce the speaking machine of Wilhelm Von Kempelen [51].
- **1913** Markov models [129].
- **1915** The first **completely automated transcontinental telephone call** was made from New York to San Francisco ushering in the age of **universal service** in which any telephone

[1] Flanagan's book [59] is the source of many of these items. Thanks also to Nelson Morgan for some items and for the reference to Radio Rex.

user could connect themselves instantly to any other user. By the beginning of the first World War it had become clear that telecommunications would have an enormous impact on society.

- **1922 Radio Rex** toy dog. The first device I am aware of to use speech recognition. A description from David and Selfridge: "It consisted of a celluloid dog with an iron base held within its house by an electromagnet against the force of a spring. Current energizing the magnet flowed through a metal bar which was arranged to form a bridge with two supporting members. This bridge was sensitive to 500Hzacoustic energy, interrupting the current and releasing the dog. The energy around 500Hzcontained in the vowel of the word *Rex* was sufficient to trigger the device when the dog's name was called." [39].

- **1934** The Sherman Antitrust Act and later the **Communications Act** were passed in response to the predatory growth of monopolies such as Western Union and later AT&T. The Communications Act of 1934 created the Federal Communications Commission (FCC) whose mandate was "regulating interstate and foreign commerce in communication by wire and radio so as to make available, so far as possible, to all the people of the U.S. a rapid, efficient, nationwide and worldwide wire and radio communication service..." [36], in other words it made telephone service a *right*. This brought telecommunications technology into the American home for good.

- **1938** The voder and vocoder were invented by Dudley [51] and found important use providing secure voice communications between Roosevelt and Churchill during World War II.

- **1946** ENIAC, an early digital computer.

- **1946** Invention of the **sonograph** [108].

- **1951** A simple **phoneme detector** [177].

- **1952** The **Audry** spoken digit recognizer [40] using formants. [2]

- **1954** Baumann word recognizer [14].

- **1956** Olson and Belar (RCA) spoken digit recognizer [142].

[2] In the early 1950's, driven by telephone company interests, there were several research groups trying to build devices that could recognize spoken digits. Computers were just being invented, and so many systems were still analogue electronics. Davis et al. demonstrated *Audry*, a single speaker isolated digit system in 1952. Audry worked by finding formants and had a surprisingly low error rate of 2%.

- **1957** Bellman's book on **dynamic programming** [18].
- **1958** Wiren-Strubs attempt to use **linguistic features** such as voicing, turbulence, and onset times.
- **1958** Dudley uses a continuous spectrum evaluation rather than formant or other feature tracking [50].
- **1959** Denes, Mathews and Fry phoneme recognizer which used derivatives of spectral energies as well. Denes also introduces a simple bigram **language model** for phonemes which is the first example of non-acoustic information in recognition [67, 44, 45].
- **1962** Review paper in IRE by David and Selfridge [39].
- **1968 Dynamic time warp** first applied to speech by Russian engineer Vintsyuk [197, 198]. Also later by Itakura and Sakoe & Chiba [166].
- **1969 Pierce's caustic letter** criticising speech processing engineers for unprincipled hackery [149].
- **1966-72** Many important theoretical advances in the study of **probabilistic functions of Markov chains** which later came to be called **hidden Markov models** by Baum and colleagues at IDA [12, 11, 13, 10]. Largely unknown in the speech community.
- **1968 Linear predictive coding** (LPC) methods first applied to speech parameterization by Itakura, Atal & Shroeder, Markel and others [6, 128, 127].
- **1971-76** First **ARPA** project for US$15M involving CMU, BBN, SDC, Lincoln Labs, SRI and Berkeley. Resulted in the **Harpy** and **Dragon** systems.
- **1975** John Ferguson at IDA, Jim and Janet Baker at CMU (now Dragon) and Fred Jelinek's group at IBM (now Hopkins) all begin **applying HMMs to speech recognition**.
- **1977** Dempster, Laird and Rubin recognize the general form of the Baum-Welch updates and name it the **EM algorithm** [43].
- **1980** A landmark symposium in Princeton hosted by IDA reviewed current work up till that point [58].

- **1985-88 Standard corpora** begin to be used across research groups. The **TIMIT** database is collected by Texas Instruments and NIST.
- **1986-7** The second ARPA project involving the **Resource Management** (RM),**Wall Street Journal** (WSJ) and **air travel** (ATIS) databases.
- **1988-90** New features such as mel-cepstra (Bridle), PLP (Hermansky), and delta/delta-delta coefficients (Furui) improve the performance of systems on larger and noisier tasks.
- **1990s** Neural nets successfully used as output models in large HMM systems [25].
- **1990s** The popular Cambridge **Hidden Markov Model Toolkit** (HTK) helped to standardize research code and increase access to new players in the field.
- **1990s** Discriminative training, vocal tract normalization, speaker adaptation.

A.2 Basics

A.2.1 What is speech?

Speech is the generic name given to sounds which carry language content. It often has a physical manifestation as a longitudinal compression wave, but for our purposes a speech signal will be a one-dimensional time varying signal. Furthermore, we will make the assumptions that the signal is band-limited, i.e. all of the useful information has its power in a finite frequency range, and that our receivers (ears or synthetic microphones) have a finite acuity. This means that we can *sample* the original time signal which is real and continuous and use instead a vector of values discretized both in time and amplitude. This definition of speech freely admits that we are ignoring many effects such as binaural hearing and other modalities, for example lip reading that are known to be important in human perception; on the other hand humans can speak and understand speech perfectly well over the telephone lines—a situation in which all these assumptions are directly and literally enforced.

A.2.2 Is the time domain representation best?

Above reference was made to "the speech signal", but it was not specified which combination of physical variables are measured to get that signal. The most direct parameter to measure is the air

pressure of the longitudinal wave created when a person speaks. This is informally known as the intensity waveform and all microphones measure something roughly proportional to this or its first derivative by transducing the movement of the air into an electrical voltage using their diaphragms. However, each individual microphone exhibits a slightly different and usually mildly nonlinear transformation from the true pressure signal to its output voltage. Furthermore, ambient noise from other sources and effects of the medium in which the sound is traveling mean that the pressure wave at the microphone is never the same as that which left the speaker's mouth. All this means that from the outset we can never hope to have access to the exact pressure waveform produced by a talker. Nonetheless, microphone intensity signals definitely contain an enormous amount of structure—after all we can record and play back speech that sounds just fine. However, this structure is extremely difficult to extract from the mass of information in the intensity waveform. Consider the waveforms in figure A.1, both of which are myself saying my name and which were recorded on different days but using the same microphone in the same room. These simple figures

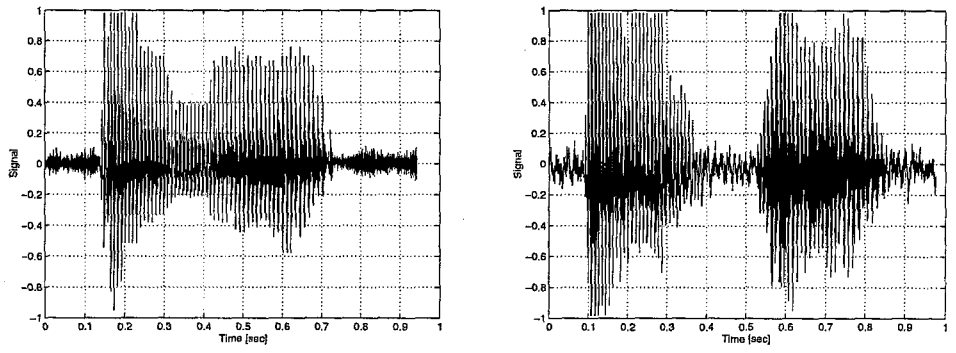


Figure A.1: The author saying his name twice. Notice that the time domain waveforms appear quite different even though the same putative sounds are being spoken by the same speaker using the same microphone in the same acoustical environment.

show that there is quite a lot of *variability* in the raw speech waveform even across the same person saying their name (something they say quite a lot) twice using the same microphone and in the same acoustical environment.

This difficulty motivates us to search for some transformation of the raw intensity waveform into a different representation where the important structure is easier to identify and the enormous amount of variability is reduced. The hope is that a new representation will make it easier to do whatever tasks we are interested in be it speech recognition, speaker recognition or even synthesis and compression.

A.2.3 A simplified model of speech production: excitation plus filtering

Our hope is to extract features of the speech signal that make our tasks easier. A few simple observations about how speech is produced will motivate the features commonly used in speech processing. The first is the relatively basic observation that speech is sometimes harmonic or "song-like" and sometimes noise-like or "breathy." These two modes relate to whether or not an oscillation[3] of the vocal cords is modulating airflow from the lungs and whether or not small constrictions in the airflow path are causing turbulent noise sources. They are called the *voiced* and *unvoiced* excitation modes of speech. Furthermore, in the voiced model there is the simple idea that some people have low pitched voices and others have high pitched voices. Loosely speaking, pitch is related to the frequency of the oscillations which drive the rest of the vocal system. In *atonal* languages like English this pitch information does not carry information about phoneme identity, although it does carry prosodic information about questions or emphasis. Another important observation is that when a person speaks, their vocal tract and articulators form into a particular sequence of configurations that give the sounds of different words their different spectral character. Again loosely, this can be thought of as related to the "modulation" or "shaping" of the excitation. A final observation is that the signal power in speech varies significantly over time and that the timing of these various modulations of pitch, energy and spectral shape is different from word to word or syllable to syllable and is perceptually very important.

These observations lead us to a bare-bones model in which we treat speech sounds as though they were produced by a periodic pulse generator or a white noise source (the vibrations of the vocal cords for voiced speech or the turbulent air flow for unvoiced speech) that was filtered by a simple linear filter (the tube shaped by the articulators). Since the spectrum of the sources are more or less broadband, the spectrum of the produced speech should be the transfer function of the filter. Since the filter is determined by the mouth and articulators, if we can capture the transfer function of the filter in this model, then we will have captured the state of the articulators.

A.2.4 Useful features for speech: spectrograms

The simplified model above suggests that if we could find a way to estimate the voicing (source characteristics), the shape of the vocal tract and articulators (modulator characteristics) and signal

[3]The oscillation of the vocal cords is a complicated nonlinear phenomenon called *Bernoulli oscillation*. The elastic force in the cords works to close the vocal folds, while the airflow from the lungs below can force them open. As the airflow speed increases, the pressure drop between the folds causes them to snap shut.

energy during speech, then we would be in relatively good shape. One more key point to make here is that since the vocal tract and voicing parameters are changing quickly over time as the person speaks, what is needed is a *time-varying* parametric description of these properties and not a single set of parameters for the entire signal. Such a time varying representation is typically achieved by breaking up the signal into segments, called *analysis windows* or *analysis frames* which are short enough that the parameters of interest can be assumed constant within each one. Parameters are then estimated within a window which slides across the signal to obtain a time-varying sequence of parameters.

There are many possible signal features to consider – how to choose between them? Ideally, we would like to select based on which features most reliably capture the articulator and voicing parameters of the signal over time. However, since all we have is the acoustical signal, we do not know what those parameters really are, and thus it is difficult to judge how well a given transform is preserving them. Instead, we can study certain mathematical properties of the acoustical signal which we belive capture in a very crude way some information about the shape of the vocal tract and the behaviour of the source over time. Then we can compare features to see how well they estimate these mathematical properties.

One obvious set of features to consider given the above are the *short-time magnitude spectra* of the signal. The magnitude spectrum tells us which frequencies contain the energy in our signal. The shape of the spectrum of a short window of speech is related to the articulator parameters through the simple models of speech production described above. Am image of such a sequence of short-time spectral features is generically known as a *spectrogram*. The name comes from a traditional method of displaying this information, illustrated in figure A.2. A spectrogram shows signal information in the time-frequency plane. Time runs across the horizontal axis, frequency up the vertical axis. The amount of energy in any region in the time-frequency plane is indicated by the darkness of shading. Such a display is constructed by taking short windows of the original time domain signal and performing (windowed) spectral analysis on each one; the resulting vectors are aligned vertically and stacked one to the right of the previous. This involves taking the discrete-time Fourier transform of each analysis window. It is usual to apply a windowing (such as a Hamming window) to reduce edge effects and also to restrict the length of the frames so that the number of samples is a power of two. These retrictions mean we can use the fast Fourier transform (FFT) rather than a slower generic DFT algorithm.

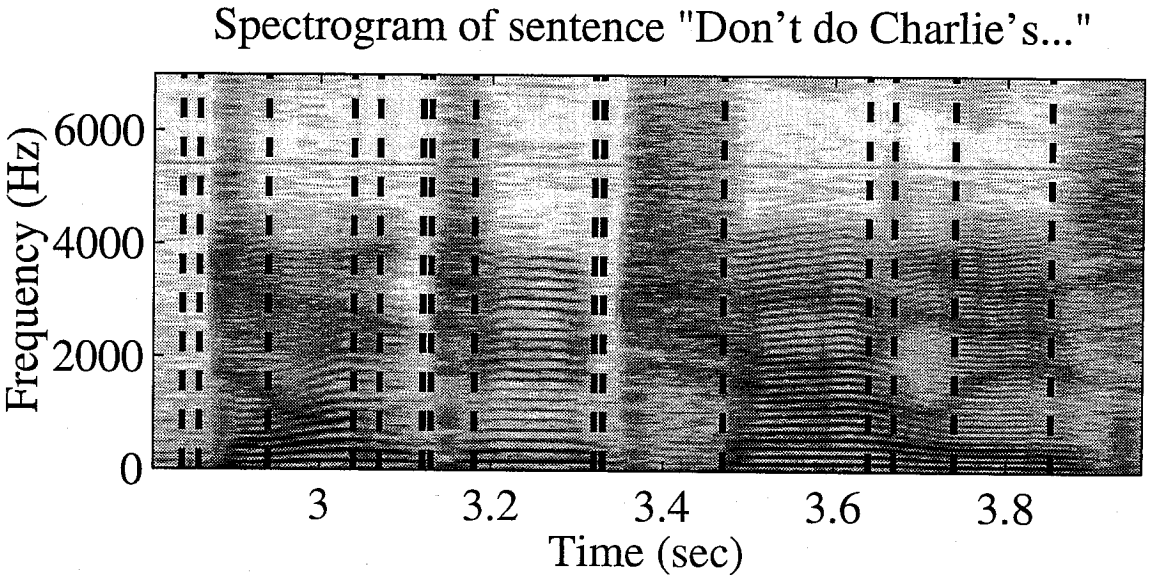


Figure A.2: A spectrogram showing signal information in the time-frequency plane. Time runs across the horizontal axis, frequency up the vertical axis. The amount of energy at any point in the time-frequency plane is indicated by the darkness of shading. Such a display is constructed by taking short windows of the original time domain signal and performing (windowed) spectral analysis on each one; the resulting vectors are aligned vertically and stacked one to the right of the previous. In this example, the vertical bars show the rough segmentation into phonetic units as determined by a forced-alignment procedure (see chapter 3).

A.2.5 Smoothed-spectrogram features: linear prediction and cepstral coefficients

Unfortunately it is often not enough to just take a speech signal, break it up into analysis windows, and take the discrete-time Fourier transform of each one. One reason is that the short-time Fourier transform of a small speech window gives a very noisy estimate of the power at any frequency. We need to find a representation that captures the essential shape of the spectrum and rejects the noise so that we can compare two spectra sensibly. The other reason is that the full short-time spectrum, even if it were not noisy, still contains just as much information as the original intensity signal; and that is too much information for us to process. We need to find a more compact representation of the spectrum. These two ideas are really one and the same: the spectra are typically *smooth* and so we want to filter our noisy estimates and represent them using fewer coefficients.

How can we go about reducing the amount of information used to represent each speech frame while at the same time preserving the basic spectral shape? We require a definition of spectral distance to compare the original spectral shape with the shape implied by the reduced representation. This can be quantified in several ways. One is to consider the *residual energy* between the two spectra. This is the integrated square difference between the *linear* spectra. Because of Parserval's relation this residual is proportional to the energy difference between the original signal and our reconstruction of the (time domain) signal based on whatever reduced representation we select. This difference measure between spectra leads us to a set of coefficients that are optimal in the sense that they minimize the squared difference under certain linear constraints. For minimizing squared error on the linear spectrum or in the time domain, we are led to the *linear predictive coding* (LPC) coefficients and closely the related *cepstral coefficients*. Note that inherent in our distance measure is the assumption that only the *magnitude* of the spectrum is important, and not the *phase*. For short time windows (less than about 40ms) this is certainly true perceptually: a signal reconstructed from only short time magnitude spectra sounds much more like its original than one reconstructed from only phase spectra; furthermore, humans are insensitive to many manipulations or randomizations of short time phase.

A.2.6 Fixed vs. signal dependent transforms

Before we examine the two important transforms mentioned above, it is important to make a brief point about the idea of *fixed* versus *signal dependent* transforms. A fixed transform is one in which the basis vectors of the new (transformed) space do not depend on the original signal.

In fixed transforms, if a sender and a receiver agree on the type of transform beforehand, then the sender can take the original signal, transform it, and send some or all of the coefficients to the receiver. The receiver can then reconstruct the signal based solely on the values of the coefficients. However, with *signal dependent* transforms, the sender must first communicate the basis vectors to be used for reconstruction in addition to the coefficients. Here we are considering fixed transforms. But it is worth remarking that if we were to permit the transforms to be signal dependent, then the *Karhunen-Loeve Transform* (also known as Principal Component Analysis, see chapter B) would be optimal in terms of residual energy. This means that the best we could ever do for minimum squared reconstruction error would be to project our speech signal onto the first few principal components of the distribution from which the signals were drawn. However, we are investigating here only fixed transform approaches. (Although it is true that a fixed PCA basis could be chosen by prior training on a large databse.)

A.2.7 Linear predictive coding (LPC)

Linear predictive coding (LPC) [128, 127] is a model of a discrete-time signal s_t in which we try to approximate each sample by a linear combination of some previous samples:

$$s_t \approx a_1 s_{t-1} + a_2 s_{t-2} + a_3 s_{t-3} + \ldots \tag{A.1}$$

The above approximation is known as an *autoregressive* (AR) signal model. The number of previous samples on which the current sample depends is the *order* of the model. We make up the difference between this approximation and the actual signal by a scalable correction function u_t called the *source* function:

$$s_t = a_1 s_{t-1} + a_2 s_{t-2} + \ldots + a_k s_{t-k} + G \cdot u_t \tag{A.2}$$

If the source function could magically take on the negative of our prediction error at any time, then the equality above would indeed be exact. We can think of the source function as the input to a system and the actual signal as the output. What kind of system lies in between? Let us take the *z-transform*:

$$S(z) = a_1 \frac{S(z)}{z} + a_2 \frac{S(z)}{z^2} + \ldots + a_k \frac{S(z)}{z^k} + G \cdot U(z) \tag{A.3}$$

Thus the transfer function from source u_t to output s_t is known as an *all-poles filter* whose impulse response are the coefficients a_k and whose frequency response is given by:

$$H(z) = \frac{S(z)}{G \cdot U(z)} = \frac{1}{1 - a_1 z^{-1} - a_2 z^{-2} - \ldots - a_k z^{-k}} = \frac{1}{A(z)} \tag{A.4}$$

Figure A.3 shows a block diagram of this model. For speech signals, the assumption of an all-

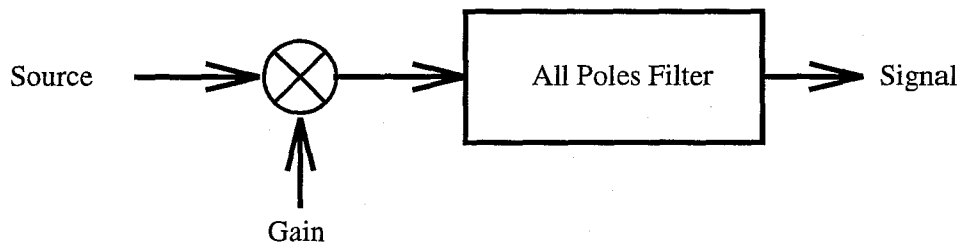


Figure A.3: Linear predictive coding model. Each sample of an incoming signal is modeled as a linear combination of some fixed number of past samples plus a driving noise or "innovation" term.

pole filter is not a bad one because of the nature of the process that generates speech.[4] Thus, the source function need only be a relatively small amplitude "noise" source, representing the small and random prediction errors. Following our simple model of speech production, we define the source as either an impulse train for voiced speech segments or white noise for unvoiced speech. Typically only a few poles are required in the filter since the usefulness of far away (long ago) samples for predicting the current sample point is quite small. A good rule of thumb is that samples up to 1.5ms in the past help in predicting the current sample, but not earlier.

To see how this works, at least to some degree, consider a concrete example: In figure A.4 I have shown a window of a speech signal that is 512 samples long (which at the sampling rate of 22kHz is about 25ms). It has been multiplied by a Hamming window. Its spectrum is also shown. Figure A.5 shows the same speech sample with the spectrum reconstructed by an LPC model of order 12. To do this, I have estimated the best 12 pole filter, kept only the 12 coefficients $a_1 \ldots a_{12}$, and reconstructed the spectrum based on these 12 numbers.

Linear predictive coding coefficients are used extensively in academic and commercial speech systems. The other extremely common features are the cepstral coefficients (introduced in the next

[4]Strictly speaking, if we define the "transfer function" of the vocal tract to be the ratio of volume velocity at the output to volume velocity at the input then there physically cannot be any zeros. However, it is important to acknowledge that it still may be empirically useful to model the observed spectra using a small number of both poles and zeros to give a better fit.

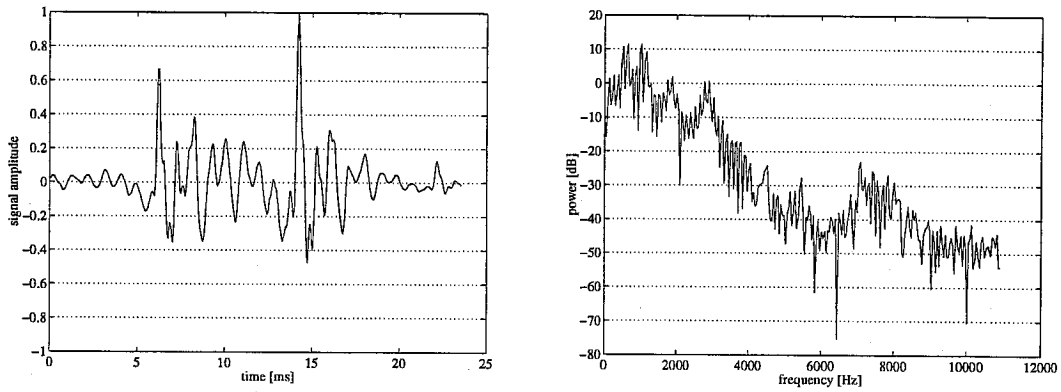


Figure A.4: A short window of speech (left) and its corresponding spectrum (right). The speech has been windowed with a Hamming window to reduce edge effects in the spectral analysis.

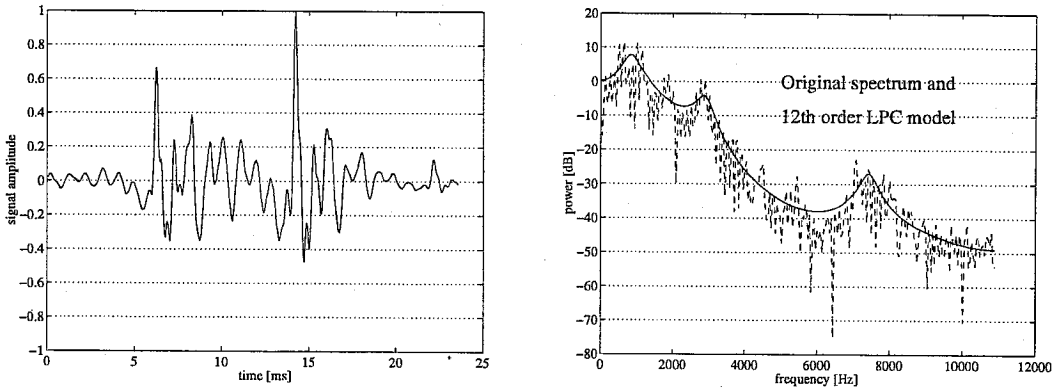


Figure A.5: The segment of speech from figure A.4. The right panel shows the spectrum obtained by linear predictive coding after retaining only a few coefficients. A 12^{th} order LPC model was used. The spectral fit is quite good in terms of residual energy. Compare with figure A.6.

section) which in some cases give superior performance and can be mathematically easier to work with.

A.2.8 Cepstral coefficients

The cepstrum[5] transform (see for example [80]), while having some complicated associated mathematics is really based on a very simple idea: the short time log magnitude spectra that we are trying to capture are quite noisy and we only want to capture their essential shape. So let us get rid of the noise by "low-pass" filtering that shape. Low-pass in which sense? In the sense that we only want to let the log magnitude spectrum change slowly across frequency. That is, we want to eliminate large changes in log magnitude over small frequency intervals. What we want to do is "smooth out" the spectrum in exactly the domain as one sees it in all the figures above; namely log magnitude.

In order to achieve this, we do what at first appears to be a crazy thing: *we take the spectrum of a spectrum.* That is, we take our original signal, compute its log magnitude spectrum and then take the spectrum of *that,* chop off all the upper coefficients (i.e. "low-pass" filter) and then inverse transform to reconstruct our original log magnitude spectrum. The domain in which we are when we chop coefficients is called the *cepstrum.* More formally, the cepstrum of a real signal $s(t)$ is defined as:

$$C(q) = F^{-1}\left(log|F\left(s(t)\right)|\right) \tag{A.5}$$

where $F(\cdot)$ denotes the discrete-time Fourier transform and $F^{-1}(\cdot)$ denotes its inverse. The variable q denotes the axis of the cepstral domain, called *quefrency*; which is something like the rate of change over frequency analogous to regular frequency ω being the rate of change over time. Notice that the cepstrum is real and also symmetric in q since the log of the magnitude spectrum is real. Finally, notice that we have taken the inverse transform F^{-1} after the log, and not the forward transform, but they differ only by a scaling factor. The cepstrum is essentially trying to

[5]The name *cepstrum* as well as other terms from the literature such as *liftering* and *quefrency* come from the amusing reversal of the beginning of the words *spectrum, filtering*, and such. This is because the cepstrum, as we will see, is an inverse transform of a transform.

approximate the log magnitude spectrum using a truncated Fourier series:

$$log|F(\omega)| \approx \sum_{k=-K}^{K} c_k e^{-jk\omega} = log|F'(\omega)|$$

where $c_k = c_{-k}$ are the real cepstral coefficients and $F'(\omega)$ is the cepstral approximation to the spectrum.

The hope of the cepstrum transform is that in the cepstral domain there will be some power at low coefficients representing the slowly varying (in frequency) component of the spectrum and some power at high coefficients representing spectral noise (and the fine spectral structure due to voicing). If we only keep the lowest few components and set all the rest to zero, then when we reconstruct the log magnitude spectrum it should be "smoother." This truncation process (or more generally a windowing process in which higher cepstral coefficients are de-emphasized) is called *liftering*. Shown in figure A.6 are the cepstral coefficients for the speech segment of figure A.4 and the reconstructed spectrum obtained by keeping only the first 12 cepstral coefficients and setting the rest to zero. One can see that the cepstral coefficients have a bump at low quefrency

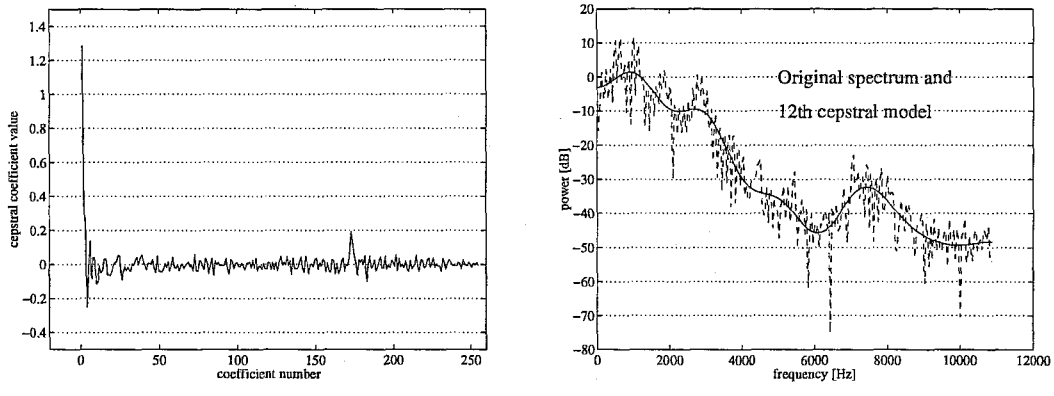


Figure A.6: The coefficients resulting from cepstral analysis on the segment of speech from figure A.4. Notice the smaller second spike at higher coefficient values. The right panel shows the spectrum obtained by keeping 12 cepstral coefficients and reconstructing.

and another bump at high quefrency. The high quefrency bump is the pitch excitation of the voiced speech. One can also see that the reconstructed spectrum satisfies the original goal of being "smoothed" in the log magnitude domain. Notice that this reconstruction is technically "worse" than the LPC reconstruction in terms of residual energy but captures the essential character of the spectrum just as well (compare with figure A.5). The cepstral coefficients are also easier and faster

to compute in some cases.

A.2.9 Spectrum-scale perceptual effects: mel and Bark scales

Why did we use a logarithmic scale for frequency in working with the cepstrum above? The reason is partly one of engineering history but also partly that this scale has been found to be perceptually more relevant than the linear frequency scale. Two important effects from human perceptual studies motivate this claim. The first observation is that humans are not sensitive to frequency information below about 200Hz or above about 16kHz for many speech processing tasks including speaker recognition. The upper limit does not typically help us since our Nyquist rate is normally far below it. But the lower limit suggests that we might be able to high-pass all of our short time spectra above 200Hz before computing our cepstral coefficients. This would save computation and possibly add some robustness against external noise sources (fans, 60 cycle hum, etc.). The second observation is that humans pay relatively more attention to the lower frequencies than the higher ones. Roughly, frequencies up to 1kHz are "weighted" in importance linearly while above about 1kHz the perceptual importance falls off logarithmically for a wide variety of psychophysical tasks and measurements. This suggests that we should "remap" our spectra to a more perceptual frequency scale before extracting coefficients. Many such perceptual scales have been proposed, the two most common of which are the *mel* [181] and *Bark* (named after the German acoustician Barkhausen) scales which are based on the results of psychophysical pitch matching and critical band experiments. The scales can be defined by simple mathematical approximations to the measured psychophysical curves. For example, the following transformations (all frequencies are in Hz) are common:

$$f_{mel} = 2595 \log_{10}(1 + f/700) \tag{A.6}$$

$$f_{Bark} = 13 \arctan(0.76 f/1000) + 3.5 \arctan((f/7500)^2) \tag{A.7}$$

These relations are just fits to human psychophysical curves; however, researchers consistently report significant improvements using the Bark and mel scales rather than linear scales. When the mel scale warping is used prior to extracting the cepstral coefficients, the resulting features are known as *mel cepstra.* Figure A.7 shows the shape of these two perceptual scales compared with logarithmic and linear shapes.

Another very common signal processing step in speech analysis is to first pass the raw sam-

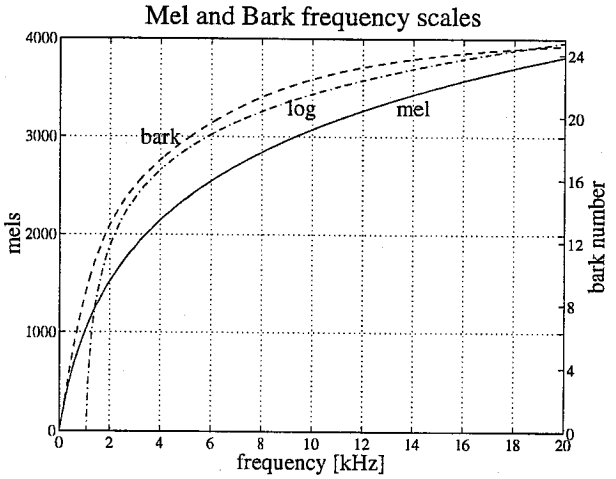


Figure A.7: The *mel* and *Bark* perceptual frequency scales. The solid line shows the mel scale, the dashed line shows the Bark scale and the dotted line shows a logarithmic shape for comparison. These scales are all fits to psychophysical curves obtained in pitch matching experiments.

pled speech signals through a *pre-emphasis* filter. This is a high pass system which flattens the spectrum. The main motivation for it is that speech typically has a $1/f^2$ amplitude spectrum which introduces some *spectral tilt* and we want to flatten that out. It has been found to improve the robustness of recognition tasks and does not take very much time. Typically a time domain smoothing rule

$$s(n) \Leftarrow s(n) - \alpha s(n-1) \tag{A.8}$$

is used, corresponding to a discrete-time transfer function of

$$H(z) = 1 - \alpha/z \tag{A.9}$$

which is a single pole high pass filter. A usual value for α is 0.95 which boots power at the Nyquist frequency by 32dBover DC power—quite a significant effect.

A.2.10 Distance measures between spectral features

Central to any pattern recognition application is the notion of the *distance* between two feature vectors. A common choice is the L^2 norm or Euclidean distance, namely the sum of squared differences between corresponding components of the two vectors. However, this is sometimes

not a very meaningful distance measure. In the present case we are interested specifically in two kinds of vectors: vectors of LPC coefficients and vectors of cepstral coefficients. What is an appropriate measure of "distance" between two LPC vectors or two cepstral vectors?

The LPC coefficients were attempting to minimize the residual energy between the true (linear) magnitude spectrum of the speech frame and the LPC spectral approximation. Thus it seems reasonable to compare two LPC vectors by comparing the residual energies between each of their reconstructed spectra and some "true" spectrum. Specifically, it can be shown (see for example page 118, Exercise 3.6 in [152]) that the residual energy between the reconstructed spectrum of an LPC vector $\mathbf{a}$ and the true spectrum is given by:

$$E = \mathbf{a}^T \mathbf{R}_x \mathbf{a} \tag{A.10}$$

where $\mathbf{R}_x$ is a Topelitz matrix formed from the autocorrelation sequence of the true signal x. Thus, the "distance" between two LPC vectors is the ratio of their residual energies:

$$D(\mathbf{a}, \mathbf{b}) = \frac{E(\mathbf{a})}{E(\mathbf{b})} = \frac{\mathbf{b}^T \mathbf{R}_x \mathbf{b}}{\mathbf{a}^T \mathbf{R}_x \mathbf{a}} \tag{A.11}$$

If we choose $\mathbf{a}$ to be the LPC vector computed on the true signal x, then $E(\mathbf{a})$ is minimized, and hence the distance above will always be greater than unity. This metric was originally proposed by Itakura and Saito and is sometimes known as the Itakura-Saito distortion between two LPC vectors. It is mathematically complex and can be difficult to understand throughly, but I have presented here the intuitive idea behind it.

Following our distance measure between LPC vectors, we can compare two cepstral vectors by comparing the residual energy between each of their reconstructed spectra and some "true" spectrum. As noted above, the cepstrum is essentially trying to approximate the log magnitude spectrum using a truncated Fourier series:

$$log|F(\omega)| \approx \sum_{k=-K}^{K} c_k e^{-jk\omega} = log|F'(\omega)| \tag{A.12}$$

where $c_k = c_{-k}$ are the real cepstral coefficients and $F'(\omega)$ is the cepstral approximation to the spectrum. Imagine that we expressed the true spectrum by its full (infinite) Fourier series with

coefficients a_k. Then by Parseval's theorem, we can write the residual energy as:

$$E = \frac{1}{2\pi}\int_{\omega} |logF(\omega) - logF'(\omega)|d\omega \tag{A.13}$$

$$= \sum_{k=-\infty}^{\infty} (c_k - a_k)^2 \tag{A.14}$$

where c_k are defined to be zero outside $-K \ldots K$. Similarly, for another cepstral vector b_k, the residual energy between its reconstructed log spectrum and the true log spectrum would be:

$$E = \sum_{k=-\infty}^{\infty} (b_k - a_k)^2 \tag{A.15}$$

Thus the difference between the residuals of b_k and c_k is given by:

$$D(b,c) = \sum_{k=-\infty}^{\infty} (b_k - c_k)^2 \tag{A.16}$$

which is just the normal Euclidean distance that we are used to. It is interesting to note that the distance between b_k and c_k does not depend at all on what the "true" spectrum a_k was. These two facts can make the cepstral vectors easier to work with and to visualize than the LPC vectors in which our intuitive sense of squared norm distance does not properly apply.

A.3 Spectral pattern matching

A.3.1 Pattern matching for static sounds: spectral peak locations

An influential study in the early history of speech processing was conducted by Peterson and Barney [148] who collected statistics on 76 speakers saying various steady state vowels at the 1939 World's Fair in New York. They noted the frequency and amplitude of the first three formants. When plotted in various ways (for example F2 vs. F3), the different vowels fall into distinct regions of the formant space. This encouraged researchers because it implied that one only had to accurately estimate formant frequencies in order to classify vowels. While this is true for steady state sounds, it was soon discovered (for example see [91]) that during the production of continuous speech the formant frequencies of the various vowels move considerably and depend on context. These *co-articulation* effects led researchers to investigate dynamic pattern matching techniques which attempt to perform pattern recognition on the time evolution of spectral infor-

mation rather than on a frame by frame basis.

A.3.2 Pattern matching for sequences of features: dynamic time warping (DTW), hidden Markov models (HMMs) and others

Motivated by the realization that steady state formant measurements were not representative of continuous speech, researchers began looking at the spectrogram representations of words. Amazingly, it was found that after condsiderable practice (see for example [151] and many classes on the subject talk by Victor Zue in the 1980's and 1990's) people could be trained to "read" spectrograms. This led to a set of pattern classification algorithms which essentially tried to match a spectrogram template to the incoming spectrogram.

Two fundamental difficulties have to be overcome for this simple approach to work at all. The first is the problem of *time-warp*. Although the spectral patterns of many instances of the same word often look similar, they do not arrive at the same rate – in some cases people will say a word faster or slower than in others. Nor is the speeding up or slowing down always uniform across the word. Certain parts of the word may be selectively sped up or slowed down. An elegant solution to this problem was found in the form of the *dynamic time warping* (DTW) algorithm. DTW is an application of Bellman's dynamic programming [18] to spectrogram matching. It provides a method by which an incoming spectrogram may be locally stretched and squished (in time) to optimally match a template spectrogram (see [197, 198, 166]).

The second difficulty is that the spectral energy patterns in multiple instances of the same word exhibit some variability across frequency. Occasionally spectral energy appears in certain frequency bands in which power is normally absent for a given word. Conversely, spectral energy is sometimes lacking in frequency bands that normally contain power. Researchers tried to model this variability using a *probabilistic* spectrogram template. Again, an elegant solution combining the probabilistic template in frequency with dynamic time warping was found in the application of *hidden Markov models* [12, 11, 13, 10, 152, 41] to speech analysis.

A.3.3 Other acoustic cues: pitch, voicing

Short-time spectral shape and signal energy are the features captured by the spectrogram and smoothed-spectrogram representations of speech such as LPC and cepstral coefficients. However, there are several other important features of the original acoustic waveform that are perceptually

very important. Two of these are *voicing* and *pitch*, discussed earlier.

The voicing signal, mentioned in section A.2.4 above, represents the state of the vibration of the vocal chords. Generally the voicing signal indicates one of three modes. During silences the vocal chords are not vibrating and there is too little airflow through the mouth to create any turbulent sources at constriction points. The result is that little or no sound is produced. During *unvoiced speech* the vocal chords are not vibrating but airflow through constrictions at the tongue, lips or teeth creates turbulent sources which excite the remainder of the vocal tract. The result is breathy or fricative sound. Finally, in *voiced speech* the chords are oscillating producing a approximately periodic impulsive excitation that is subsequently "coloured" or shaped in frequency response by the vocal tract. Voicing can provide imporant perceptual clues about the identity of words and sounds, for example many consonants are distinguished mainly by their *voice onset time*.

Another acoustic feature of interest is *pitch*. Pitch is the term for the rate of excitation of the vocal cords. voiced speech. It is the fundamental frequency of voiced sounds and is thus sometimes known as F0. Tracking pitch can be important for improved signal analysis; for example it is easy to confuse peaks due to resolved pitch harmonics with formant peaks especially for female speakers. In English, pitch also carries information about emphasis, questions and parenthetical statements. In tonal languages pitch contours actually carry information about the identity of words and thus pitch estimation is even more desirable. Traditional short-time spectral features such as cepstral coefficients do not capture this.

Automatic algorithms for estimating pitch and voicing exist, but they are typically heuristic and have performance which varies considerably from database to database. Currently, most systems are unable to make use of such cue. Improved estimation of these and other acoustic cues and effective integration of this information into recognition and compression algorithms is one clear direction of research for improving existing speech processing techniques.

A.3.4 Extra-acoustic cues: language models, lipreading

Speech processing may also benefit from information not present in the acoustic signal. One such piece of information comes from *language models* which attempt to harness the strong statistical patterns in word co-occurrences. The most basic language models measure the unigram, bigram or trigram frequency of words, word doublets or word triplets from enormous spoken text corpora and use these frequencies to evaluate the likelihood of upcoming words in a recognition task.

(This is generally difficult to do because of the so-called *Zipf's Law* property of English which means that the most common words occur much more frequently than the less common words. Figure A.8 shows the frequency ranking and cumulative frequency of the most common words in conversational speech as measured on the Switchboard database.[6])

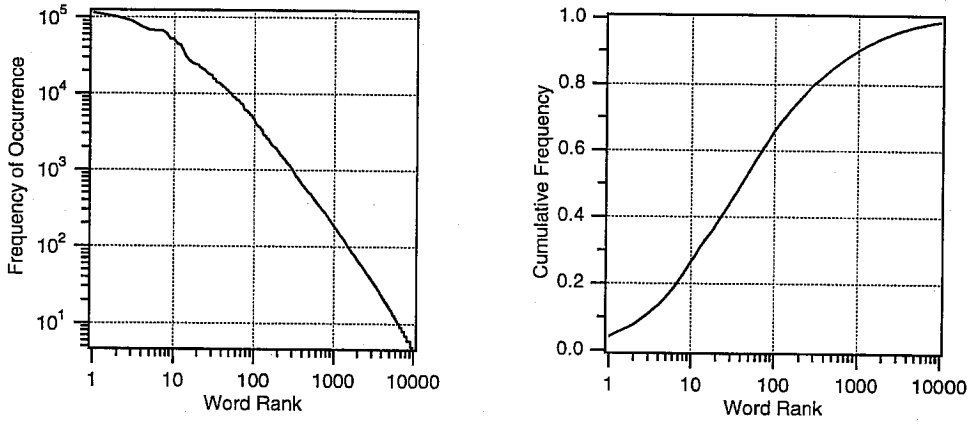


Figure A.8: The frequency of common words in spoken language. The left panel shows the frequency of word occurrence compared with word rank indicating that less common words are logarithmically less likely than the more common words in accordance with *Zipf's Law*. The right panel shows the cumulative frequency compared with word rank indicating that even in conversational English roughly 1000 words cover in excess of 90% of the tokens.

Another non-acoustic source of information from which speech processing may also benefit is visual input. Human lipreaders have the ability to infer much of what is being said from lip movements alone. But even normal listeners rely heavily on lip information in noisy environments. As video cameras on workstations become more and more common, speech engineers are looking for ways to exploit this new modality to do audio-visual recognition.

A.4 State of the art: How well are we doing?

State of the art performance for speech recognition still lags far behind that of humans for all tasks from conversational speech to nonsense syllable tasks. The systems are also extremely brittle – any small deviation from "normal" conditions causes system performance to degrade dramatically. Even changes that are virtually undetectable to humans such as switching the microphone or changing the sampling rate of the audio signal will cause several-fold increases in the error

[6]Thanks to Steve Greenberg for these figures.

rates of most systems. Furthermore, humans maintain high performance even in the presence of considerable noise while recognition systems again suffer considerably.

To quantify system performance it is common to quote a statistic known as the *word error rate* which is the percentage of words in the reference (true) transcript that the artificial system incorrectly reported. Insertions, substitutions and deletions of words are all counted as errors (though usually tabulated separately). Competitions on standard databases are administered by the National Institute of Standards and Technology (NIST) on a roughly biannual basis. The current evaluation, called Hub4e, uses two databases which contain conversations between two parties over the telephone. These are challenging tasks because they include all of the disfluencies (such as filler words and restarts) that occur in spontaneous speech. They also involve two unknown speakers and an unknown telephone channel distortion. The results are generally quite poor, with most systems getting between 45% and 50% of the words wrong. The systems also run several hundred times slower than real-time on extremely powerful computers. In comparison, humans who listen to the conversations only a few times (making their speed a few times real-time) obtain error rates of only a few percent on the same databases. Table A.4 contains the numerical results of this evaluation on the two conversational large vocabulary databases called Switchboard (SB) and Call Home (CH).

System Name	Word Error Rates (%)		
	Switchboard	Call Home	Average
BBN	35.5	53.7	44.9
Boston University	41.5	58.2	50.1
Carnegie Mellon University (ISL)	35.1	54.4	45.1
Cambridge University – HTK	39.2	57.6	48.7
Dragon Systems	39.9	57.4	48.9
SRI	42.5	57.5	50.2

Table A.1: Word error rates for state of the art systems in the most recent NIST competition for large vocabulary continuous speech recognition on two conversational telephone speech databases. Results are from the NIST web site [63].

For speech synthesis it is difficult to quantify performance of a system in any way other than having a large group of listeners score generated utterances according to their "naturalness."[7] Thus, by way of example of state of the art synthesis, I have included generated files from several major synthesizers all saying the same sentence. The sentence was chosen to be one which

[7]Although there is much current interest in designing automatic algorithms to predict the results of such human evaluations on synthetic speech (i.e. automatic scoring of synthesis), there is currently no reliable standard.

could not contain pre-memorized phrases, since many synthesis systems have a library of pre-synthesized common phrases which they insert into longer synthesized sentences. The sentence reads: *On Thursdays I polish my vulcanized uncle.* Sound A.4.1 has the example.

Audio file A.4.1 (`track 2`) *Synthesis examples for the sentence "On Thurdays I polish my vulcanized uncle."*

These example results are indicative of a general trend in speech synthesizers to be intelligible but not at all natural sounding. In general, speech recognition and speech synthesis are still far behind human performance. Since we use these skills daily, our expectations for artificial systems are quite high.

Speaker identification and speech compression on the other hand are quite advanced. State-of-the-art speaker identification systems compare well to human performance. With hundreds to thousands of users, they can operate at a false rejection rate of 1% and a false acceptance rate of 0.1%. They work for both males and females in the presence of limited noise and do not require a specific phrase to be repeated for identification. Speech compression systems have advanced to the point where *transparent coding* is the de-facto standard. This means that on average listeners cannot tell the difference between the coded version and the original version of an utterance. For speech-only compression (as opposed to general audio including music, etc.) transparent coding can be achieved with a rate as low as 2 kilobits per second, 32 times more compact than a standard telephone channel.

Audio file A.4.2 (`track 3`) *Speech compression examples using various standard coders.*

A.4.1 Overview of current state of the art recognition system architecture

All modern speech recognition systems are based on statistical pattern recognition techniques. The systems are trained on a large number of examples of speech that have been correctly labelled by human transcription. During training, internal parameters of probabilistic models are tuned by algorithms that attempt to maximize how well the models fit the training data. Systems can then be evaluated by showing them new utterances not part of the corpus used to fit parameters and measuring their recognition accuracy. Often part of the training data is reserved for pre-testing diagnosis of the system—known as *validation*. This helps to avoid an *over-fitting* effect in which the models memorize the patterns of training data but do not generalize well to new cases.

Modern system architectures are quite complex and it is certainly beyond the scope of this introduction to describe them in detail. Figure A.9 shows a schematic of the basic structure shared by most modern speech recognition systems. An extremely brief description is included, but the reader is refered to the texts [152, 41] for extremely complete reviews. A language model examines the current context of words and proposes several choices for the next word based on co-occurrence statistics. Various pronounciations of these words are looked up in a phonetic dictionary producing candidate phoneme strings. Each phoneme string generates a hidden Markov model composed of the concatenation of the appropriate basic models (e.g. phones or triphones). Each of these generated HMMs gives a probability score to the observed sequence of acoustic features from the incoming speech. This score is combined with a score from the language model indicating the probability of the corresponding word sequence. The choice with the highest combined acoustic and language model score is selected as the result of the recognition process.

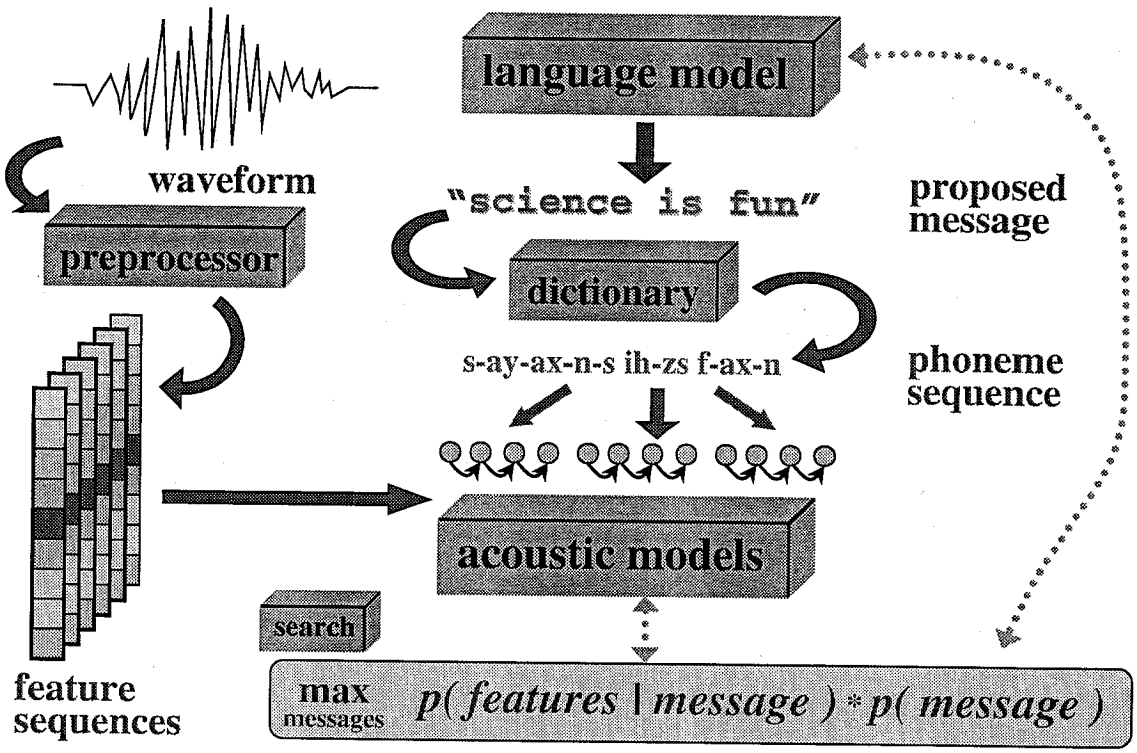


Figure A.9: A schematic of the basic structure shared by most modern speech recognition systems. A language model examines the current context of words and proposes several choices for the next word based on co-occurrence statistics. Various pronounciations of these words are looked up in a phonetic dictionary producing candidate phoneme strings. Each phoneme string generates a hidden Markov model composed of the concatenation of the appropriate basic models (e.g. phones or triphones). Each of these generated HMMs gives a probability score to the observed sequence of acoustic features from the incoming speech. This score is combined with a score from the language model indicating the probability of the corresponding word sequence. The choice with the highest combined acoustic and language model score is selected as the result of the recognition process.

Appendix B Machine Learning Background

In this chapter I briefly review the mathematical techniques in machine learning and signal processing that have found important application in this thesis.[1] As in chapter A it is of course impossible to provide a comprehensive review. However, several textbooks have been published which do this from a variety of perspectives. The recent reviews by Bishop and Ripley [20, 159] focus on pattern recognition and statistical theory. The classic volume by Hertz, Krogh and Palmer [81] still provides one of the best available introductions to associative memories and Hopfield networks with a statistical physics focus. The readers edited by Anderson [4] and the original PDP volumes [165] provide essential original source reading for any any newcomer into the field of machine learning.

See section 1.2 for a description of what is meant by machine learning and the motivation for its use in sensory computation problems. Machine learning has been applied (and is applied in this thesis) to pattern classification and recognition, pattern completion, pattern synthesis, and data re-representation—coding for transmission, compression or easier computation.

Section B.1 presents a very informal introduction to systems and signals for the reader not familiar at all with signal processing. Section B.2 presents a more technical machine learning introduction to the models most heavily used in this work known as linear-Gaussian models.

B.1 Systems, signals and transforms

A *signal* is nothing more than a name for **a reasonably well behaved function**.[2] Typically in an engineering context we think of the independent variable (domain) of a signal as time (sometimes space) and the dependent variable (range) as some **physically meaningful** quantity such as voltage or current or pressure.

[1] Thanks to the members of Pietro Perona's vision research group for helpful comments and discussions on this chapter. Some material for this chapter was taken from an early draft of [163] which was authored solely by myself. Discussions with my future co-author Zoubin Ghahramani have improved these sections, but I have omitted any sections added after this became a joint work; for these refer to the published work.

[2] Recall that a function is just a rule that takes one or more numbers as input and produces one or more numbers as output. The rule must be non-ambiguous (for each valid input there is one and only one output) and deterministic (the same input always gives the same output). The *domain* of a function is the set of inputs for which the rule works, i.e. the set of inputs for which we can work out what the output should be. The *range* is the set of outputs that the domain is mapped into, i.e. the set of all possible outputs the rule can produce.

For a function to be "well behaved" can mean many things, but loosely we are interested in functions that are not too "jumpy" or "wiggly" and do not "blow up" anywhere.

When we speak of a *transform* we almost always mean **an operation that converts one function into another**. For example, differentiation and integration are familiar operations that are perfectly valid transforms. A *system* is any (usually physical) process that produces an output signal based on an input signal. Now here is a key point: Generally we like to model the action of a **system** by inventing a **transform** that we believe approximates well the relationship between the input and output of the system. Usually we think of the original signal as the "input" to the real system and the transformed signal as the "output" of the real system. In some cases we are very interested in the discrepancies between the action of the real system and our model transform, for example in control theory. But at other times we are willing to accept our model as adequate from the beginning.

There are two distinct views of transforms, and it is the use of the same word, *transform*, to refer to both views which sometimes causes confusion. One view is the modelling view presented above in which the use of transforms is to approximate the input-output behaviour of some real world system(s). In this case, the transforms do things like filtering, integrating, and modulating; in general they convert one signal in time (say f_t) to another, different signal in time (say g_t). A transform in this sense is *a mathematical model for a system.*

Another view of transforms is as a way of re-representing a signal in another space. In this case, transforms are used to convert the information in a signal into a different form, but it is fundamentally the same signal. For example we might use a simple rotational transformation to re-represent the point (x, y) as another point (u, v) in a different set of axes. A transform in this sense is *a converter from one representation of a signal to another.* There are many such transforms used by signal processing practitioners but in this thesis we will be primarily concerned with two extremely common transforms that do nothing but rotate the coordinate system of the original data. The first is the *discrete fourier transform* (DFT) of the signal. This transform converts a finite length discrete time series into another vector of the same length with contains information about the power in the original signal at various temporal frequencies. The second is the *Karhunen-Loève transform* (KLT), also known as principal component analysis (PCA) [93, 106, 124]. Unlike the DFT which always applies the same rotation to a fixed length vector, the PCA basis depends on the orignal dataset. The new axes are those which decorrelate the original data to second order. Thus, PCA cannot be definite for a single input time series. One must have a collection of

time series from which one computes the data-dependent decorrelating transform which is then identically applied to each one.

B.1.1 Why transform?

Signal transformation is common in many fields and applications. It is clear why we would want to use transforms in the modelling sense described above since we often have physical systems to study and we need mathematical descriptions of them. But why would we ever want to use transforms in the re-representation way? It turns out that some interesting features or facts about a signal can be much easier to discover in some representations than in others. The original time or space domain representation is not always the most informative.

Broadly speaking, there are two classes of transformations to consider: those that are *invertible* and those that are not. Invertible transforms preserve all the information but convert it into a different form, while non-invertible transformations throw out some information. While the latter class may sound undesirable at first, it turns out that we will almost always want to destroy some information either in the process of taking the transform, or afterwards as a separate operation. In fact making it easy to throw out the right information is very often the goal of transforms. Why is this?

The motivation behind many transformations stems from the following general observation: Almost all the signals we are interested in processing in the real world are generated by some kind of orderly process and hence nearby (in time or space) parts of the signal are often heavily *correlated*[3]; this means that the signal is in some way *redundant.* This observation, coupled with the hard fact that we always have a limit on the resources available to apply to our signal (bandwidth to communicate it or processing time to analyze it) suggests that what we really want to do is squeeze out that redundnacy and only spend our energy on the limited amount of real information present in the signal.

But there is a problem: if we have a highly correlated signal and we only want to process or send some of it, which parts do we throw away to get rid of redundancy? It is impossible to say, because in the original signal the information is distributed thinly across all the domain, so throwing some samples away always causes us to lose some important information. The whole idea of transforms is that they move us to a space in which the coefficients are *highly uncorrelated*

[3]If nearby parts of a signal are correlated, it means that knowing the samples close to a particular sample help us a lot to guess what the value of that sample will be. Random strings are uncorrelated: Guess the missing digit in 110101001?0101. More regular strings like engl?sh text are highly correlated.

and so we can simply take the "biggest" (in some sense) ones and throw away all the others and be confident that we have kept the bulk of the information and thrown out mostly noise. Image compression schemes like MPEG, for example, typically take the discrete cosine transform (DCT) of the signal, transmit or store only the largest coefficients and then reconstruct from those.

B.1.2 Linear transforms

All of the so-called frequency-domain transforms such as the DFT and the DCT are *linear*. This fact is not merely convenient or useful for tractability; it is fundamental to the action of these transforms and is inextricably linked to the information they provide and the kinds of systems they allow us to analyze.

Recall that a transformation $\mathcal{H}$ is linear if and only if $\mathcal{H}(ax + by) = a\mathcal{H}(x) + b\mathcal{H}(y)$ where x and y are signals and a and b are (possibly complex) constants. In other words, linear transforms have two essential properties: (1) if we scale the input, we scale the output by the same amount and (2) if we add inputs the output is just the sum of the outputs to the individual inputs.

Property (1) means that we really don't have to worry about the numbers on the vertical axes of any of our plots. We could multiply or divide any of our signals by 10^{100} and nothing would change except that all the axes everywhere would get multiplied by the same amount. So we might as well always work with input signals that are normalized in some way, for example so their maximum is unity or their energy is unity.

Property (2) lays the groundwork for an intriguing possibility for being lazy. What if we could discover a "library" of simple signals such that: (a) the action of the transform on each of the library signals was easy to determine, and (b) linear combinations of these library functions can generate almost all of the (not so simple) signals that we will ultimately be interested in transforming. In such a case *all we need to understand is how the transform affects each of the signals in the library* and then we have understood how it affects all possible signals that can be built from the library. Transforming any signal, no matter how nasty, could then be achieved by the following lazy method: Represent the nasty signal as a linear combination of the library signals. Find the transform of all the library signals. Combine these transform outputs to get the transform of the original signal. This approach begs two important questions: First, how can we ever hope to find such a magical library? Second, even if we do find one, how do we know that the first step in the lazy method, "Represent the signal as a linear combination of library signals," is not really hard? Fortunately, there exists both such a library and a method for representing any signal using

it.

The complex exponential basis functions $e^{-j\omega t}$ are exactly such a library. They are useful because they are the *eigenfunctions* of linear systems, meaning that if the input to a linear system is a complex exponential then the output is a constant multiple of the same complex exponential. This is half of what we need—the remaining problem is how to represent signals in this basis. The Fourier transform and Fourier series are the machines which do this for us, and that is why they are so essential to all of signal processing.

B.2 A review of linear Gaussian models

Many common statistical techniques for modeling multidimensional static datasets and multidimensional time series can be seen as variants of one underlying model. As shown below, these include factor analysis (FA), principal component analysis (PCA), mixtures of Gaussian clusters, vector quantization (VQ), Kalman filter models (a.k.a. linear dynamical systems) and hidden Markov models (HMMs). The relationships between some of these models has been noted in passing in the recent literature. For example, Hinton et al. (1995) note that factor analysis and PCA are closely related and Digalakis et al. (1993) relate the forward–backward algorithm for HMMs to Kalman filtering. This chapter unifies many of the disparate observations made by previous authors (Rubin and Thayer, 1982; Delyon, 1993; Digalakis et al., 1993; Hinton et al., 1995; Elliott et al., 1995; Ghahramani and Hinton, 1996a, 1996b, 1997; Hinton and Ghahramani, 1997) and presents a review of all these algorithms as instances of a single basic generative model. This unified view allows one to show some interesting relations between previously disparate algorithms. The framework also makes it possible to derive a new model for static data which is based on PCA but has a sensible probabilistic interpretation (see chapter 4) as well as a novel concept of spatially adaptive observation noise. I also review some of the literature involving global and local mixtures of the basic models.

B.2.1 Gaussian noise and least squares

A d-dimensional multidimensional Gaussian (normal) density for $\mathbf{x}$ is:

$$\mathcal{N}(\mu, \Sigma) = (2\pi)^{-d/2}|\Sigma|^{-1/2} \exp\left[-\frac{1}{2}(\mathbf{x}-\mu)^T\Sigma^{-1}(\mathbf{x}-\mu)\right] \tag{B.1}$$

The popularity of linear Gaussian models comes from several fortunate analytical properties of Gaussian processes: the sum of two independent Gaussian distributed quantities is also Gaussian distributed[4] and the output of a linear system whose input is Gaussian distributed is again Gaussian distributed. In particular,

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \Rightarrow (\mathbf{A}\mathbf{x} + \mathbf{y}) \sim \mathcal{N}\left(\mathbf{A}\boldsymbol{\mu} + \mathbf{y}, \mathbf{A}\boldsymbol{\Sigma}\mathbf{A}^T\right) \tag{B.2}$$

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \Rightarrow \boldsymbol{\Sigma}^{-1/2}(\mathbf{x} - \boldsymbol{\mu}) \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \tag{B.3}$$

$$\mathbf{x} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma}) \Rightarrow (\mathbf{x} - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x} - \boldsymbol{\mu}) \sim \chi^2_n \tag{B.4}$$

Furthermore, problems involving the log likelihood of independent datapoints under a Gaussian distribution often reduces to least squares problems since the log likelihood is just the sum of matrix quadratic forms:

$$L = \log P(\{\mathbf{x}_1 \ldots \mathbf{x}_N\}) = \log \prod_{n=1}^{N} P(\mathbf{x}_n) \tag{B.5}$$

$$= \sum_{n=1}^{N} \log P(\mathbf{x}_n) = \sum_{n=1}^{N} \log \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})\,|_{\mathbf{x}_n} \tag{B.6}$$

$$= \sum_{n=1}^{N} \left[(\mathbf{x}_n - \boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}_n - \boldsymbol{\mu}) + \log|\boldsymbol{\Sigma}|\right] + \text{constant} \tag{B.7}$$

Problems involving optimizing cost functions that are quadratics (such as least or greatest squares) can be solved easily, exactly and quickly using techniques from linear algebra for solving systems of linear equations since all fixed points of such cost functions are specified by setting their first derivatives to zero which yield linear equations. For example, the well known result that the sample mean is the maximum likelihood parameter setting for a Gaussian model:

$$\frac{\partial L}{\partial \mu} = 2\boldsymbol{\Sigma}^{-1} \sum_{n=1}^{N} (\mathbf{x}_n - \boldsymbol{\mu}) \tag{B.8}$$

[4]In other words the convolution of two Gaussians is again a Gaussian. In particular, the convolution of $\mathcal{N}(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$ and $\mathcal{N}(\boldsymbol{\mu}_2, \boldsymbol{\Sigma}_2)$ is $\mathcal{N}(\boldsymbol{\mu}_1 + \boldsymbol{\mu}_2, \boldsymbol{\Sigma}_1 + \boldsymbol{\Sigma}_2)$. This is not the same as the (false) statement that the sum of two Gaussians is a Gaussian but is the same as the (Fourier domain equivalent) statement that the multiplication of two Gaussians is a Gaussian (although no longer normalized).

B.2.2 The basic model

The basic models we will work with are discrete time linear dynamical systems with Gaussian noise. In such models we assume that the state of the process in question can at any time be summarized by a k-vector of *state variables* or *causes* $\mathbf{x}$ which we cannot observe directly. However, the system also produces at each time step an *output* or *observable* p-vector $\mathbf{y}$ to which we do have access.

The state $\mathbf{x}$ is assumed to evolve according to simple first-order Markov dynamics; each output vector $\mathbf{y}$ is generated from the current state by a simple linear observation process. Both the state evolution and the observation processes are corrupted by additive Gaussian noise which is also hidden. If we work with a continuous valued state variable $\mathbf{x}$, the basic generative model can be written[5] as:

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{w}_t = \mathbf{A}\mathbf{x}_t + \mathbf{w}_\bullet \qquad \mathbf{w}_\bullet \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{B.9a}$$

$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v}_\bullet \qquad \mathbf{v}_\bullet \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \tag{B.9b}$$

where $\mathbf{A}$ is the $k \times k$ *state transition matrix* and $\mathbf{C}$ is the $p \times k$ *observation*, *measurement*, or *generative* matrix.

The k-vector $\mathbf{w}$ and p-vector $\mathbf{v}$ are random variables representing the state evolution and observation noises respectively which are independent of each other and of the values of $\mathbf{x}$ and $\mathbf{y}$. Both of these noise sources are temporally white (uncorrelated from time step to time step) and spatially Gaussian distributed[6] with zero mean and covariance matrices which I shall denote $\mathbf{Q}$ and $\mathbf{R}$ respectively. I have written $\mathbf{w}_\bullet$ and $\mathbf{v}_\bullet$ in place of $\mathbf{w}_t$ and $\mathbf{v}_t$ to emphasize that the noise processes do not have any knowledge of the time index. The restriction to zero mean noise sources is not a loss of generality.[7] Since the state evolution noise is Gaussian and its dynamics are linear, $\mathbf{x}_t$ is a *first-order Gauss-Markov random process*. Note that the noise processes are essential elements of the model: without the process noise $\mathbf{w}_\bullet$, the state $\mathbf{x}_t$ would always either shrink exponentially to zero or blow up exponentially in the direction of the leading eigenvector

[5] All vectors are column vectors. To denote the transpose of a vector or matrix, I use the notation $\mathbf{x}^T$. The determinant of a matrix is denoted by $|\mathbf{A}|$ and matrix inversion by $\mathbf{A}^{-1}$. The symbol $\sim$ means "distributed according to." A multivariate normal (Gaussian) distribution with mean $\boldsymbol{\mu}$ and covariance matrix $\boldsymbol{\Sigma}$ is written as $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. The same Gaussian evaluated at the point $\mathbf{z}$ is denoted $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})|_{\mathbf{z}}$.

[6] An assumption that is weakly motivated by the Central Limit Theorem but more strongly by analytic tractability.

[7] Specifically we could always add a $k+1^{st}$ dimension to the state vector which is fixed at unity. Then augmenting $\mathbf{A}$ with an extra column holding the noise mean and an extra row of zeros (except unity in the bottom right corner) takes care of a nonzero mean for $\mathbf{w}_\bullet$. Similarly, adding an extra column to $\mathbf{C}$ takes care of a nonzero mean for $\mathbf{v}_\bullet$.

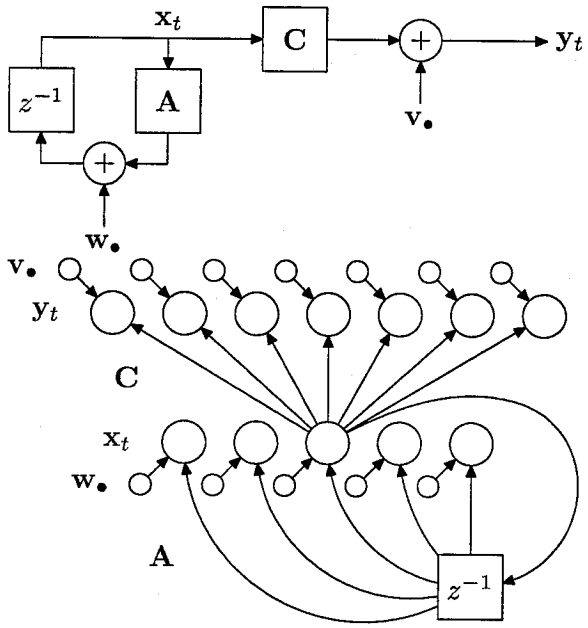


Figure B.1: Linear dynamical system generative model. The z^{-1} block is a unit delay. The covariance matrix of the input noise $\mathbf{w}$ is $\mathbf{Q}$ and the covariance matrix of the output noise $\mathbf{v}$ is $\mathbf{R}$. In the network model below, the smaller circles represent noise sources and all units are linear. Outgoing weights have only been drawn from one hidden unit. This model is equivalent to a Kalman filter model (linear dynamical system).

of $\mathbf{A}$; similarly in the absence of the observation noise $\mathbf{v}_\bullet$ the state would no longer be hidden. Figure B.1 illustrates this basic model using both the engineering system block form and the network form more common in machine learning.

Notice that there is degeneracy in the model: all of the structure in the matrix $\mathbf{Q}$ can be moved into the matrices $\mathbf{A}$ and $\mathbf{C}$. This means that we can without loss of generality work with models in which $\mathbf{Q}$ is the identity matrix.[8] Of course $\mathbf{R}$ cannot be restricted in the same way since the values $\mathbf{y}_t$ are observed and hence we are not free to whiten or otherwise rescale them. Finally, the components of the state vector can be arbitrarily reordered; this corresponds to swapping the columns of $\mathbf{C}$ and also of $\mathbf{A}$. Typically we choose an ordering based on the norms of the columns of $\mathbf{C}$ which resolves this degeneracy.

The network diagram of figure B.1 can be unfolded in time to give separate units for each time step. Such diagrams are the standard method of illustrating *graphical models*, also known as probabilistic independence networks, a category of models which includes Markov networks, Bayesian (or belief) networks, and other formalisms [145, 112, 203, 179]. A graphical model is a representation of the dependency structure between variables in a multivariate probability distribution. Each node corresponds to a random variable, and the absence of an arc between two variables corresponds to a particular conditional independence relation. While graphical models are beyond the scope of this review, it is important to point out that they provide a very general framework for working with the models considered below. In this review, I unify and extend some well known statistical models and signal processing algorithms by focusing on variations of linear graphical models with Gaussian noise.

The main idea of the models in equations (B.9) is that the hidden state sequence $\mathbf{x}_t$ should be an *informative lower dimensional projection* or *explanation* of the complicated observation sequence $\mathbf{y}_t$. With the aid of the dynamical and noise models, the states should summarize the underlying causes of the data much more succinctly than the observations themselves. For this reason we often work with state dimensions much smaller than the number of observables — in other words $k \ll p$.[9] We assume that both $\mathbf{A}$ and $\mathbf{C}$ are of rank k and that $\mathbf{Q}$,$\mathbf{R}$ and $\mathbf{Q}_1$ (introduced

[8] In particular, since is it a covariance matrix, $\mathbf{Q}$ is symmetric positive semi-definite and thus can be diagonalized to the form $\mathbf{E}\mathbf{\Lambda}\mathbf{E}^T$ (where $\mathbf{E}$ is a rotation matrix of eigenvectors and $\mathbf{\Lambda}$ is a diagonal matrix of eigenvalues). Thus for any model in which $\mathbf{Q}$ is not the identity matrix, we can generate an *exactly equivalent* model using a new state vector $\mathbf{x}' = \mathbf{\Lambda}^{-1/2}\mathbf{E}^T\mathbf{x}$ with $\mathbf{A}' = (\mathbf{\Lambda}^{-1/2}\mathbf{E}^T)\mathbf{A}(\mathbf{E}\mathbf{\Lambda}^{1/2})$ and $\mathbf{C}' = \mathbf{C}(\mathbf{E}\mathbf{\Lambda}^{1/2})$ such that the new covariance of $\mathbf{x}'$ is the identity matrix: $\mathbf{Q}' = \mathbf{I}$.

[9] More precisely, in a model where all the matrices are full-rank, the problem of inferring the state from a sequence of τ consecutive observations is well-defined as long $k \leq \tau p$ (a notion related to *observability* in systems theory [77]). For this reason, in dynamic models it is sometimes useful to use state spaces of larger dimension than the observations,

below) are always full rank.

B.2.3 Probability computations

If we assume the initial state $\mathbf{x}_1$ of the system to be Gaussian distributed:

$$\mathbf{x}_1 \sim \mathcal{N}(\boldsymbol{\mu}_1, \mathbf{Q}_1) \tag{B.10}$$

then all future states $\mathbf{x}_t$ and observations $\mathbf{y}_t$ will also be Gaussian distributed. In fact, we can write explicit formulae for the conditional expectations of the states and observables:

$$P(\mathbf{x}_{t+1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{A}\mathbf{x}_t, \mathbf{Q})\,|_{\mathbf{x}_{t+1}} \tag{B.11a}$$

$$P(\mathbf{y}_t|\mathbf{x}_t) = \mathcal{N}(\mathbf{C}\mathbf{x}_t, \mathbf{R})\,|_{\mathbf{y}_t} \tag{B.11b}$$

Furthermore, because of the Markov properties of the model and the Gaussian assumptions about the noise and initial distributions, it is easy to write an expression for the joint probability of a sequence of τ states and outputs:

$$P(\{\mathbf{x}_1 \ldots \mathbf{x}_\tau\}, \{\mathbf{y}_1 \ldots \mathbf{y}_\tau\}) = P(\mathbf{x}_1) \prod_{t=1}^{\tau-1} P(\mathbf{x}_{t+1}|\mathbf{x}_t) \prod_{t=1}^{\tau} P(\mathbf{y}_t|\mathbf{x}_t) \tag{B.12}$$

The negative log probability (cost) is just the sum of matrix quadratic forms:

$$\begin{aligned} -2\log P(\{\mathbf{x}_1 \ldots \mathbf{x}_\tau\}, \{\mathbf{y}_1 \ldots \mathbf{y}_\tau\}) = &\sum_{t=1}^{\tau} \left[(\mathbf{y}_t - \mathbf{C}\mathbf{x}_t)^T \mathbf{R}^{-1} (\mathbf{y}_t - \mathbf{C}\mathbf{x}_t) + \log|\mathbf{R}|\right] + \\ &\sum_{t=1}^{\tau-1} \left[(\mathbf{x}_{t+1} - \mathbf{A}\mathbf{x}_t)^T \mathbf{Q}^{-1} (\mathbf{x}_{t+1} - \mathbf{A}\mathbf{x}_t) + \log|\mathbf{Q}|\right] + \\ &(\mathbf{x}_1 - \boldsymbol{\mu}_1)^T \mathbf{Q}_1^{-1} (\mathbf{x}_1 - \boldsymbol{\mu}_1) + \log|\mathbf{Q}_1| + \tau(p+k)\log 2\pi \end{aligned} \tag{B.13}$$

B.2.4 Learning and estimation problems

Latent variable models have a wide spectrum of application in data analysis. In some scenarios we know exactly what the hidden states are supposed to be and just want to estimate them. For example in a vision problem the hidden states may be the location or pose of an object; in a

$k > p$, in which case a single state vector provides a representation of a *sequence* of observations.

tracking problem the states may be positions and momenta. In these cases we can often write down *a priori* the observation and/or state evolution matrices based on our knowledge of the problem structure or physics. The emphasis is on accurate inference of the unobserved information from the data we do have, for example, from an image of an object or radar observations. In other scenarios we are trying to discover explanations or causes for our data and have no explicit model for what these causes should be. The observation and state evolution processes are mostly or entirely unknown. The emphasis is instead on robustly learning a few parameters which model the observed data well (assign it high likelihood). Speech modeling is a good example of such a situation — our goal is to find economical models that perform well for recognition tasks but the particular values of hidden states in our models may not be meaningful or important to us. These two goals, estimating the hidden states given observations and a model, and learning the model parameters, typically manifest themselves in the solution of two distinct problems: *inference* and *system identification.*

Inference: filtering and smoothing

The first problem is that of *inference* or *filtering/smoothing* which asks: *given* fixed model parameters $\{\mathbf{A}, \mathbf{C}, \mathbf{Q}, \mathbf{R}, \mu_1, \mathbf{Q}_1\}$ what can be said about the unknown hidden state sequence given some observations? This question is typically made precise in several ways. A very basic quantity we would like to be able to compute is the *total likelihood* of an observation sequence:

$$P(\{\mathbf{y}_1 \ldots \mathbf{y}_\tau\}) \neq \int_{\text{all possible } \{\mathbf{x}_1 \ldots \mathbf{x}_\tau\}} P(\{\mathbf{x}_1 \ldots \mathbf{x}_\tau\}, \{\mathbf{y}_1 \ldots \mathbf{y}_\tau\})\ d\{\mathbf{x}_1 \ldots \mathbf{x}_\tau\} \tag{B.14}$$

This marginalization requires an efficient way of integrating (or summing) the joint probability (easily computed by (B.13) or similar formulae) over all possible paths through state space.

Once this integral is available it is simple to compute the *conditional distribution* for any *one* proposed hidden state sequence given the observations by dividing the joint probability by the total likelihood of the observations:

$$P(\{\mathbf{x}_1 \ldots \mathbf{x}_\tau\} | \{\mathbf{y}_1 \ldots \mathbf{y}_\tau\}) = \frac{P(\{\mathbf{x}_1 \ldots \mathbf{x}_\tau\}, \{\mathbf{y}_1 \ldots \mathbf{y}_\tau\})}{P(\{\mathbf{y}_1 \ldots \mathbf{y}_\tau\})} \tag{B.15}$$

Often we are interested in the distribution of the hidden state at a particular time t. In *filtering* we

attempt to compute this conditional posterior probability:

$$P(\mathbf{x}_t|\{\mathbf{y}_1 \ldots \mathbf{y}_t\}) \tag{B.16}$$

given all the observations up to and including time t. In *smoothing* we compute the distribution over $\mathbf{x}_t$:

$$P(\mathbf{x}_t|\{\mathbf{y}_1 \ldots \mathbf{y}_\tau\}) \tag{B.17}$$

given the entire sequence of observations. (It is also possible to ask for the conditional state expectation given observations that extend only a few time steps into the future – *partial smoothing* or that end a few time steps before the current time – *partial prediction.*) These conditional calculations are closely related to the computation of (B.14) and often the intermediate values of a recursive method used to compute (B.14) give the desired distributions (B.16) or (B.17). Filtering and smoothing have been extensively studied for continuous state models in the signal processing community, starting with the seminal works of Kalman [103, 104] and Rauch [155, 156] although this literature is often not well known in the machine learning community. For the discrete state models much of the literature stems from the work of Baum and colleagues [12, 11, 13, 10] on hidden Markov models and of Viterbi [199] and others on optimal decoding. The recent book by Elliott and colleagues (1995) contains a thorough mathematical treatment of filtering and smoothing for many general models.

Learning (system identification)

The second problem of interest with linear Gaussian models is the *learning* or *system identification* problem: given only an observed sequence (or perhaps several sequences) of outputs $\{\mathbf{y}_1 \ldots \mathbf{y}_\tau\}$ find the parameters $\{\mathbf{A}, \mathbf{C}, \mathbf{Q}, \mathbf{R}, \mu_1, \mathbf{Q}_1\}$ which maximize the likelihood of the observed data as computed by (B.14).

The learning problem has been investigated extensively by neural network researchers for static models and also for some discrete state dynamic models such as hidden Markov models or the more general Bayesian belief networks. There is a corresponding area of study in control theory known as *system identification* which investigates learning in continuous state models. For linear Gaussian models there are several approaches to system identification [122], but to

clarify the relationship between these models and the others reviewed in this paper, the focus is on system identification methods based on the *EM*algorithm, described below. The *EM*algorithm for linear Gaussian dynamical systems was originally derived by Shumway and Stoffer (1982) and recently reintroduced (and extended) in the neural computation field by Ghahramani and Hinton (1996a, 1996b). Digalakis *et al.* (1993) made a similar reintroduction and extension in the speech processing community. Once again I mention the book by Elliott et al. (1995) which also covers learning in this context.

The basis of all the learning algorithms presented by these authors is the powerful *EM*or *expectation-maximization* algorithm [12, 43]. The objective of the *EM*algorithm is to maximize the likelihood of the observed data (equation (B.14)) in the presence of hidden variables. Let us denote the observed data by $\mathbf{Y} = \{\mathbf{y}_1 \ldots \mathbf{y}_\tau\}$, the hidden variables by $\mathbf{X} = \{\mathbf{x}_1 \ldots \mathbf{x}_\tau\}$, and the parameters of the model by θ. Maximizing the likelihood as a function of θ is equivalent to maximizing the log likelihood:

$$\mathcal{L}(\theta) = \log P(\mathbf{Y}|\theta) = \log \int_{\mathbf{X}} P(\mathbf{X}, \mathbf{Y}|\theta)\, d\mathbf{X} \tag{B.18}$$

Using *any* distribution Q over the hidden variables, we can obtain a lower bound on $\mathcal{L}$:

$$\begin{aligned}
\log \int_{\mathbf{X}} P(\mathbf{Y}, \mathbf{X}|\theta)\, d\mathbf{X} &= \log \int_{\mathbf{X}} Q(\mathbf{X}) \frac{P(\mathbf{X}, \mathbf{Y}|\theta)}{Q(\mathbf{X})}\, d\mathbf{X} && \text{(B.19a)} \\
&\geq \int_{\mathbf{X}} Q(\mathbf{X}) \log \frac{P(\mathbf{X}, \mathbf{Y}|\theta)}{Q(\mathbf{X})}\, d\mathbf{X} && \text{(B.19b)} \\
&= \int_{\mathbf{X}} Q(\mathbf{X}) \log P(\mathbf{X}, \mathbf{Y}|\theta)\, d\mathbf{X} - \int_{\mathbf{X}} Q(\mathbf{X}) \log Q(\mathbf{X})\, d\mathbf{X} && \text{(B.19c)} \\
&= \mathcal{F}(Q, \theta) && \text{(B.19d)}
\end{aligned}$$

where the middle inequality is known as Jensen's inequality and can be proven using the concavity of the log function. If we define the *energy* of a global configuration $(\mathbf{X}, \mathbf{Y})$ to be $-\log P(\mathbf{X}, \mathbf{Y}|\theta)$, then some readers may notice that the lower bound $\mathcal{F}(Q, \theta) \leq \mathcal{L}(\theta)$ is the negative of a quantity known in statistical physics as the *free energy*: the expected energy under Q minus the entropy of Q [136]. The *EM*algorithm alternates between maximizing $\mathcal{F}$ with respect to the *distribution* Q and the *parameters* θ, respectively, holding the other fixed. Starting from some initial parameters

θ_0:

$$\textbf{E step:} \qquad Q_{k+1} \leftarrow \arg\max_{Q} \; \mathcal{F}(Q, \theta_k) \tag{B.20a}$$

$$\textbf{M step:} \qquad \theta_{k+1} \leftarrow \arg\max_{\theta} \; \mathcal{F}(Q_{k+1}, \theta) \tag{B.20b}$$

It is easy to show that the maximum in the E step results when Q is exactly the conditional distribution of $\mathbf{X}$: $Q_{k+1}(\mathbf{X}) = P(\mathbf{X}|\mathbf{Y}, \theta_k)$, at which point the bound becomes an equality: $\mathcal{F}(Q_{k+1}, \theta_k) = \mathcal{L}(\theta_k)$. The maximum in the M step is obtained by maximizing the first term in (B.19c), since the entropy of Q does not depend on θ:

$$\textbf{M step:} \qquad \theta_{k+1} \leftarrow \arg\max_{\theta} \int_{\mathbf{X}} P(\mathbf{X}|\mathbf{Y}, \theta_k) \log P(\mathbf{X}, \mathbf{Y}|\theta) \, d\mathbf{X}. \tag{B.21}$$

This is the expression most often associated with the *EM*algorithm, but it obscures the elegant interpretation of *EM*as coordinate ascent in $\mathcal{F}$ [136]. Since $\mathcal{F} = \mathcal{L}$ at the beginning of each M step, and since the E step does not change θ, we are guaranteed not to decrease the likelihood after each combined *EM*step.

Therefore, at the heart of the *EM*learning procedure is the following idea: use the solutions to the filtering/smoothing problem to estimate the unknown hidden states given the observations and the current model parameters. Then use this fictitious complete data to solve for new model parameters. Given the estimated states obtained from the inference algorithm, it is usually easy to solve for new parameters. For linear Gaussian models this typically involves minimizing quadratic forms such as (B.13) which can be done with linear regression. This process is repeated, using these new model parameters to infer the hidden states again, and so on. I now touch upon one general point which often causes confusion. Our goal is to maximize the *total* likelihood (B.14) (or equivalently maximize the total log likelihood) of the observed data with respect to the model parameters. This means integrating (or summing) over all ways in which the generative model could have produced the data. As a consequence of using the *EM*algorithm to do this maximization we find ourselves needing to compute (and maximize) the *expected* log likelihood of the joint data, where the expectation is taken over the distribution of hidden values predicted by the current model parameters and the observations. Thus, it appears that we are maximizing the incorrect quantity,

but doing so is in fact guaranteed to increase (or keep the same) the quantity of interest at each iteration of the algorithm.

B.2.5 Continuous state linear Gaussian systems

Having described the basic model and learning procedure, I now focus on specific linear instances of the model in which the hidden state variable $\mathbf{x}$ is continuous and the noise processes are Gaussian. This will allow me to elucidate the relationship between factor analysis, PCA, and Kalman filter models. The discussion is divided into models which generate *static data* and those which generate *dynamic data.* Static data has no temporal dependence – no information would be lost by permuting the ordering of the data points $\mathbf{y}_t$; whereas for dynamic data the time ordering of the data points is crucial.

Static Data Modeling: Factor Analysis, SPCA and PCA

In many situations we have reason to believe (or at least to assume) that each point in our dataset was generated independently and identically. In other words, there is no natural (temporal) ordering to the data points; they merely form a collection. In such cases we assume that the underlying state vector $\mathbf{x}$ has no dynamics, *i.e.* the matrix $\mathbf{A}$ is the zero matrix, and therefore $\mathbf{x}$ is simply a constant (which we can take without loss of generality to be the zero vector) corrupted by noise. The new generative model then becomes:

$$\mathbf{A} = \mathbf{0} \quad \Rightarrow \quad \mathbf{x}_\bullet = \mathbf{w}_\bullet \qquad \mathbf{w}_\bullet \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{B.22a}$$

$$\mathbf{y}_\bullet = \mathbf{C}\mathbf{x}_\bullet + \mathbf{v}_\bullet \qquad \mathbf{v}_\bullet \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \tag{B.22b}$$

Notice that since $\mathbf{x}_t$ is driven only by the noise $\mathbf{w}_\bullet$ and since $\mathbf{y}_t$ depends only on $\mathbf{x}_t$, all temporal dependence has disappeared. This is the motivation for the term *static* and for the notations $\mathbf{x}_\bullet$ and $\mathbf{y}_\bullet$ above. We also no longer use a separate distribution for the initial state: $\mathbf{x}_1 \sim \mathbf{x}_\bullet \sim \mathbf{w}_\bullet \sim \mathcal{N}(\mathbf{0}, \mathbf{Q})$.

This model is illustrated in figure B.2. We can analytically integrate (B.14) to obtain the marginal distribution of $\mathbf{y}_\bullet$, which is the Gaussian

$$\mathbf{y}_\bullet \sim \mathcal{N}\left(\mathbf{0}, \mathbf{C}\mathbf{Q}\mathbf{C}^T + \mathbf{R}\right) \tag{B.23}$$

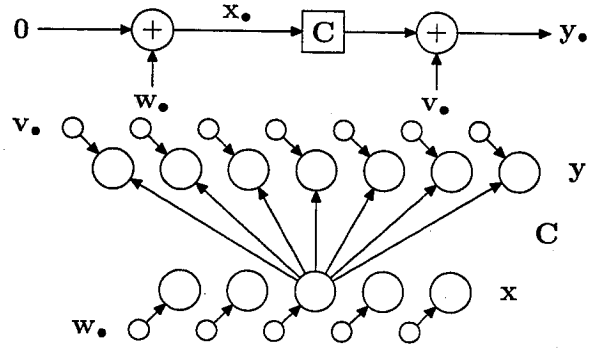


Figure B.2: Static generative model (continuous state). The covariance matrix of the input noise $\mathbf{w}$ is $\mathbf{Q}$ and the covariance matrix of the output noise $\mathbf{v}$ is $\mathbf{R}$. In the network model below, the smaller circles represent noise sources and all units are linear. Outgoing weights have only been drawn from one hidden unit. This model is equivalent to factor analysis, SPCA and PCA models depending on the output noise covariance. For factor analysis, $\mathbf{Q} = \mathbf{I}$ and $\mathbf{R}$ is diagonal. For SPCA, $\mathbf{Q} = \mathbf{I}$ and $\mathbf{R} = \alpha\mathbf{I}$. For PCA, $\mathbf{Q} = \mathbf{I}$ and $\mathbf{R} = \lim_{\epsilon \to 0} \epsilon\mathbf{I}$.

Two things are important to notice. First, the degeneracy mentioned above persists between the structure in $\mathbf{Q}$ and $\mathbf{C}$.[10] This means there is no loss of generality in restricting $\mathbf{Q}$ to be diagonal. Furthermore, there is arbitrary sharing of scale between a diagonal $\mathbf{Q}$ and $\mathbf{C}$: typically we either restrict the columns of $\mathbf{C}$ to be unit vectors or make $\mathbf{Q}$ the identity matrix to resolve this degeneracy. In what follows I will assume $\mathbf{Q} = \mathbf{I}$ without loss of generality.

Second, the covariance matrix $\mathbf{R}$ of the observation noise *must* be restricted in some way for the model to capture any interesting or informative projections in the state $\mathbf{x}_\bullet$. If $\mathbf{R}$ were not restricted, learning could simply choose $\mathbf{C} = \mathbf{0}$ and then set $\mathbf{R}$ to be the sample covariance of the data thus trivially achieving the maximum likelihood model by explaining all of the structure in the data as noise. (Remember that since the model has reduced to a single Gaussian distribution for $\mathbf{y}_\bullet$, we can do no better than having the covariance of our model equal the sample covariance of our data.) Note that restricting $\mathbf{R}$, unlike making $\mathbf{Q}$ diagonal, *does* constitute some loss of generality from the original model of equations (B.22).

There is an intuitive spatial way to think about this static generative model: we use white noise to generate a spherical ball (since $\mathbf{Q} = \mathbf{I}$) of density in k-dimensional state space. This ball is then stretched and rotated into p-dimensional observation space by the matrix $\mathbf{C}$ where it looks like a k-dimensional pancake. The pancake is then convolved with the covariance density

[10] If we diagonalize $\mathbf{Q}$ and rewrite the covariance of $\mathbf{y}_\bullet$ the degeneracy becomes clear: $\mathbf{y}_\bullet \sim \mathcal{N}\left(\mathbf{0}, (\mathbf{C}\mathbf{E}\mathbf{\Lambda}^{1/2})(\mathbf{C}\mathbf{E}\mathbf{\Lambda}^{1/2})^T + \mathbf{R}\right)$. To make $\mathbf{Q}$ diagonal we simply replace $\mathbf{C}$ with $\mathbf{CE}$.

of $\mathbf{v}_\bullet$ (described by $\mathbf{R}$) to get the final covariance model for $\mathbf{y}_\bullet$. We want the resulting ellipsoidal density to be as close as possible to the ellipsoid given by the sample covariance of our data. If we restrict the shape of the $\mathbf{v}_\bullet$ covariance by constraining $\mathbf{R}$, we can force interesting information to appear in both $\mathbf{R}$ and $\mathbf{C}$ as a result.

Finally, observe that all varieties of filtering and smoothing reduce to the same problem in this static model because there is no time dependence. We are only seeking the posterior probability $P(\mathbf{x}_\bullet|\mathbf{y}_\bullet)$ over a single hidden state given the corresponding single observation. This inference is easily done by linear matrix projection and the resulting density is itself Gaussian:

$$P(\mathbf{x}_\bullet|\mathbf{y}_\bullet) = \frac{P(\mathbf{y}_\bullet|\mathbf{x}_\bullet)\,P(\mathbf{x}_\bullet)}{P(\mathbf{y}_\bullet)} = \frac{\mathcal{N}(\mathbf{C}\mathbf{x}_\bullet, \mathbf{R})\,|_{\mathbf{y}_\bullet}\;\mathcal{N}(\mathbf{0}, \mathbf{I})\,|_{\mathbf{x}_\bullet}}{\mathcal{N}(\mathbf{0}, \mathbf{C}\mathbf{C}^T + \mathbf{R})\,|_{\mathbf{y}_\bullet}} \tag{B.24a}$$

$$P(\mathbf{x}_\bullet|\mathbf{y}_\bullet) = \mathcal{N}(\beta\mathbf{y}_\bullet, I - \beta\mathbf{C})\,|_{\mathbf{x}_\bullet}\,, \qquad \beta = \mathbf{C}^T(\mathbf{C}\mathbf{C}^T + \mathbf{R})^{-1} \tag{B.24b}$$

from which we obtain not only the expected value $\beta\mathbf{y}_\bullet$ of the unknown state but also an estimate of the uncertainty in this value in the form of the covariance $\mathbf{I} - \beta\mathbf{C}$. Computing the likelihood of a data point $\mathbf{y}_\bullet$ is merely an evaluation under the Gaussian in equation (B.23). The learning problem now consists of identifying the matrices $\mathbf{C}$ and $\mathbf{R}$. There are a family of *EM*algorithms to do this for the various cases discussed below which are given in detail at the end of this note.

Factor Analysis

If we restrict the covariance matrix $\mathbf{R}$ which controls the observation noise to be *diagonal* (in other words the covariance ellipsoid of $\mathbf{v}_\bullet$ is axis aligned) and we set the state noise $\mathbf{Q}$ to be the identity matrix, then we recover exactly a standard statistical model known as maximum likelihood *factor analysis*. The unknown states $\mathbf{x}$ are called the *factors* in this context; the matrix $\mathbf{C}$ is called the *factor loading matrix* and the diagonal elements of $\mathbf{R}$ are often known as the *uniquenesses*. (See Everitt (1984) for a brief and clear introduction.) The inference calculation is done exactly as in (B.24b) above. The learning algorithm for the loading matrix and the uniquenesses is exactly an *EM*algorithm except that we must take care to constrain $\mathbf{R}$ properly (which is fortunately as easy as taking the diagonal of the unconstrained maximum likelihood estimate – see [164, 73]). If $\mathbf{C}$ is completely free, this procedure is called *exploratory factor analysis*; if we build *a priori* zeros into $\mathbf{C}$, it is *confirmatory factor analysis*. Note that in exploratory factor analysis we are

trying to model the covariance structure of our data with $p + pk - k(k-1)/2$ free parameters[11] instead of the $p(p+1)/2$ free parameters in a full covariance matrix.

The diagonality of $\mathbf{R}$ is the key assumption here. Factor analysis attempts to explain the covariance structure in the observed data by putting all the variance unique to each coordinate in the matrix $\mathbf{R}$ and putting all the correlation structure into $\mathbf{C}$ (this observation was first made by Lyttkens (1966) in response to work by Wold). In essence, factor analysis considers the axis rotation in which the original data arrived to be special because observation noise (often called *sensor noise*) is independent along the coordinates in these axes. However, the original scaling of the coordinates is unimportant: if we were to change the units in which we measured some of the components of $\mathbf{y}$, factor analysis could merely rescale the corresponding entry in $\mathbf{R}$ and row in $\mathbf{C}$ and achieve a new model which assigns the rescaled data identical likelihood. On the other hand if we rotate the axes in which we measure the data, we could not easily fix things since the noise $\mathbf{v}$ is constrained to have axis aligned covariance ($\mathbf{R}$ is diagonal).[12]

PCA and SPCA

If instead of restricting $\mathbf{R}$ to be merely diagonal, we require it to be a multiple of the identity matrix (in other words the covariance ellipsoid of $\mathbf{v}_\bullet$ is spherical), then we have a model which we will call *sensible principal component analysis* or SPCA [162] (see chapter 4). The columns of $\mathbf{C}$ span the *principal subspace* (the same subspace found by PCA—see below), and we will call the scalar value on the diagonal of $\mathbf{R}$ the *global noise level*. Note that SPCA uses $1 + pk - k(k-1)/2$ free parameters to model the covariance. Once again, inference is done with equation (B.24b) and learning by the *EM*algorithm (except that we now take the trace of the maximum likelihood estimate for $\mathbf{R}$ to learn the noise level – see [162]). Unlike factor analysis, SPCA considers the original axis rotation in which the data arrived to be unimportant: if the measurement coordinate system were rotated, SPCA could (left) multiply $\mathbf{C}$ by the same rotation and the likelihood of the new data would not change. On the other hand, the original scaling of the coordinates *is* privileged

[11]The correction $k(k-1)/2$ comes in because of degeneracy in unitary transformations of the factors – see for example [54].

[12]Zoubin Ghahramani pointed out that *EM*for factor analysis has been criticised as being quite slow [164]. Indeed, the standard method for fitting a factor analysis model [99] is based on a quasi-Newton optimization algorithm [62] which has been found empirically to converge faster than *EM*. I present the *EM*algorithm here, not because it is the most efficient way of fitting a factor analysis model, but to emphasize that for factor analysis and all the other latent variable models reviewed in this paper, *EM*provides a unified approach to learning. Finally, recent work in on-line learning has shown that it is possible to derive a family of *EM*-like algorithms with faster convergence rates than the standard *EM*algorithm [107, 9].

because SPCA assumes that the observation noise has the same variance in all directions in the measurement units used for the observed data. If we were to rescale one of the components of $\mathbf{y}$, the model could not be easily corrected since $\mathbf{v}$ has spherical covariance ($\mathbf{R} = \epsilon\mathbf{I}$). The SPCA model is very similar to the independently proposed *probabilistic principal component analysis* [191].

If we go even further and take the limit $\mathbf{R} = \lim_{\epsilon \to 0} \epsilon\mathbf{I}$ (while keeping the diagonal elements of $\mathbf{Q}$ finite[13]), then we obtain the standard *principal component analysis* or PCA model. The directions of the columns of $\mathbf{C}$ are known as the *principal components*. Inference now reduces to simple least squares projection[14]:

$$P(\mathbf{x}_\bullet|\mathbf{y}_\bullet) = \mathcal{N}(\beta\mathbf{y}_\bullet, I - \beta\mathbf{C})|_{\mathbf{x}_\bullet}, \qquad \beta = \lim_{\epsilon \to 0} \mathbf{C}^T(\mathbf{C}\mathbf{C}^T + \epsilon\mathbf{I})^{-1} \tag{B.25a}$$

$$P(\mathbf{x}_\bullet|\mathbf{y}_\bullet) = \mathcal{N}\left((\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{y}_\bullet, \mathbf{0}\right)|_{\mathbf{x}_\bullet} = \delta(\mathbf{x}_\bullet - (\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T\mathbf{y}_\bullet) \tag{B.25b}$$

Since the noise has become infinitesimal, the posterior over states collapses to a single point and the covariance becomes zero. There is still an *EM*algorithm for learning [162], although it can of course only learn $\mathbf{C}$. For PCA, we could just diagonalize the sample covariance of the data and take the leading k eigenvectors multiplied by their eigenvalues to be the columns of $\mathbf{C}$. This approach would give us $\mathbf{C}$ in one step but has many problems.[15] The *EM*learning algorithm amounts to an iterative procedure for finding these leading eigenvectors without explicit diagonalization.

An important final comment is that (regular) PCA does not define a proper density model in the observation space. So we cannot ask directly about the likelihood assigned by the model to some

[13]Since isotropic scaling of the data space is arbitrary, we could just as easily take the limit as the diagonal elements of $\mathbf{Q}$ became infinite while holding $\mathbf{R}$ finite or take both limits at once. The idea is that the noise variance becomes infinitesimal compared to the scale of the data.

[14]Recall that if $\mathbf{C}$ is $p \times k$ with $p > k$ and is rank k, then left multiplication by $\mathbf{C}^T(\mathbf{C}\mathbf{C}^T)^{-1}$ (which appears not to be well defined because $(\mathbf{C}\mathbf{C}^T)$ is not invertible) is *exactly equivalent to* left multiplication by $(\mathbf{C}^T\mathbf{C})^{-1}\mathbf{C}^T$. This is the same as the singular value decomposition idea of defining the "inverse" of the diagonal singular value matrix as the inverse of an element unless it is zero in which case it remains zero. The intuition is that even though $\mathbf{C}\mathbf{C}^T$ truly is not invertible, the directions along which it is not invertible are exactly those which $\mathbf{C}^T$ is about to project out.

[15]It is computationally very hard to diagonalize or invert large matrices. It also requires an enormous amount of data to make a large sample covariance matrix full rank. So if we are working with patterns in a large (thousands) number of dimensions and want to extract only a few (tens) principal components, we cannot naively try to diagonalize the sample covariance of our data. Techniques like the *snap-shot method* [176] attempt to address this but still require the diagonalization of an N by N matrix where N is the number of data points. The *EM*algorithm approach solves all of these problems, requiring no explicit diagonalization whatsoever and the inversion of only a k by k matrix. It is guaranteed to converge to the true *principal subspace* (i.e the same subspace spanned by the principal components). Empirical experiments [162] indicate that it converges in a few iterations, unless the ratio of the leading eigenvalues is near unity.

data. We can, however, examine a quantity which is proportional to the negative log likelihood in the limit of zero noise. This is the sum squared deviation of each data point from its projection. It is this "cost" which the learning algorithm ends up minimizing and which is the only available evaluation of how well a PCA model fits new data. This is one of the most critical failings of PCA: translating points by arbitrary amounts inside the principal subspace has no effect on the model error.

Time Series Modeling: Kalman Filter Models

We use the term *dynamic data* to refer to observation sequences in which the temporal ordering is important. For such data we do not want to ignore the state evolution dynamics, as it provides the only aspect of the model capable of capturing temporal structure. Systems described by the original dynamic generative model:

$$\mathbf{x}_{t+1} = \mathbf{A}\mathbf{x}_t + \mathbf{w}_t = \mathbf{A}\mathbf{x}_t + \mathbf{w}_\bullet \qquad \mathbf{w}_\bullet \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{B.9a}$$

$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v}_\bullet \qquad \mathbf{v}_\bullet \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \tag{B.9b}$$

are known as *linear dynamical systems* or *Kalman filter models* and have been extensively investigated by the engineering and control communities for decades. The emphasis has traditionally been on inference problems: the famous Discrete Kalman Filter [103, 104] gives an efficient recursive solution to the optimal filtering and likelihood computation problems, while the RTS recursions [155, 156] solve the optimal smoothing problem. Learning of unknown model parameters was studied by Shumway and Stoffer (1982) ($\mathbf{C}$ known) and by Ghahramani and Hinton (1996a) and Digalakis *et al.* (1993) (all parameters unknown). Figure B.1 illustrates this model, and the appendix gives pseudocode for its implementation.

We can extend our spatial intuition of the static case to this dynamic model. Just as before, any point in state space is surrounded by a ball (or ovoid) of density (described by $\mathbf{Q}$) which is stretched (by $\mathbf{C}$) into a pancake in observation space and then convolved with the observation noise covariance (described by $\mathbf{R}$). However, unlike the static case in which we always centred our ball of density on the origin in state space, the centre of the state space ball now "flows" from time step to time step. The flow is according to the field described by the eigenvalues and eigenvectors of the matrix $\mathbf{A}$. We move to a new point according to this flow field, then we centre our ball on that point and pick a new state. From this new state we again flow to a new point and

then apply noise. If $\mathbf{A}$ is the identity matrix (not the zero matrix), then the "flow" does not move us anywhere and the state just evolves according to a random walk of the noise set by $\mathbf{Q}$.

B.2.6 Discrete state linear Gaussian models

We now consider a simple modification of the basic continuous state model in which the state at any time takes on one of a finite number of discrete values. Many real world processes, especially those which have distinct modes of operation, are better modeled by internal states that are not continuous. (It is also possible to construct models that have a mixed continuous and discrete state, see for example [72, 133].) The state evolution is still first-order Markovian dynamics and the observation process is still linear with additive Gaussian noise. The modification involves the use of the *winner-take-all* nonlinearity $\mathbf{WTA}[\cdot]$ defined such that $\mathbf{WTA}[\mathbf{x}]$ for any vector $\mathbf{x}$ is a new vector with unity in the position of the largest coordinate of the input and zeros in all other positions. The discrete state generative model is now simply:

$$\mathbf{x}_{t+1} = \mathbf{WTA}[\mathbf{A}\mathbf{x}_t + \mathbf{w}_t] = \mathbf{WTA}[\mathbf{A}\mathbf{x}_t + \mathbf{w}_\bullet] \tag{B.26a}$$

$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v}_\bullet \tag{B.26b}$$

where $\mathbf{A}$ is no longer known as the *state transition matrix* (although we will see said matrix shortly). As before, the k-vector $\mathbf{w}$ and p-vector $\mathbf{v}$ are temporally white and spatially Gaussian distributed noises independent of each other and of $\mathbf{x}$ and $\mathbf{y}$. The initial state $\mathbf{x}_1$ is generated in the obvious way:

$$\mathbf{x}_1 = \mathbf{WTA}[\mathcal{N}(\mu_1, \mathbf{Q}_1)] \tag{B.27}$$

(though we will soon see that without loss of generality $\mathbf{Q}_1$ can be restricted to be the identity matrix). This discrete state generative model is illustrated in figure B.3.

Static data modeling: mixtures of Gaussians and vector quantization

Just as in the continuous state model we can consider situations in which there is no natural ordering to our data, and so set the matrix $\mathbf{A}$ to be the zero matrix. In this discrete state case the

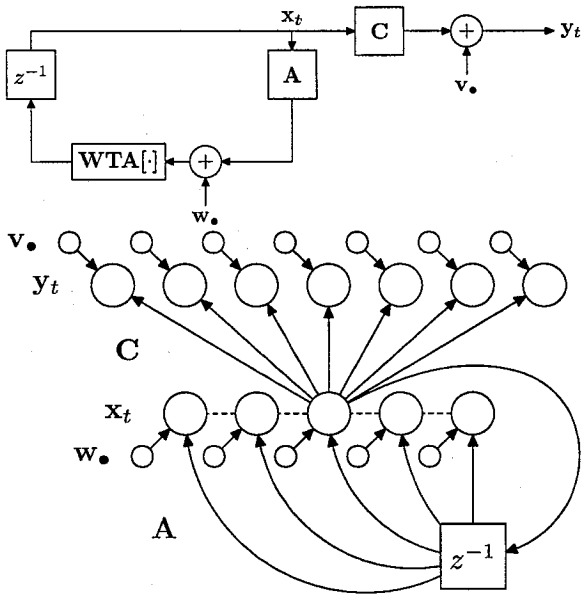


Figure B.3: Discrete state generative model for dynamic data. The $\mathbf{WTA}[\cdot]$ block implements the winner-take-all nonlinearity. The z^{-1} block is a unit delay. The covariance matrix of the input noise $\mathbf{w}$ is $\mathbf{Q}$ and the covariance matrix of the output noise $\mathbf{v}$ is $\mathbf{R}$. In the network model below, the smaller circles represent noise sources and the hidden units $\mathbf{x}$ have a winner-take-all behaviour (indicated by dashed lines). Outgoing weights have only been drawn from one hidden unit. This model is equivalent to a hidden Markov model with tied output covariances.

generative model becomes:

$$\mathbf{A} = \mathbf{0} \quad \Rightarrow \quad \mathbf{x}_\bullet = \mathbf{WTA}[\mathbf{w}_\bullet] \qquad \mathbf{w}_\bullet \sim \mathcal{N}(\boldsymbol{\mu}, \mathbf{Q}) \tag{B.28a}$$

$$\mathbf{y}_\bullet = \mathbf{C}\mathbf{x}_\bullet + \mathbf{v}_\bullet \qquad \mathbf{v}_\bullet \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \tag{B.28b}$$

Each state $\mathbf{x}_\bullet$ is generated independently[16] according to a fixed discrete probability histogram controlled by the mean and covariance of $\mathbf{w}_\bullet$. Specifically, $\pi_j = P(\mathbf{x}_\bullet = \mathbf{e}_j)$ is the probability assigned by the Gaussian $\mathcal{N}(\boldsymbol{\mu}, \mathbf{Q})$ to the region of k-space in which the j^{th} coordinate is larger than all others. (Here $\mathbf{e}_j$ is the unit vector along the j^{th} coordinate direction.) Notice that to obtain non-uniform priors π_j with the $\mathbf{WTA}[\cdot]$ nonlinearity, we require a non-zero mean $\boldsymbol{\mu}$ for the noise $\mathbf{w}_\bullet$. Once the state has been chosen, the corresponding output $\mathbf{y}_\bullet$ is generated from a Gaussian whose mean is the j^{th} column of $\mathbf{C}$ and whose covariance is $\mathbf{R}$. This is exactly the standard mixture of Gaussian clusters model except that the covariances of all the clusters are constrained to be the same. The probabilities $\pi_j = P(\mathbf{x}_\bullet = \mathbf{e}_j)$ correspond to the mixing coefficients of the clusters and the columns of $\mathbf{C}$ are the cluster means. Constraining $\mathbf{R}$ in various ways corresponds to constraining the shape of the covariance of the clusters. This model is illustrated in figure B.4 below.

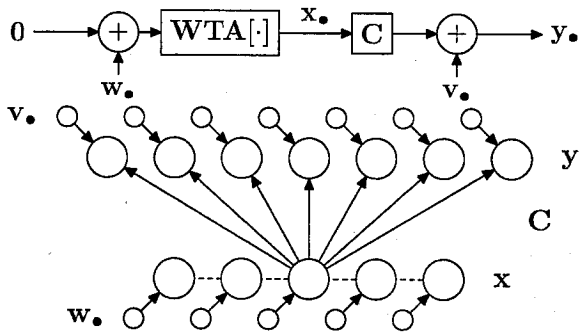


Figure B.4: Static generative model (discrete state). The $\mathbf{WTA}[\cdot]$ block implements the winner-take-all nonlinearity. The covariance matrix of the input noise $\mathbf{w}$ is $\mathbf{Q}$, and the covariance matrix of the output noise $\mathbf{v}$ is $\mathbf{R}$. In the network model below, the smaller circles represent noise sources and the hidden units $\mathbf{x}$ have a winner-take-all behaviour (indicated by dashed lines). Outgoing weights have only been drawn from one hidden unit. This model is equivalent to a mixture of Gaussian clusters with tied covariances $\mathbf{R}$ or to vector quantization (VQ) when $\mathbf{R} = \lim_{\epsilon \to 0} \epsilon \mathbf{I}$.

To compute the likelihood of a data point, we can explicitly perform the sum equivalent to the

[16] As in the continuous static case we again dispense with any special treatment of the initial state.

integral (B.14) since it contains only k terms:

$$P(\mathbf{y}_\bullet) = \sum_{i=1}^{k} P(\mathbf{x}_\bullet = \mathbf{e}_j, \mathbf{y}_\bullet) = \sum_{i=1}^{k} \mathcal{N}(\mathbf{C}_i, \mathbf{R})\,|_{\mathbf{y}_\bullet}\, P(\mathbf{x}_\bullet = \mathbf{e}_i) = \sum_{i=1}^{k} \mathcal{N}(\mathbf{C}_i, \mathbf{R})\,|_{\mathbf{y}_\bullet}\, \pi_i \tag{B.29}$$

where $\mathbf{C}_i$ denotes the i^{th} column of $\mathbf{C}$. Again, all varieties of inference and filtering are the same and we are simply seeking the set of discrete probabilities $P(\mathbf{x}_\bullet = \mathbf{e}_j|\mathbf{y}_\bullet)\ j = 1 \ldots k$. In other words, we need to do probabilistic classification. The problem is easily solved by computing the *responsibilities* $\hat{\mathbf{x}}_\bullet$ that each cluster has for the data point $\mathbf{y}_\bullet$:

$$(\hat{\mathbf{x}}_\bullet)_j = P(\mathbf{x}_\bullet = \mathbf{e}_j|\mathbf{y}_\bullet) = \frac{P(\mathbf{x}_\bullet = \mathbf{e}_j, \mathbf{y}_\bullet)}{P(\mathbf{y}_\bullet)} = \frac{P(\mathbf{x}_\bullet = \mathbf{e}_j, \mathbf{y}_\bullet)}{\sum_{i=1}^{k} P(\mathbf{x}_\bullet = \mathbf{e}_i, \mathbf{y}_\bullet)} \tag{B.30a}$$

$$(\hat{\mathbf{x}}_\bullet)_j = \frac{\mathcal{N}(\mathbf{C}_j, \mathbf{R})\,|_{\mathbf{y}_\bullet}\, P(\mathbf{x}_\bullet = \mathbf{e}_j)}{\sum_{i=1}^{k} \mathcal{N}(\mathbf{C}_i, \mathbf{R})\,|_{\mathbf{y}_\bullet}\, P(\mathbf{x}_\bullet = \mathbf{e}_i)} = \frac{\mathcal{N}(\mathbf{C}_j, \mathbf{R})\,|_{\mathbf{y}_\bullet}\, \pi_j}{\sum_{i=1}^{k} \mathcal{N}(\mathbf{C}_i, \mathbf{R})\,|_{\mathbf{y}_\bullet}\, \pi_i} \tag{B.30b}$$

The mean $\hat{\mathbf{x}}_\bullet$ of the state vector given a data point is exactly the vector of responsibilities for that data point. This quantity defines the entire posterior distribution of the discrete hidden state given the data point. As a measure of the randomness or uncertainty in the hidden state, one could evaluate the entropy or *normalized entropy*[17] of the discrete distribution corresponding to $\hat{\mathbf{x}}_\bullet$. Although this may seem related to the variance of the posterior in factor analysis, this analogy is deceptive. Since $\hat{\mathbf{x}}_\bullet$ defines the entire distribution, no other "variance" measure is needed. Learning consists of finding the cluster means (columns of $\mathbf{C}$), the covariance $\mathbf{R}$, and the mixing coefficients π_j. This is easily done with *EM*and corresponds exactly to maximum likelihood competitive learning [48, 140] except that all the clusters share the same covariance. Later we introduce extensions to the model which remove this restriction.

Vector quantization

As in the continuous state case, we can consider the limit as the observation noise becomes infinitessimal compared to the scale of the data. What results is the standard *vector quantization* or VQ model. Vector Quantization (VQ) is a technique in which we try to find a few "prototype" vectors with which we can represent well all the vectors in a particular dataset. The name comes from an analogy to scalar quantization in which we try to find a few discrete levels with which we can

[17] The entropy of the distribution divided by the logarithm of k so that it always lies between zero and one.

represent well all the amplitudes that a signal takes on. The basic assumption inherent in using VQ is that the vectors in the set are in some way "clustered" or grouped together in the space. Thus, if we approximate each vector by the centre of the cluster that it is nearest to, then we have not incurred too much approximation error. Approximation error is defined as the distance from each vector to its best approximation, i.e. its nearest prototype. Notice that this requires a well defined notion of **distance between vectors**. Figure B.5 show cartoon examples of well clustered and unclustered datasets in two dimensions. The number of prototypes, or cluster centres, is known

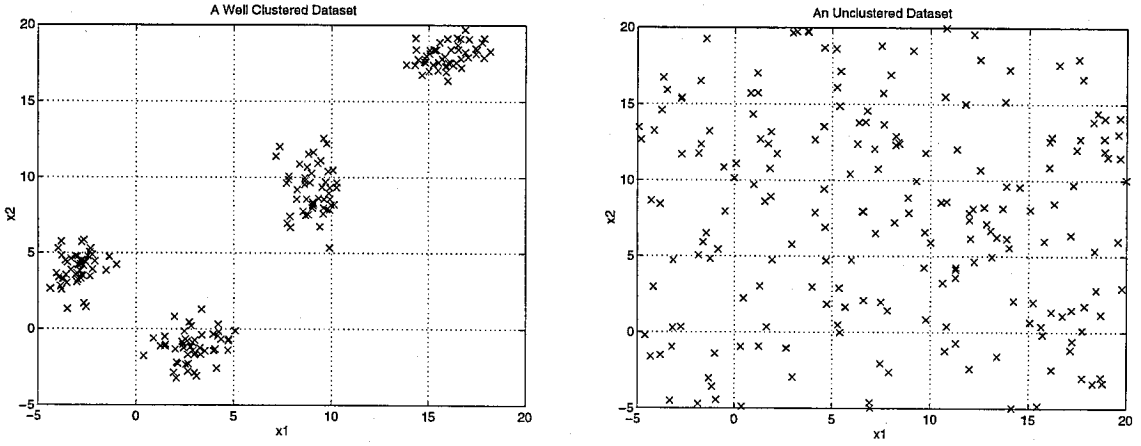


Figure B.5: Well clustered and unclustered datasets. Vector quantization will be successful on the clustered dataset but not on the unclustered one.

as the *codebook size* and is under the control of the designer. In the dataset of figure B.5 it is clear that four would be a good choice, but it is not always easy to tell especially with high-dimensional datasets. In practice one often ends up picking the codebook size by cross validation of likelihood or error to discover which size works best.

Vector quantization can be used both for coding of data and for patten classification. The idea of coding with VQ is that to encode a large dataset, we need first to describe the locations of the prototype vectors (cluster centres) and then each vector can be very compactly described by just the index of the cluster to which it belongs and the small deviation away from that prototype. The set of all the prototype vectors for a particular dataset is known as the *codebook* for that dataset. The idea of pattern classification with VQ is to have several vector quantizers, each of which were trained on examples from its own class. Given a new vector, the class of vector quantizer which does the best job encoding it (in the sense that the approximation error of the closest prototype in the vector quantizer is smallest) is the most likely class to which to the vector belongs.

The inference (classification) problem is now solved by the *one-nearest-neighbour* rule, using Euclidean distance if $\mathbf{R}$ is a multiple of the identity matrix, or Mahalanobis distance in the unscaled matrix $\mathbf{R}$ otherwise. Similarly to PCA, since the observation noise has disappeared, the posterior collapses to have all of its mass on one cluster (the closest) and the corresponding uncertainty (entropy) becomes zero.

Vector Quantizer design/learning

Learning with *EM*is equivalent to using a batch version of the *k-means* algorithm such as that proposed by Lloyd (see [123] but the algorithm was first introduced in an unpublished Bell Labs technical report earlier). As with PCA, vector quantization does not define a proper density in the observation space. Once again, we examine the sum squared deviation of each point from its closest cluster centre as a quantity proportional to the likelihood in the limit of zero noise. Batch *k-means* algorithms minimize this cost in lieu of maximizing a proper likelihood. *Llyod's algorithm* or *k-means* training proceeds as follows:

1) Given a set of N vectors design a codebook of C prototypes.
   Initialize the C prototypes to be equal to C randomly chosen vectors from the pool of N.

2) For each of the N vectors in the set, label it with the number of the prototype vector *closest* to it.

3) Replace each prototype vector with the *average* of all the vectors in the set labelled with its number.

4) Compute the total quantization error of the set of N vectors under the current codebook. If it has gone down, go to (2), otherwise terminate.

The algorithm is relatively straightforward to implement and requires only routines to implement *closest* and *average* in the steps above. In practice, one usually sets some threshold for error change in step 4 rather than waiting for the error to actually reach a stable minimum, although Llyod showed that this procedure is guaranteed to always converge to a local minimum of distortion.

Time Series Modeling: Hidden Markov Models

We return now to the fully dynamic discrete state model introduced above:

$$\mathbf{x}_{t+1} = \mathbf{WTA}[\mathbf{A}\mathbf{x}_t + \mathbf{w}_t] = \mathbf{WTA}[\mathbf{A}\mathbf{x}_t + \mathbf{w}_\bullet] \qquad \mathbf{w}_\bullet \sim \mathcal{N}(\mathbf{0}, \mathbf{Q}) \tag{B.26a}$$

$$\mathbf{y}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v}_t = \mathbf{C}\mathbf{x}_t + \mathbf{v}_\bullet \qquad \mathbf{v}_\bullet \sim \mathcal{N}(\mathbf{0}, \mathbf{R}) \tag{B.26b}$$

Our key observation is that the dynamics described by equation (B.26a) are exactly equivalent to the more traditional discrete Markov chain dynamics using a state transition matrix $\mathbf{T}$, where

$$\mathbf{T}_{ij} = P(\mathbf{x}_{t+1} = \mathbf{e}_j | \mathbf{x}_t = \mathbf{e}_i) \tag{B.31}$$

It is easy to see how to compute the equivalent state transition matrix $\mathbf{T}$ given $\mathbf{A}$ and $\mathbf{Q}$ above: $\mathbf{T}_{ij}$ is the probability assigned by the Gaussian whose mean is the i^{th} column of $\mathbf{A}$ (and whose covariance is $\mathbf{Q}$) to the region of k-space in which the j^{th} coordinate is larger than all others. It is also true that for any transition matrix $\mathbf{T}$ (whose rows each sum to unity) there exist matrices $\mathbf{A}$ and $\mathbf{Q}$ such that the dynamics are equivalent.[18] Similarly, the initial probability mass function for $\mathbf{x}_1$ is easily computed from $\boldsymbol{\mu}_1$ and $\mathbf{Q}_1$ and for any desired histogram over the states for $\mathbf{x}_1$ there exist a $\boldsymbol{\mu}_1$ and $\mathbf{Q}_1$ which achieve it.

Similar degeneracy exists in this discrete state model as in the continuous state model except that it is now between the structure of $\mathbf{A}$ and $\mathbf{Q}$. Since for *any* noise covariance $\mathbf{Q}$ the means in the columns of $\mathbf{A}$ can be chosen to set any equivalent transition probabilities $\mathbf{T}_{ij}$, we can without loss of generality restrict $\mathbf{Q}$ to be the identity matrix and use only the means in the columns of $\mathbf{A}$ to set probabilities. Equivalently, we can restrict $\mathbf{Q}_1 = \mathbf{I}$ and use only the mean $\boldsymbol{\mu}_1$ to set the probabilities for the initial state $\mathbf{x}_1$.

Thus, this generative model is equivalent to a standard hidden Markov model except that the emission probability densities are all constrained to have the same covariance. Likelihood and filtering computations are performed with the so-called *forward* (alpha) recursions, while complete smoothing is done with the *forward-backward* (alpha-beta) recursions. The *EM*algorithm for

[18] Although harder to see. Sketch of proof: Without loss of generality, always set the covariance to the identity matrix. Next, set the dot product of the mean vector with the k-vector having unity in all positions to be zero since moving along this direction does not change the probabilities. Now there are $(k-1)$ degrees of freedom in the mean and also in the probability model. Set the mean randomly at first (except that it has no projection along the all unity direction). Move the mean along a line defined by the constraint that all probabilities but two should remain constant until one of those two probabilities has the desired value. Repeat this until all have been set correctly.

learning is exactly the well known Baum-Welch re-estimation procedure [12, 13]. See chapter 7 for a detailed review of these HMM algorithms.

There is an important and peculiar consequence of discretizing the state which impacts the smoothing problem. The state *sequence* formed by taking the most probable state of the posterior distribution at each time (as computed by the forward-backward recursions given the observed data and model parameters) is *not* the single state sequence most likely to have produced the observed data. In fact, the sequence of states obtained by concatenating the states, which individually have maximum posterior probability at each time step, may have zero probability under the posterior. This creates the need for separate inference algorithms to find the single most likely state sequence given the observations. Such algorithms for filtering and smoothing are called *Viterbi decoding* methods [199]. Why was there no need for similar decoding in the continuous state case? It turns out that due to the smooth and unimodal nature of the posterior probabilities for individual states in the continuous case (all posteriors are Gaussian), the sequence of maximum a posteriori states *is exactly* the single most likely state trajectory. So the regular Kalman filter and RTS smoothing recursions suffice. It is possible (see for example [154]) to learn the discrete state model parameters based on the results of the Viterbi decoding instead of the forward-backward smoothing — in other words to maximize the joint likelihood of the observations and the single most likely state sequence rather than the total likelihood summed over all possible paths through state space.

B.2.7 Comments and extensions

There are several advantages, both theoretical and practical, to a unified treatment of the unsupervised methods reviewed in this note. From a theoretical viewpoint the treatment emphasizes that all of the techniques for inference in the various models are essentially the same and just correspond to probability propagation in the particular model variation. Similarly, all the learning procedures are nothing more than an application of the *EM*algorithm to iteratively increase the total likelihood of the observed data. Furthermore, the origin of zero noise limit algorithms such as VQ and PCA are easy to see. A unified treatment also highlights the relationship between similar questions across the different models. For example, picking the number of clusters in a mixture model or state dimension in a dynamical system or the number of factors or principal components in a covariance model or the number of states in a hidden Markov model are all really the same question.

From a practical standpoint, a unified view of these models allows us to apply well known solutions to hard problems in one area to similar problems in another. For example, in this framework it is obvious how to deal properly with missing data in solving both the learning and inference problems. This topic has been well understood for many static models [121, 194, 74] but is typically not well addressed in the linear dynamical systems literature. As another example, it is easy to design and work with models having a mixed continuous and discrete state vector (such as for example Hidden Filter HMMs [66]) which is something not directly addressed by the individual literatures on discrete or continuous models.

Another practical advantage is the ease with which natural extensions to the basic models can be developed. For example, using the hierarchical mixture of experts formalism developed by Jordan and Jacobs (1994) , one can consider global mixtures of any of the model variants discussed. In fact, most of these mixtures have already been considered: mixtures of linear dynamical systems are known as *switching state-space models* (see [173, 72] and references therein); mixtures of factor analyzers [73] and of pancakes (PCA) [83]; mixtures of hidden Markov models [178]. A mixture of m of our constrained mixtures of Gaussians each with k clusters gives a mixture model with mk components in which there are only m possible covariance matrices. This "tied covariance" approach is popular in speech modeling to reduce the number of free parameters. (For $k = 1$ this corresponds to a full "unconstrained" mixture of Gaussians model with m clusters.)

It is also possible to consider "local mixtures" in which the conditional probability $P(\mathbf{y}_t|\mathbf{x}_t)$ is no longer a single Gaussian but a more complicated density such as a mixture of Gaussians. For our (constrained) mixture of Gaussians model this is another way to get a "full" mixture. For hidden Markov models this is a well known extension and is usually the standard approach for emission density modeling [154]. It is even possible to use constrained mixture models as the output density model for a HMM (see for example [169] which uses factor analysis as the HMM output density). However, I am not aware of any work that considers this variation in the continuous state cases, either for static or dynamic data.

Another important natural extension is *spatially adaptive observation noise*. The idea here is that the observation noise $\mathbf{v}$ can have different statistics in different parts of (state or observation) space rather than being described by a single matrix $\mathbf{R}$. For discrete mixture models, this idea is well known and it is achieved by giving each mixture component a private noise model. However, for continuous state models this idea is relatively unexplored, and is an interesting area for further investigation. The crux of the problem is how to parameterize a positive definite matrix over some

space. I propose some simple ways to achieve this. One possibility is replacing the single covariance shape $\mathbf{Q}$ for the observation noise with a *conic*[19] linear blending of k "basis" covariance shapes. In the case of linear dynamical systems or factor analysis, this amounts to a novel type of model in which the local covariance matrix $\mathbf{R}$ is computed as a conic linear combination of several "canonical" covariance matrices through a tensor product between the current state vector $\mathbf{x}$ (or equivalently the "noiseless" observation $\mathbf{Cx}$) and a master noise tensor $\mathcal{R}$.[20] Another approach would be to drop the conic restriction (allow general linear combinations) and then add a multiple of the identity matrix to the resulting noise matrix in order to make it positive definite. A third approach is to represent the covariance shape as the compression of an elastic sphere due to a spatially varying force field. This representation is easier to work with because the parameterization of the field is unconstrained, but it is hard to learn the local field from measurements of the effective covariance shape. Bishop (1995, section 6.3) and others have considered simple non-parametric methods for estimating input dependent noise levels in regression problems. Goldberg, Williams and Bishop (1998) have also explored this idea in the context of Gaussian processes.

It is also interesting to consider what happens to the *dynamic* models when the output noise tends to zero. In other words, what are the dynamic analogues of PCA and VQ? For both linear dynamical systems and hidden Markov models, this causes the state to no longer be hidden. In linear dynamical systems the optimal observation matrix is then found by performing PCA on the data and using the principal components as the columns of $\mathbf{C}$; for hidden Markov models $\mathbf{C}$ is found by vector quantization of the data (using the codebook vectors as the columns of $\mathbf{C}$). Given these observation matrices the state is no longer hidden. All that remains is to identify a first-order Markov dynamics in state space: this is a simple AR(1) model in the continuous case or a first order Markov chain in the discrete case. Such zero-noise limits are not only interesting models in their own right, but are also valuable as good choices for initialization of learning in linear dynamical systems and hidden Markov models.

[19] A conic linear combination is one in which all the coefficients are positive.

[20] For mixtures of Gaussians or hidden Markov models, this kind of linear "blending" merely selects the j^{th} submatrix of the tensor if the discrete state is $\mathbf{e}_j$. This is yet another way to recover the conventional "full" or unconstrained mixture of Gaussians or hidden Markov model emission density in which each cluster or state has its own private covariance shape for observation noise.